

Data Structer

COMP 242

د. س. س. س.

III Hand shake problems

*
$$h(n) = \begin{cases} \text{base} \\ 1 & , n \leq 2 \\ h(n-1) + (n-1) & , n > 2 \end{cases}$$

Recursive Pairs

* هذا - اذ من كثرة اقوالهم

Sum (into n) 9

if $(n == 1)$

return 13

else

return $n_1 + \text{Sum}(n-1);$

صلى: حسب ال Suma للحد الذي يليه

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

* بجواب رقم

100

$$n + \text{Sum}(n-1) = \sum_{i=1}^{100} i = 1 + 2 + 3 + \dots + 100$$

10

$$1 + \text{Sum}(n-1) = 1 + \text{Sum}(9)$$

$$1 + \text{Sum}(8)$$

$$1 + \text{Sum}(7)$$

⋮

31 Factorial

Public Static long Fact(int n) {

if (n == 0)

return 1;

else

return n * Fact(n-1);

}

$$F(n) = \begin{cases} 1 & , n == 0 \\ n * \text{Fact}(n-1) & , n > 0 \end{cases}$$

* يا إما بخرطين الكود وبدء الفاكشن أو الحالتين .

[4] Fibonacci:

0 1 1 2 3 5 8 13 21 ...

$$\text{Fib}(n) = \begin{cases} 0 & , n=0 \\ 1 & , n=1 \\ \text{Fib}(n-1) + \text{Fib}(n-2) & , n > 1 \end{cases}$$

```
Public static int Fib(int n){
```

```
    if (n == 0)
```

```
        return 0;
```

```
    if (n == 1)
```

```
        return 1;
```

```
    else
```

```
        return Fib(n-1) + Fib(n-2);
```

Fib(10) →

= 55

^
Fib(8) Fib(9)

^

^
Fib(6) Fib(7)

^

^

^

[5] Reverse

321 ← 123 : قبل الـ reverse

```
static int sum = 0;
```

```
public static int Reverse (int n) {
```

```
    if (n == 0)
```

```
        return sum;
```

```
    else
```

```
        sum = sum * 10 + n % 10;
```

```
        Reverse n / 10;
```

```
}
```

[6] Count Char

يوجد عدد ا حروف .

```
public static int countChar (String str, char s) {
```

```
    if (str.length() == 0)
```

```
        (str.isEmpty())
```

يقدر استبدالها بـ

```
        return 0;
```

```
    else if (str.charAt(0) == s)
```

```
        return 1 + countChar (str.substring(1), s);
```

```
    else
```

```
        return countChar (str.substring(1), s);
```

```
}
```

[7] Count String

يوجد كلمة بوا حروف .

```
int countString (String str, String s) {
```

```
    if (str.length() < s.length())
```

```
        return 0;
```

```
    if (str.substring(0, s.length()).equals(s))
```

```
        return 1 + countString (str.substring(1), s);
```

```
    else
```

```
        return countString (str.substring(1), s);
```

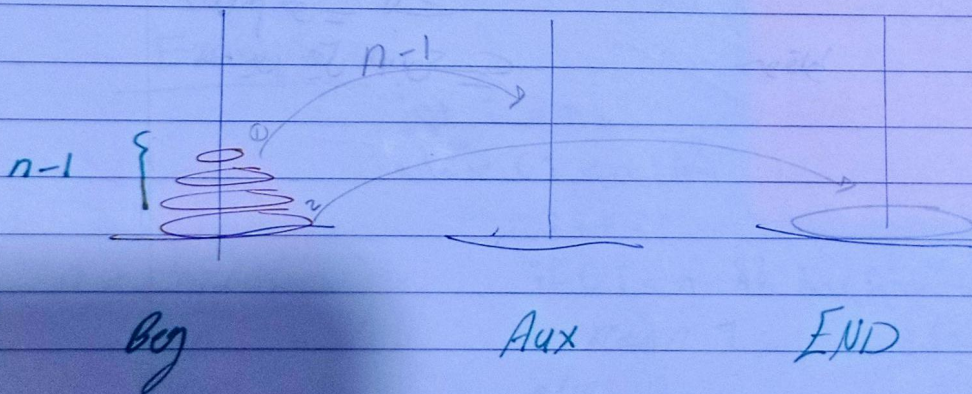

[8] Revers String.

```
public static String Reverse (String str) {
    if (str.isEmpty())
        return str;
    else
        return Reverse (str.substring(1)) +
            str.charAt(0);
}
```

nam olleh ← Hello man

[9] Hori Tower.

```
HoriTower (int n, char Beg, char Aux, char END) {
    if (n > 0) {
        HoriTower (n-1, Beg, end, aux);
        System.out.printf ("Mov. disk %d from %c to %c\n",
            n, Beg, END);
        HoriTower (n-1, Aux, Beg, END);
    }
}
```



Palindrome

```
public static boolean isPali (int [] arr, int i, int j){
```

```
    if (i > j)
```

```
        return True ;
```

```
    else if (arr [i] != arr [j] )
```

```
        return false ;
```

```
    else
```

```
        return is Pali (arr, ++i, --j ) ;
```

```
}
```

* Done

بسم الله الرحمن الرحيم

اللهم قوِّ بكَ
(اللَّهُمَّ اِنْفِضْ
الْوَقْتِمْ - ٥)

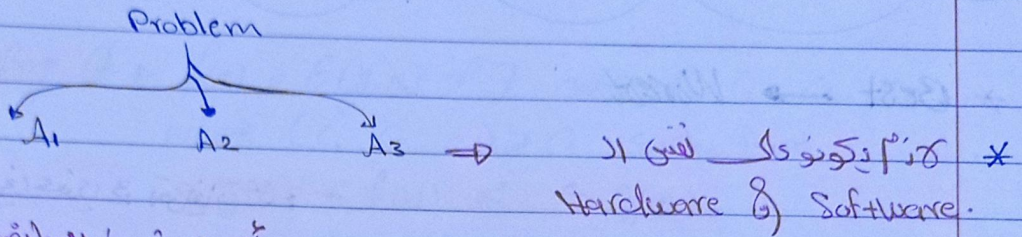
اللهم قوِّ بكَ
حين نقول آمين

Analysis of Algorithm

Chapter 2

(OS - AOA)

Problem Solving :-



Algorithm (نفسه في الـ Algorithm)
Speed (قدرة الأداء).
Space.

عندما نعرف أنو الحل والـ Algorithm المقبول أقرب

Experiments → implement

timing (time)

Plotting (نصفه في الـ Algorithm)

Asymptotic → $T(n)$

All possibilities.

Example :-

```
int i = 0;
while (i < n && Arr[i] != key) {
    i++;
    if (i < n && Arr[i] == key)
        return 1;
    else
        return 0;
}
```

n = element
in
Array

Best case $\rightarrow 1$: 3 عنصری
 Average case \rightarrow Array element ای
 Worst case $\rightarrow N$

$$1 \leq \log n \leq n \leq n \log n \leq n^2 \leq 2^n \leq \dots$$

$$n! \leq \dots \leq n^n$$

* Best \rightarrow Worst

: 3 آریفاد

Big - O \rightarrow upper bound
 Big - Ω \rightarrow lower bound
 Big - Θ \rightarrow Average bound.

$T(n) = O(g(n))$ if there exist constant C and n_0 , such that $T(n) \leq Cg(n)$ where $C \geq 0, n_0 \geq 0$

$$T(n) = \frac{n(n-1)}{2} = \frac{n^2 - n}{2} \leq \frac{1}{2} n^2 \leq n^2$$

? $O(n^2)$

* تا وقت ای
order

Big O : $f(n) = O(g(n))$ if \exists constant C, n_0 such $\forall C \geq 0, n_0 \geq 0$ where $f(n) \leq Cg(n)$.

Big Ω : $f(n) = \Omega(g(n))$, if \exists constant C, n_0 such $\forall C \geq 0, n_0 \geq 0$ where $f(n) \geq Cg(n)$

Big Θ : $f(n) = \Theta(g(n))$ if \exists constant C_1, C_2, n_0 where $(C_1, C_2, n_0) \geq 0$ such that $C_1 g(n) \leq f(n) \leq C_2 g(n)$.

Ex $f(n) = 2n + 3$
 $= f(n) = O(n)$

$$f(n) \leq Cg(n)$$

$$2n + 3 \leq 1n + 3n \quad \checkmark$$

$$2n + 3 \leq 5n$$

Constant 5 is least \checkmark

$$\therefore f(n) = \Omega(n)$$

$$f(n) \geq Cg(n)$$

$$2n + 3 \geq 1n \quad \checkmark$$

$$\therefore f(n) = \Theta(n)$$

$$C_1 g(n) \leq f(n) \leq C_2 g(n)$$

$$1n \leq 2n + 3 \leq 5n$$

Ex $n^2 \log n.$

$$f(n) = O(n^2 \log n)$$

$$f(n) \leq c g(n)$$

$$n^2 \log n + c \leq 10 n^2 \log n.$$

$$f(n) = \Omega(n^2 \log n)$$

$$f(n) \geq c g(n)$$

$$n^2 \log n + n \geq n^2 \log n.$$

$$f(n) = \Theta(n^2 \log n).$$

Ex $f(n) = n!$

$$1 \leq 1 \times 2 \times 3 \times \dots \times (n-1) \times n < n^n$$

$$f(n) = O(n^n)$$

lower

upper

$$f(n) = \Omega(1)$$

$$f(n) = \Theta \text{ unknown.}$$

$$f(n) = n^2 + n$$

$$g(n) = n \log n.$$

[1] $f(n) + g(n) = \text{Max}(f(n), g(n))$
 $(n^2 + n.)$

??

[2] $f(n) * g(n) = O(f(n) * g(n)) (O(n^3 \log n).)$

$$\log n! \rightarrow O(n \log n)$$

$$f(n) = n^2 \log(n)$$

$$\log n^2 \log(n)$$

$$\log n (\log n)^{10}$$

$$g(n) = n (\log n)^{10}$$

$$\frac{2 \log n + \log n \log n}{n^2} > \frac{\log n + 10 \log n \log n}{10n}$$

$$f(n) = 3n^{\sqrt{n}}$$

$$3n^{\sqrt{n}}$$

$$2^{\sqrt{n} \log n}$$

$$a^{\log_c b}$$

$$g(n) = 2^{\sqrt{n} \log n}$$

$$3n^{\sqrt{n}}$$

$$= (n^{\sqrt{n}})^{\log_2 3}$$

$$= b^{\log_c a}$$

$$f(n) = 2^{\log n}$$

$$\log 2^{\log n}$$

$$\log n^{\sqrt{n}}$$

$$g(n) = n^{\sqrt{n}}$$

$$\log(n) (\log_2 2)$$

$$<$$

$$\sqrt{n} \log n$$

$$f(n) = 2^n$$

$$\log 2^n$$

$$\log 2^{2n}$$

$$g(n) = 2^{2n}$$

$$n \log_2 2$$

$$\log_2 2^{2n}$$

$$\frac{n \log_2 2}{2}$$

$$<$$

$$\frac{2n \log_2 2}{2}$$

$$f(n) = n^{\log n}$$

$$g(n) = 2^n$$

$$\log 2^n$$

$$\log n^{\log n}$$

$$n \log_2 2$$

$$(\log n) (\log n)$$

$$\log n$$

$$\log (\log(n))^2$$

$$\log n$$

$$<$$

$$2 \log (\log n)$$

Sum (A, n) {

$\delta = 0$; ①

 for ($i = 0$; $i \leq n$; $i++$) ③ $2n+2$

$\delta = \delta + A[i]$; ② $2n$

 return δ ; ④

$$\begin{aligned} \text{Big } O(n) &= 2n+2 + 2n \\ &= 4n+2 \\ &= \underline{\underline{O(n)}} \end{aligned}$$

Time Complexity

$\delta = 0$ ①

$i = 0$ ①

Array $\underline{\underline{O(n)}}$

Space Complexity

variable δ & i

Ram. δ & i are stored in memory

Add (A, B, n) {

(n+1) for ($i = 0$; $i \leq n$; $i++$) ①

$n(n+1)$ for ($j = 0$; $j \leq n$; $j++$) ①

$\{ \quad \quad \quad \frac{n^2}{\quad} \quad \quad \frac{n^2}{\quad} \quad \quad \frac{n^2}{\quad} \quad \quad \}$ $C[i][j] = A[i][j] + B[i][j];$

Time

$$\begin{aligned} O(n) &= 2n^2 + 2n + 1 \\ &= \underline{\underline{O(n^2)}} \end{aligned}$$

Space. $3n^2 + 2$

$$\underline{\underline{O(n^2)}}$$

Multi (A, B, n) {

$n+1$ For ($i=0$; $i \leq n$; $i++$)

$n(n+1)$ For ($j=0$; $j \leq n$; $j++$)

$n \times n (n+1)$ For ($k=0$; $k \leq n$; $k++$)

$n \times n \times n$ $C = C + A + B;$
}

Time = $O(n^3)$

Space = $O(n^2)$. ?

for (int i=0; i ≤ n; i += 2) { i = 0, 2, 4, 6, ... k
 Statement;
 } Stop in i > n
 k > n

* increment and decrement
 same time O() = O(n).

for (int i=1; i ≤ n; i *= 2) { i = 2⁰, 2¹, ... 2^k
 Statement;
 } Stop in i > n
 log 2^k > n log
 ∴ O(log n). ← k > log n

for (int i=n; ~~i > 1~~ i ≥ 1; i /= 2) { i
 Statement;
 } Stop in i < 1
 log $\frac{n}{2^k} < 1$ log
 ∴ O(log n) $\frac{n}{2^k} \rightarrow \frac{n}{2^0}$
 $\frac{n}{2} \rightarrow \frac{n}{2^1}$
 $\frac{n}{m} \rightarrow \frac{n}{2^k}$

O(n) ← ضرب / قسمة

O(log n). ← ضرب / قسمة

*


```

P = 0;
for (i = 1; i < n; i = i * 2) {
    P++;
}
for (j = 1; j < P; j = j * 2) {
    Statement;
}

```

Annotations: $\log(n)$ for the first loop, $\log(\log(n))$ for the second loop.

$\therefore \log(\log n)$.

```

for (i = 0; i < n; i++)
    for (j = 1; j < n; j = j * 2)
        Statement;

```

Annotations: n for the first loop, $\log n$ for the second loop. Total complexity: $\therefore O(n \log n)$.

```

P = 0;
for (int i = 1; P < n; i++) {
    P += i;
    for (int j = 2; j > 1; j = j * 2)
        Statement;
}

```

Annotations: \sqrt{n} for the first loop, $\log n$ for the second loop.

$\therefore O(\sqrt{n} \log n)$.

while if for
ليس السجدة


```
Test (int n) {
    if (n > 0) {
        print (n);
```

Recursion.

* Print Down.

```
} | Test (n-1); | } // recursive for n > 0 *
// next 11
```

$$[1] \quad T(n) = \begin{cases} 1 & n \leq 0 \\ T(n-1) + 1 & n > 0 \end{cases}$$

$$[2] \quad T(n) = T(n-1) + 1$$

$$T(n-1) = T(n-2) + 1$$

$$T(n-2) = T(n-3) + 1$$

* recursive formula

$$\begin{aligned} T(n) &= T(n-1) + 1 \\ &= T(n-2) + 1 + 1 \end{aligned}$$

\vdots

$$T(n) = T(n-k) + k$$

* Assume $n-k = 0$
 $n = k$

$$T(k) = T(0) + n$$

$$T(k) = 1 + n$$

$$\therefore O(n).$$


```
void Test (int n) {
```

```
    if (n > 0)
```

```
        for (int i = 1; i <= n; i += 2)  $\Rightarrow \log n$ 
```

```
            System.out.print(i);
```

```
        Test (n-1);
```

```
    }
```

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1) + \log n & n > 0 \end{cases}$$

$$T(n-1) = T(n-2) + \log(n-1) \quad \sim (1)$$

$$T(n-2) = T(n-3) + \log(n-2) \quad \sim (2)$$

$$T(n) = (T(n-2) + \log(n-1)) + \log n$$

$$T(n) = T(n-2) + \log(n-1) + \log n$$

$$T(n) = T(n-3) + \log(n-2) + \log(n-1) + \log n$$

⋮

$$T(n) = T(n-k) + \log(n-k-1) + \log(n-k-2) + \dots + \log n$$

Assume $k-n=0$ $T(n) = T(n-n) + \log(n-n-1) + \log(n-1) + \log n$

$k=n$ $= T(0) + \log(1) + \log(2) + \dots + \log(n-1) + \log n$

$$T(n) = 1 + \log(1 \times 2 \times 3 \times 4 \times \dots \times (n-1) \times n)$$

$$T(n) = \log n!$$

$$= O(n \log n)$$

$$T(n) = 4T(n/2) + n + 1$$

$$T(n/2) = 4T(n/2^2) + n/2 + 1 \quad \sim \textcircled{1}$$

$$T(n/2^2) = 4T(n/2^3) + n/2^2 + 1 \quad \sim \textcircled{2}$$

$$T(n) = 4T(n/2) + n + 1$$

$$= 4[4T(n/2^2) + n/2 + 1] + n + 1$$

$$= 4^2 T(n/2^2) + 2n/2 + 4 + n + 1$$

$$= 4^3 T(n/2^3) + 4^2 n/2^2 + \textcircled{4} + 2n + \textcircled{4} + n + 1$$

$$= 4^3 T(n/2^3) + 7n + \textcircled{7}$$

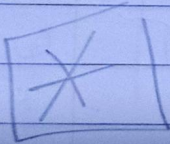
$$T(n) = 4^k T(n/2^k) + \overset{\text{Constant}}{\underline{\underline{c}}} n$$

$$\underline{\underline{2^n (1) + cn.}}$$

$$\log n = 2^k \log$$

$$\log n = k$$

Big O (2^n)



* هذا حل السؤال الأخير

في آمل من صفحتي

في التام كومبيوتر

$$\begin{aligned}
 T(n) &= T(n+1) + 1 \rightarrow O(n) \\
 &= T(n+1) + n \rightarrow O(n^2) \\
 &= T(n) + \log n \rightarrow O(n \log n) \\
 &= T(n) + n^2 \rightarrow O(n^3) \\
 &= T(n) + n \log n \rightarrow O(n^2 \log n)
 \end{aligned}$$

constant

$$T(n) = T(n-2) + 1 \rightarrow O(n)$$

$$T(n) + b \rightarrow O(nb)$$

↓
فرض متان و ثابت

algo

```

Test (int n) {
    if (n > 0) {
        Print (n);
        Test (n-1);
        Test (n-1);
    }
}

```

$$T(n) = \begin{cases} 1 & n=0 \\ 2T(n-1)+1 & n>0 \end{cases}$$

$$T(n) = 2T(n-1) + 1 \rightarrow T(n-1) = 2T(n-2) + 1 \quad (1)$$

$$T(n-2) = 2T(n-3) + 1 \quad (2)$$

$$T(n) = 2(2T(n-2) + 1) + 1$$

$$= 2^2 T(n-2) + 2 + 1 \sim \text{by (1)}$$

$$= 2^2 (2T(n-3) + 1) + 2 + 1$$

$$= 2^3 T(n-3) + 2^2 + 2^1 + 2^0 \sim \text{by (2)}$$

$$\text{Assume } n-k=0 \rightarrow \boxed{n=k}$$

$$T(n) = 2^k T(n-k) + 2^{k-1} + 2^{k-2} + \dots + 2^2 + 2^1 + 2^0$$

$$= 2^n + 2^{n-1} + 2^{n-2} + \dots + 2^0 \quad n=k \text{ steps} *$$

$$\sum_{k=0}^n a^k = \frac{1-a^{n+1}}{1-a} \Rightarrow \sum_{i=0}^n 2^i = \frac{1-2^{n+1}}{1-2} = \left\lceil \frac{2^{n+1}-1}{2-1} \right\rceil$$

$$* \left\lceil O(2^n) \right\rceil *$$

$$T(n) = 2T(n-1) + 1 \rightarrow O(2^n)$$

$$T(n) = 3T(n-1) + 1 \rightarrow O(3^n)$$

$$T(n) = 3T(n-1) + n \rightarrow O(n 3^n)$$

$$T(n) = 3T(n-1) + \log n \rightarrow O(3^n \log n)$$

```
Public static int Test(int n){
```

```
    if (n == 0)
```

```
        return 0;
```

```
    else
```

```
        return 2T(n/2) + n;
```

```
}
```

$$T(n) = \begin{cases} 1, & n=0 \\ 2T(n/2) + n, & n>0 \end{cases}$$

$$T(n/2) = 2T(n/2^2) + (n/2) \quad \text{--- (1)}$$

$$T(n/2^2) = 2T(n/2^3) + (n/2^2) \quad \text{--- (2)}$$

$$T(n) = 2T(n/2) + n$$

$$= 2(2T(n/2^2) + n/2) + n$$

$$= 2^2 T(n/2^2) + 2n/2^1 + n/2^0$$

$$= 2^2 (2T(n/2^3) + n/2^2) + 2n/2^1 + n/2^0$$

$$= 2^3 T(n/2^3) + \cancel{2^2 n / 2^2} + \cancel{2n / 2} + n/2^0$$

$$\hookrightarrow 2^3 T(n/2^3) + n$$

$$T(n) = 2^k T(n/2^k) + kn$$

$$n + n \log n$$

$$\therefore O(n \log n)$$

$$\frac{n}{2^k} = 1$$

$$\log n = 2^k \log$$

$$\log n = k$$

Ex: 10, 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, 10240, 20480, 40960, 81920, 163840, 327680, 655360, 1310720, 2621440, 5242880, 10485760, 20971520, 41943040, 83886080, 167772160, 335544320, 671088640, 1342177280, 2684354560, 5368709120, 10737418240, 21474836480, 42949672960, 85899345920, 171798691840, 343597383680, 687194767360, 1374389534720, 2748779069440, 5497558138880, 10995116277760, 21990232555520, 43980465111040, 87960930222080, 175921860444160, 351843720888320, 703687441776640, 1407374883553280, 2814749767106560, 5629499534213120, 11258999068426240, 22517998136852480, 45035996273704960, 90071992547409920, 180143985094819840, 360287970189639680, 720575940379279360, 1441151880758558720, 2882303761517117440, 5764607523034234880, 11529215046068469760, 23058430092136939520, 46116860184273879040, 92233720368547758080, 184467440737095516160, 368934881474191032320, 737869762948382064640, 1475739525896764129280, 2951479051793528258560, 5902958103587056517120, 11805916207174113034240, 23611832414348226068480, 47223664828696452136960, 94447329657392904273920, 188894659314785808547840, 377789318629571617095680, 755578637259143234191360, 1511157274518286468382720, 3022314549036572936765440, 6044629098073145873530880, 12089258196146291747061760, 24178516392292583494123520, 48357032784585166988247040, 96714065569170333976494080, 193428131138340667952988160, 386856262276681335905976320, 773712524553362671811952640, 1547425049106725343623905280, 3094850098213450687247810560, 6189700196426901374495621120, 12379400392853802748991242240, 24758800785707605497982484480, 49517601571415210995964968960, 99035203142830421991929937920, 198070406285660843983859875840, 396140812571321687967719751680, 792281625142643375935439503360, 1584563250285286751870879006720, 3169126500570573503741758013440, 6338253001141147007483516026880, 12676506002282294014967032053760, 25353012004564588029934064107520, 50706024009129176059868128215040, 101412048018258352119736256430080, 202824096036516704239472512860160, 405648192073033408478945025720320, 811296384146066816957890051440640, 1622592768292133633915780102881280, 3245185536584267267831560205762560, 6490371073168534535663120411525120, 12980742146337069071326240823050240, 25961484292674138142652481646100480, 51922968585348276285304963292200960, 103845937170696552570609926584401920, 207691874341393105141219853168803840, 415383748682786210282439706337607680, 830767497365572420564879412675215360, 1661534994731144841129758825350430720, 3323069989462289682259517650700861440, 6646139978924579364519035301401722880, 13292279957849158729038070602803445760, 26584559915698317458076141205606891520, 53169119831396634916152282411213783040, 106338239662793269832304564822427566080, 212676479325586539664609129644855132160, 425352958651173079329218259289710264320, 850705917302346158658436518579420528640, 1701411834604692317316873037158841057280, 3402823669209384634633746074317682114560, 6805647338418769269267492148635364229120, 13611294676837538538534984297270728458240, 27222589353675077077069968594541456916480, 54445178707350154154139937189082913832960, 108890357414700308308279874378165827665920, 217780714829400616616559748756331655331840, 435561429658801233233119497512663310663680, 871122859317602466466238995025326621327360, 1742245718635204932932477990050653242654720, 3484491437270409865864955980101306485309440, 6968982874540819731729911960202612970618880, 13937965749081639463459823920405225941237760, 27875931498163278926919647840810451882475520, 55751862996326557853839295681620903764951040, 111503725992653115707678591363241807529902080, 223007451985306231415357182726483615059804160, 446014903970612462830714365452967230119608320, 892029807941224925661428730905934460239216640, 1784059615882449851322857461811868920478433280, 3568119231764899702645714923623737840956866560, 7136238463529799405291429847247475681913733120, 14272476927059598810582859694494951363827466240, 28544953854119197621165719388989902727654932480, 57089907708238395242331438777979805455309864960, 114179815416476790484662877555959610910619729920, 228359630832953580969325755111919221821239459840, 456719261665907161938651510223838443642478919680, 913438523331814323877303020447676887284957839360, 1826877046663628647754606040895353774569915678720, 3653754093327257295509212081790707549139831357440, 7307508186654514591018424163581415098279662714880, 14615016373309029182036848327162830196559325429760, 29230032746618058364073696654325660393118650859520, 58460065493236116728147393308651320786237301719040, 116920130986472233456294786617302641572474603438080, 233840261972944466912589573234605283144949206876160, 467680523945888933825179146469210566289898413752320, 935361047891777867650358292938421132579796827504640, 1870722095783555735300716585876842265159593655009280, 3741444191567111470601433171753684530319187310018560, 7482888383134222941202866343507369060638374620037120, 14965776766268445882405732687014738121276749240074240, 29931553532536891764811465374029476242553498480148480, 59863107065073783529622930748058952485106996960296960, 119726214130147567059245861496117904970213993920593920, 239452428260295134118491722992235809940427987841187840, 478904856520590268236983445984471619880855975682375680, 957809713041180536473966891968943239761711951364751360, 1915619426082361072947933783937886479523423902729502720, 3831238852164722145895867567875772959046847805459005440, 7662477704329444291791735135751545918093695610918010880, 15324955408658888583583470271503091836187391221836021760, 30649910817317777167166940543006183672374782443672043520, 61299821634635554334333881086012367344749564887344087040, 122599643269271108668667762172024734689499129774688174080, 245199286538542217337335524344049469378998259549376348160, 490398573077084434674671048688098938757996519098752696320, 980797146154168869349342097376197877515993038197505392640, 1961594292308337738698684194752395755031986076395010785280, 3923188584616675477397368389504791510063972152790021570560, 7846377169233350954794736779009583020127944305580043141120, 15692754338466701909589473558019166040255888611160086282240, 31385508676933403819178947116038332080511777222320172564480, 62771017353866807638357894232076664161023554444640345128960, 125542034707733615276715788464153328322047108889280690257920, 251084069415467230553431576928306656644094217778561380515840, 502168138830934461106863153856613313288188435557122761031680, 1004336277661868922213726307713226626576376871114245522063360, 2008672555323737844427452615426453253152753742228491044126720, 4017345110647475688854905230852906506305507484456982088253440, 8034690221294951377709810461705813012611014968913964176506880, 16069380442589902755419620923411626025222029937827928353013760, 32138760885179805510839241846823252050444059875655856706027520, 64277521770359611021678483693646504100888119751311713412055040, 128555043540719222043356967387293008201776239502623426824110080, 257110087081438444086713934774586016403552479005246853648220160, 514220174162876888173427869549172032807104958010493707296440320, 1028440348325753776346855739098344065614209916020987414592880640, 2056880696651507552693711478196688131228419832041974829185761280, 4113761393303015105387422956393376262456839664083949658371522560, 8227522786606030210774845912786752524913679328167899316743045120, 16455045573212060421549691825573505049827358656335798633486090240, 32910091146424120843099383651147010099654717312671597266972180480, 65820182292848241686198767302294020199309434625343194533944360960, 131640364585696483372397534604588040398618869250686389067888721920, 263280729171392966744795069209176080797237738501372778135777443840, 526561458342785933489590138418352161594475477002745556271554887680, 1053122916685571866979180276836704323188950954005491112543109775360, 2106245833371143733958360553673408646377901908010982225086219550720, 4212491666742287467916721107346817292755803816021964450172439101440, 8424983333484574935833442214693634585511607632043928900344878202880, 16849966666969149871666884429387269171023215264087857800689756405760, 33699933333938299743333768858774538342046430528175715601379512811520, 67399866667876599486667537717549076684092861056351431202759025623040, 134799733335753198973335075435098153368185722112702862405518051246080, 269599466671506397946670150870196306736371444225405724811036102492160, 539198933343012795893340301740392613472742888450811449622072204984320, 1078397866686025591786680603480785226945485776901622899244144409968640, 2156795733372051183573361206961570453890971553803245798488288819937280, 4313591466744102367146722413923140907781943107606491596976577639874560, 8627182933488204734293444827846281815563886215212983193953155279749120, 17254365866976409468586889655692563631127772430425966387906310559498240, 34508731733952818937173779311385127262255544860851932775812621118996480, 69017463467905637874347558622770254524511089721703865551625242237992960, 138034926935811275748695117245540509049022179443407731103250484475985920, 276069853871622551497390234491081018098044358886815462206500968951971840, 552139707743245102994780468982162036196088717773630924413001937903943680, 1104279415486490205989560937964324072392177435547261848826003875807887360, 2208558830972980411979121875928648144784354871094523697652007751615774720, 4417117661945960823958243751857296289568709742189047395304015503231549440, 8834235323891921647916487503714592579137419484378094790608031006463098880, 17668470647783843295832975007429185158274838968756189581216062012926197760, 35336941295567686591665950014858370316549677937512379162432124025852395520, 70673882591135373183331900029716740633099355875024758324864248051704791040, 141347765182270746366663800059433481266198711750049516649728496103409582080, 282695530364541492733327600118866962532397423500099033299456992206819164160, 565391060729082985466655200237733925064794847000198066598913984413638328320, 1130782121458165970933310400475467850129589694000396133197827968827276656640, 2261564242916331941866620800950935700259179388000792266395655937654553313280, 4523128485832663883733241601901871400518358776001584532791311875309106626560, 9046256971665327767466483203803742801036717552003169065582623750618213253120, 18092513943330655534932966407607485602073435104006338131165247501236426506240, 36185027886661311069865932815214971204146870208012676262330495002472853012480, 72370055773322622139731865630429942408293740416025352524660990004945706024960, 144740111546645244279463731260859884816587480832050705049321980009891412049920, 289480223093290488558927462521719769633174961664101410098643960019782824099840, 578960446186580977117854925043439539266349923328202820197287920039565648199680, 1157920892373161954235709850086879078532699846656405640394575840079131296399360, 2315841784746323908471419700173758157065399693312811280789151680158262592798720, 4631683569492647816942839400347516314130799386625622561578303360316525185597440, 9263367138985295633885678800695032628261598773251245123156606720633050371194880, 18526734277970591267771357601390065256523197546502490246313213441266100742389760, 37053468555941182535542715202780130513046395093004980492626426882532201484779520, 74106937111882365071085430405560261026092790186009960985252853765064402969559040, 148213874223764730142170860811120522052185580372019921970505707530128805939118080, 296427748447529460284341721622241044104371160744039843941011415060257611878236160, 592855496895058920568683443244482088208742321488079687882022830120515223756472320, 1185710993790117841137366886488964176417484642976159375764045660241030447512944640,

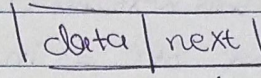
Chapter 3

Linked List

class

Node

Node → Data



header →

→ location in Memory

class Node {

private String name;

private int id;

Node next;

Operations-

- 1) insert
- 2) delete
- 3) update
- 4) Traversal.

public class Node {

private roll;

private int age;

Node next;

constructor

Node () { }

Node (int roll, int age) {

this.roll = roll;

this.age = age;

next = null; }

getter, setter

printlnf

method


```

Node header = new Node ();
File file = new File ("data.txt");
Scanner in = new Scanner (file);

```

```

while (in.hasNextLine ()) {

```

```

    String str = in.nextLine ();

```

```

    String [] data = str.split (" : ");

```

```

    insert LL (header, Integer.parseInt (data [0],
header = Integer.parseInt (data [1] );

```

```

}

```

```

}

```

insert في الباق.

```

public static Node insert LL (Node header, int roll, int age) {

```

```

    Node first = header;

```

```

    Node newnode = new Node (roll, age);

```

```

    if (first == null) {

```

```

        return newnode; header = newnode;

```

```

    }

```

```

    else {

```

```

        newnode.next = first;

```

```

        first = newnode; header = newnode;

```

```

    }

```

```

    return first;

```

```

}

```

Big O (1)

+ constant

عن ان قأك بدل method دمج

~ print List (Node header) ?

Node ptr = header ;

while (ptr != null) {

ptr.printInfo() ;

ptr = ptr.next ;

}

→ O(n).

public static Node insertAtPosition (Node header , int roll ,

int age , int key) {

Node ptr = header ;

Node newNode = new Node (roll , age) ;

if (ptr == null)

ptr = newNode ; return newNode ;

else {

while (ptr != null && ptr.getRoll() != key) {

ptr = ptr.next ;

if (ptr.getRoll() == key) {

newNode.next = ptr.next ;

ptr.next = newNode ; }

if (ptr.next == null) {

newNode.next = ptr ;

ptr = newNode ;

}

return ptr ;

Array and linked list store collection of data.

Array \Rightarrow Store all element in one block of memory.

LinkedList \Rightarrow allocate memory for each element separately and only when necessary.

Disadvantages of Array.

I] Size is fixed. \rightarrow حجم ثابت لا يمكن تغييره
حجمه ثابتاً لا يمكن تغييره
1) Array الى حجمه ليس في الذاكرة
2) 90 (30 ٪ من الذاكرة) من الذاكرة والباقي يكون فارغاً
3) لا يمكن تغيير حجمه

II] Inserting and deleting elements into the middle of array is potentially expensive because existing elements need to be shifted over to make room.

الحاجة الى تغيير حجم الـ Array
الحاجة الى تغيير حجم الـ Array
الحاجة الى تغيير حجم الـ Array
الحاجة الى تغيير حجم الـ Array

Array versus Linked list:-

Array \Rightarrow insert and delete at end.

Randomly accessing any element
searching the list for a particular value.

LinkedList \Rightarrow Insert and Delete

Application where sequential access is required
number of element can't be predicted
before hand.


```

public void insertLast (int data) {
    Node ptr = head;
    Node newNode = new Node(data);

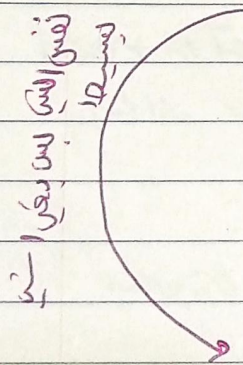
    if (ptr == null) {
        ptr = newNode;
    }
    else {
        while (ptr.next != null) {
            ptr = ptr.next;
        }
        ptr.next = newNode;
    }
}

```

```

Public void deleteFirst() {
    Node ptr = head;
    if (head == null) {
        System.out.println("Empty!");
    }
    else if (head.next == null) {
        head = null;
    }
    else {
        ptr = head.next;
        head.next = ptr;
    }
}

```



delete last

```

else if (head.next.next == null) {
    head.next = null;
}
else {
    while (ptr.next.next != null) {
        ptr = ptr.next;
    }
    ptr.next = null;
}

```



```
public void deletePosition (int key) {
```

```
Node ptr = head; next;
```

```
Node prev = head;
```

```
if (head == null) {  
    System.out.println("Linked List Empty!");  
}
```

```
else {
```

```
while (ptr != null && ptr.getData() != key) {
```

```
    prev = ptr; ptr = ptr.next;
```

```
ptr = head; ptr = ptr.next;
```

```
}
```

```
if (ptr == null) {
```

```
    System.out.print("Not Found");
```

```
}
```

```
else {
```

```
    prev.next = ptr.next;
```

```
    ptr.next = null;
```

```
}
```

```
}
```

```
}
```


double Linked List

chapter 4

```
class Node {  
    private key;  
    Node next;  
    عدادة في الجداول Node prev;
```

```
Node () {  
}
```

```
Node (int key) {  
    this.key = key;  
}
```

getter, setter
to string

```
class DLinkedList {
```

```
    DLinkedList () {  
        tailer  
        Node header, prev;  
        ? header =
```

insert At First. →

```
public void insertAtFirst (int key) {  
    Node newnode = new Node ();  
    { if (header == null) {  
        إدخال في الجداول header = tailer = newnode ;  
    }
```

```
    else {
```

```
        newnode.next = header ;
```

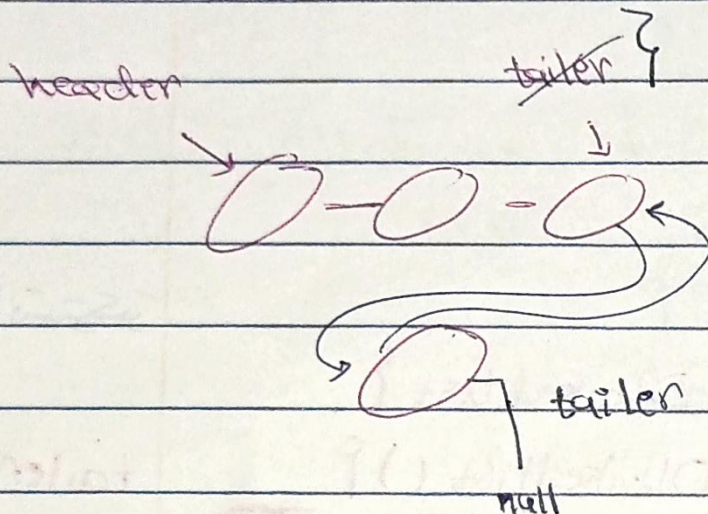
```
        header.prev = newnode ;
```

```
        header = newnode ;
```

```
    }  
}
```


→ insert At Last DI

```
public void insertAtLastDI (int key) {  
    Node newNode = new Node (key);  
    tailer.next = newNode; if (tailer == null) {  
        header.prev ← tailer = header = newNode;  
    }  
    else {  
        tailer.next = newNode;  
        newNode.prev = tailer;  
        tailer = newNode;  
    }  
}
```



? ← Serqular jalsi

Delete from position:-

```
public void deleteElement (int key) {
```

```
* Node Ptr = header; Node current;
```

```
if (header == null) {
```

```
    System.out.print ("out of flow");
```

```
    return;
```

```
}
```

```
else if (header.getKey() == key && header.next == null)
```

```
    header = null;
```

```
}
```

```
else {
```

```
    Prev = Ptr;
```

```
    Ptr = Ptr.next;
```

```
while (Ptr != null && Ptr.getKey() != key)
```

```
current = Ptr;
```

```
if (Ptr.getKey() == key)
```

```
    Ptr = Ptr.next;
```

```
if (Ptr.getKey() == key) {
```

```
    Prev.next = Ptr.next;
```

```
    Ptr.next.prev = Prev;
```

```
    Ptr.next = null;
```

```
    Ptr.prev = null;
```

```
}
```

Prev

current

next Ptr

current = header

Prev = current

next ptr = current.next;

Prev.next = next Ptr;

next Ptr.prev = previous;

STUDENTS-HUB.com

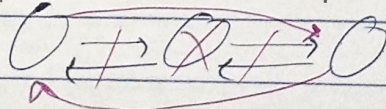
current.next = null;

Prev = null

Prev

curr

Uploaded By: anonymous



~~Delete at first~~

Delete from first :-

```
public void deleteFromFirst () {  
    Node ptr ;  
    if (header == null) {  
        // print("empty");  
    }  
    else if (header.next == null) {  
        header = null;  
    }  
    else {  
        ptr = header.next;  
        header.next = null;  
        ptr.prev = null;  
        header = ptr;  
    }  
}
```

System.gc(); | memory dump is in ~~...~~

Delete from last :-

```
public void deleteFromLast () {  
    if (tailer == null) {  
        // print("empty");  
    }  
    else if (tailer.prev == null) {  
        tailer = null;  
    }  
    else {  
        ptr = tailer.prev;  
        tailer.prev = null;  
        ptr.next = null;  
        tailer = ptr;  
    }  
}
```


insert At position :-

```
public void insertAtPosition (int data , int element) {
```

```
    Node current = header;    data.
```

```
    Node newNode = new Node ();
```

```
    if (header == null) {
```

```
        print ("data not found");
```

```
        header = tailer = newNode;
```

```
    }
```

```
    else { while (curr.next != null && curr.getData() != element) {
```

```
        curr = curr.next;
```

```
        else if (curr.getData() == element && curr.next != null) {
```

```
            newNode.next = curr.next;
```

```
            newNode.prev = curr;
```

```
            curr.next.prev = newNode;
```

```
            curr.next = newNode;
```

```
        }
```

```
    else {
```

```
        curr.next = newNode;
```

```
        newNode.prev = curr;
```

```
    }
```

```
}
```

```
}
```


Radix Sort (int [] a) {

int max = getMax(a);

for (int i = max; i > 0; max /= 10) {

countSort(a, P); ^{→ phase.}

element

P = P * 10;

}

$O(d * (n + k))$

number of digit

Range

دالة $O(d * (n + k))$ ←

Normalization ←

public int getMax(int arr[]) {

int max = arr[0];

for (int i = 1; i < arr.length(); i++)

if (arr[i] > max)

max = arr[i];

return max;

}

* Counting Sort :- No comparison, Sort according to keys counting the number having distinct key values.

Ex

0	2	1	1	0	2	5	4	0	2	8	7	7	9	2	0	1	9	16
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----

Range $\rightarrow 0 \leq a[i] \leq 9^k$

Size = $k+1$

Count

0	1	2	3	4	5	6	7	8	9
3	3	4	0	1	1	0	2	1	2

Update/Count

3	4	10	10	11	12	12	14	15	17
---	---	----	----	----	----	----	----	----	----

 phase

Count Sort (a, int/p)

count { for (int i = 0; i < n; i++) { ++count[a[i] / p % 10]; } } $O(n)$

update { for (int i = 1; i < k; i++) { count[i] = count[i] + count[i-1]; } } $O(k)$

المسألة في
Rang 1
تيرة كس

b { for (int i = n-1; i >= 0; i--) { b[i - count[a[i]]] = a[i]; } } $O(n)$

b

0	0	0	1	1	1	2	2	2	2	4	5	7	7	8	9	9
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

for (int i = 0; i < n; i++) { a[i] = b[i]; }

المراد من
Radix 11

Uploaded By: anonymous

$O(n+k)$

Stack

 $O(1)$

insert

[illegible] $O(1)$

delete.

← POP (C)

Stack overflow \Rightarrow file stack is full

stack underflow \Rightarrow top is stack is 0

(1-) size.

Top ← $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$

دقی عنری ~~دقی~~ ہار حبیب اللہ علی الواحہ . () Peel

الهدف والا هاته من النهاية.

FILAO &

→ Spure - ist nur Linkech ist nicht *

Application of stack is

داعیہیں سوال علیہں .

Main uses state are:

1- Evaluation of Arithmetic expression

• $E = A - ((B + C) * D)$

2- Execute recursive function.

→ depend on priority:

1 () ١٠٠٠ ١٠٠٠ ١٠٠٠

2 ^ القوة

3 * / 7000000

21 + - 21 9 21

الآن → الوسيلة

[illegible]

infix \rightarrow Prefix

$291 \rightarrow 2581$

$$\langle x \rangle \langle opp \rangle \langle y \rangle \rightarrow \langle opp \rangle \langle x \rangle \langle y \rangle$$

$\overline{G} \cong \overline{G} \times \overline{G}$

* التوكيف في ال infix \rightarrow postfix

$$-++A/-B*CDFEFGH$$

Pre \rightarrow in

فاندرپس لاله انوار
بالاول فاند

HG/FEDC-B-1A 4+3-

← Reverse

$$\langle y \rangle \langle \rho \rangle \langle x \rangle$$

Stacks Expression.

C HG

C^T HGF x y

$C \mid HGFEDC$

(*) H G ~~F~~ E D I C (B)

(-1)^T HGFE ~~DE~~ B* C/D A

(- HG FE A/B* C/D

$C + {}^T HGF \quad A/B * C/D - E$

$$C^{+} \quad H G^{-} \quad A B * C D - E + F$$

C- H $AB * C/D - E + F + G$

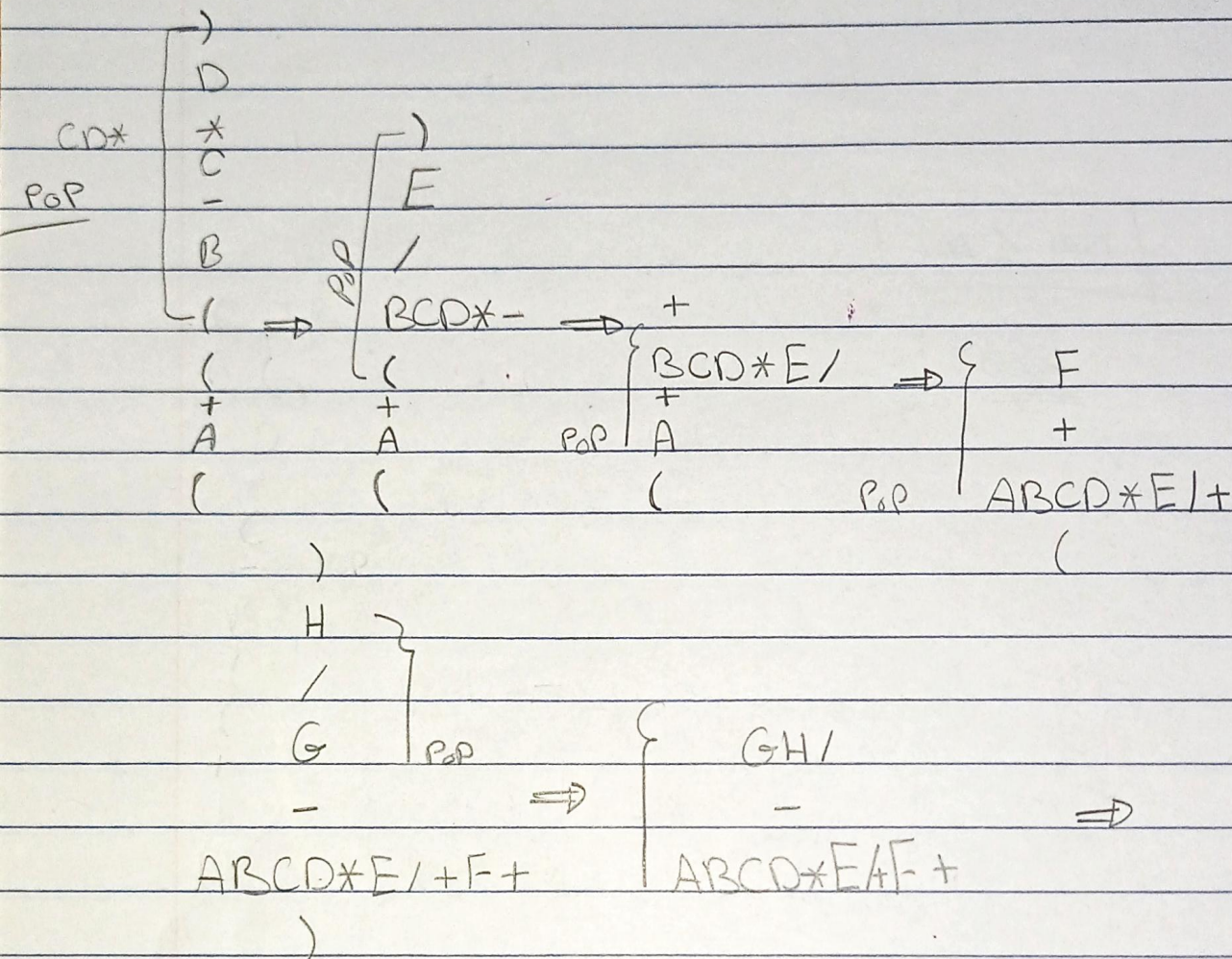
STUDENTS-HUB.com

$A/B * C/D - E + F + G - H$ Prefix

Examples

II $A + ((B - C * D) / E) + F - G / H$ In \rightarrow Post

$$\langle x \rangle \langle op \rangle \langle y \rangle \rightarrow \langle x \rangle \langle y \rangle \langle op \rangle$$

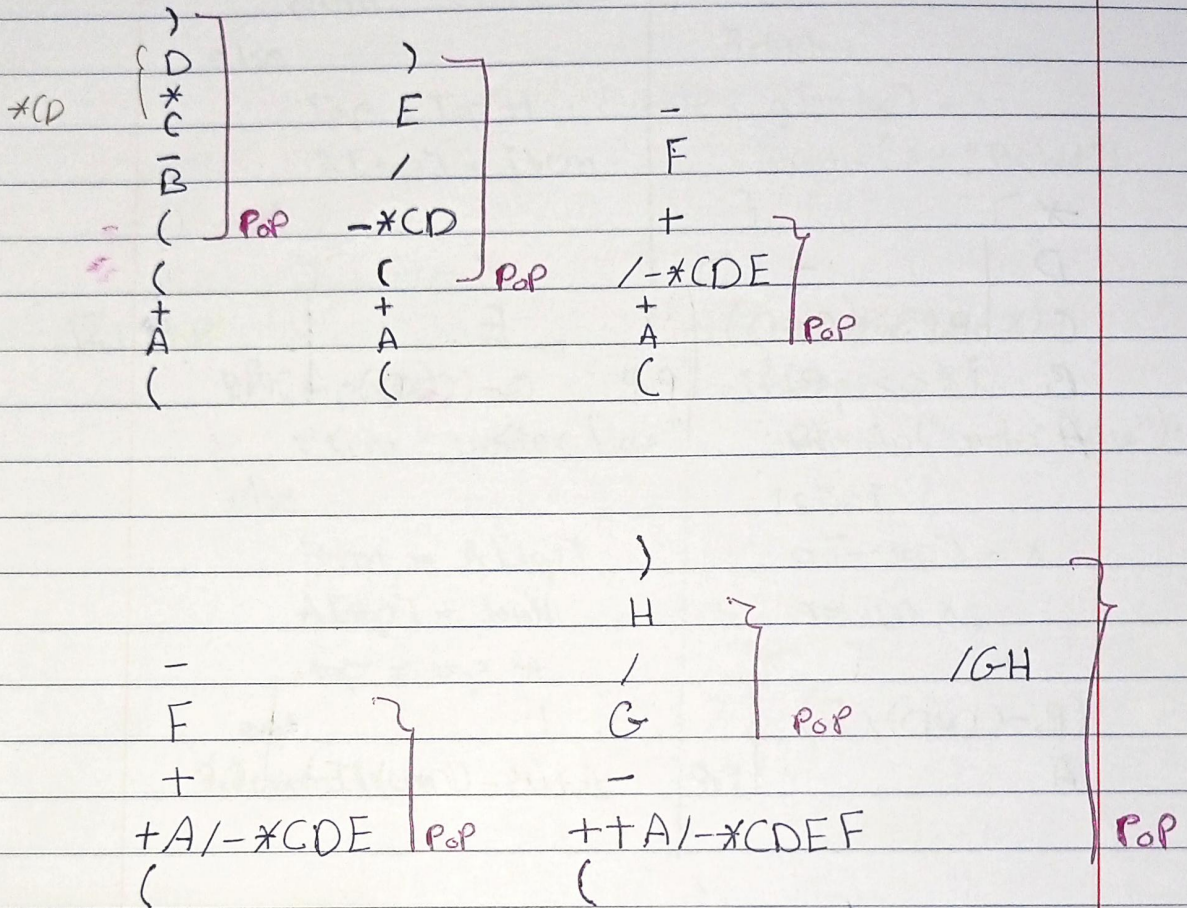


$$ABCD * E / + F + G H / -$$

Postfix

[2] $A + ((B - C * D) / E) + F - G / H$ in \rightarrow Pre

$\langle x \rangle \langle \text{opp} \rangle \langle y \rangle \rightarrow \langle \text{opp} \rangle \langle x \rangle \langle y \rangle$



$- ++A / - * C D E F / G H$ Prefix

[3] ~~ABCD*-E/+E+G/H/-~~ Post \rightarrow in

$\langle x \rangle \langle y \rangle \langle \text{opp} \rangle \rightarrow \langle x \rangle \langle \text{opp} \rangle \langle y \rangle$

*	}						
D			-	}	*	}	
C	Pop	(C*D)			E		
B		B	Pop		B-(C*D)	Pop	
A		A			A		

	+	}			+	}	
	(B-(C*D)/E)				F		
A		Pop			A+(B-(C*D)/E)	Pop	

	/	}			-	}	
H					G/H		
G		Pop					
A+(B-(C*D)/E)+F					Pop		

Stack using Array :-

[1] Push.

```
if (top > size)
    print "overflow"
else
    Top = Top + 1
    A[top] = item
end
```

```
{
    Push(int x) {
        if (top > (siz - 1))
            Println("overflow");
        else {
            a[++top] = x;
            Println(x + "Pushed");
        }
    }
}
```

[2] Pop

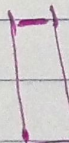
```
if (top < 0)
    print "under flow"
else
    item = A[top]
    A[top] = Null
    top = top - 1
end
return (item)
```

```
int Pop(int x) {
    if (top < 0)
        Println("under flow");
    else {
        a[--top] = x
        return x;
    }
}
```

```
int Peek() {
    if (top < 0)
        Print("under flow");
    else {
        int x = a[top];
        return x;
    }
}
```

هذا القيس
بـ (أ) حرفه

Stack (3)
الي دي الواد



Stack using linked List :-

```
1] boolean isEmpty() {  
    if (header == null) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

*Q15/13, up 20/11 *
Q16/16 Stack 11*

```
2] void push(int data) {  
    Node temp = header;  
    Node newNode = new Node(data);  
    if (header == null) {  
        header = newNode;  
    } else {  
        header = newNode;  
        newNode.next = temp;  
    }  
}
```

*insert First (ques) up **

```
3] void pop() {  
    if (header == null) {  
        Print("Stack is empty");  
    } else {  
        header = header.next;  
    }  
}
```

*delete last (ques) up **


```

4 | int peek () {
    if (header == null) {
        println("Stack empty!");
    }
    else {
        return header.data;
    }
}

```

من القائمة الى اليمين
الاول

Queue

مبدأ ال Queue ← زى الطابور
 الـ بـيـجـيـه اـوـلـ بـيـطـلـح اـوـلـ
 الـ بـيـجـيـه اـخـر اـخـر بـيـطـلـح اـخـر اـخـر
 FIFO
 LIFO

* الـ بـيـطـلـح اـوـل = dequeue = List.deleteLast()
 * الـ بـيـطـلـح اـخـر = enqueue = List.insertFirst()

* فـيـحـتـاج 2 pointer
 front = بـيـطـلـح اـوـل
 rear = بـيـطـلـح اـخـر

Stack \Rightarrow solve problem works on recursion.

Queue \Rightarrow solve problem have sequential processing.

Queue using Linked List :-

[1] void enqueue (int data) {

Node newnode = new Node (data);

if (rear == null) {

Front = rear = newnode;

rear.next = Front;

}

else {

rear.next = newnode;

rear = newnode;

rear.next = Front;

Front = newnode;

front و rear على الـ Queue

ليساو الـ newnode الى next الـ rear و الـ front

[2] int dequeue () {

Node temp;

if (front == null) { rear = null;

Print (" Empty!");

return 0;

}

else {

Node temp = front; temp = front;

front = front.next;

rear.next = front;

temp.next = null;

}

return temp;

}

Running time

Linked List \rightarrow

- insert $O(1)$
- delete $O(1)$
- Find $O(N)$

Stack \rightarrow

- Push $O(1)$
- Pop $O(1)$

Queue \rightarrow

- enqueue $O(1)$
- dequeue $O(1)$

Stack

based in the **LIFO** or **FILO**
first in last out or
last in first out.

insertion and deleting in stacks
take place only from one end
of the list called the **top**

insertion called **Push**
Delete called **Pop**

we need **one pointer** which
always point to the last element
called **top**

Solving problems works on
recursion

Queue

based on the **FIFO** or
LIFO, first in first out or
last in last out.

insertion ~~take~~ takes place at
the **rear** of the list and
deletion from **front**.

insertion called **enqueue**
delete called **dequeue**

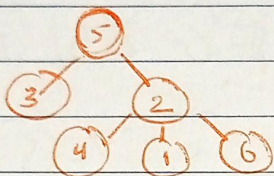
we need ~~to~~ **two pointer**
front → first element inserted
in the list and is still present,
rear → always point to the
last inserted element.

solve problems having
sequential processing.

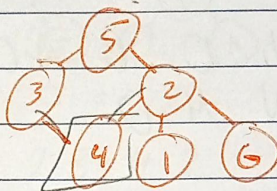
1. root هو عبارة عن Node ليس فيه ابناء

۳۔ کہ Node کہاں ہے، یعنی آؤ اُنکو من اُنکولاؤ۔

5. \rightarrow إذا وجد لك Mode \neq من بين القيم المتكررة، والى

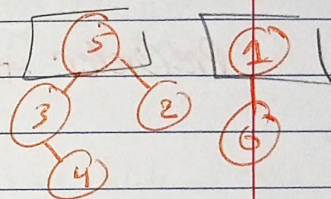


tree.



Not free

خِيَال (4) حَوَقِی



Not tree

$$2\sqrt{3}$$

00 - 2000 - 1000

Ancestor

Proper ancestor

Descendent

Proper descendent

Subtree

leaf

internal

Siblings

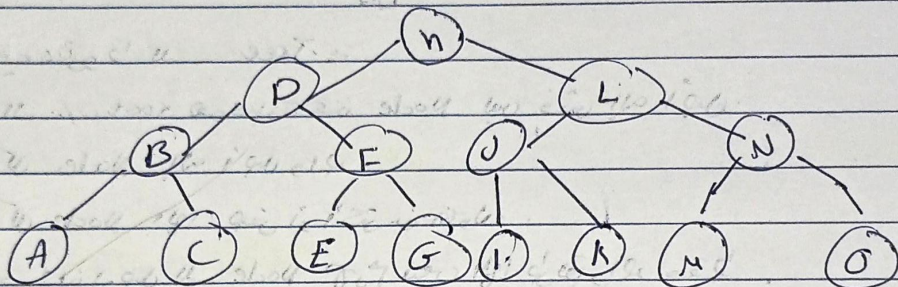
size

level

Light

Tree traversal

- ① Pre Order \Rightarrow root \rightarrow left \rightarrow right
- ② In Order \Rightarrow left \rightarrow root \rightarrow right
- ③ Post Order \Rightarrow left \rightarrow right \rightarrow root.



Pre Order \Rightarrow h D B A C F E G L J I K
N M O

InOrder \Rightarrow A B C D E F G h I J K L
M N O

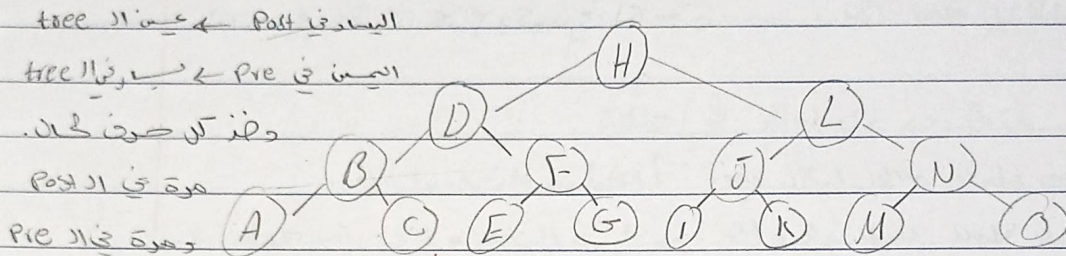
Post Order \Rightarrow A C B E G F D I K J M O N
L h

Exercise

important

Construct the binary tree whose traversal are given below.

Pre $\overset{\text{root}}{\text{H}} \text{ D B A C F E G I K J M O}$ root left right
Post $\text{A C B E F D I K J M O X K H}$ left right root.



Pre $\text{①} - + 9 9 + * 2 3 4 2$
infix $9 + 9 - 2 * 3 + 4 / 2$
Postfix $\Rightarrow ?$

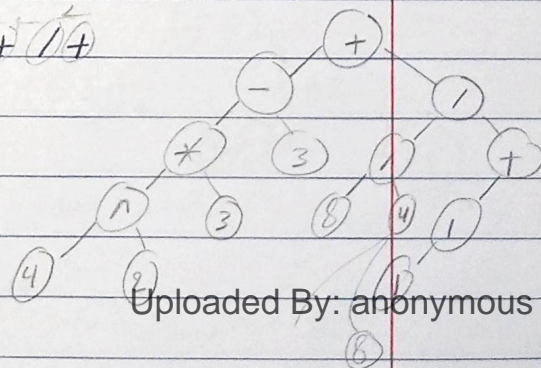
root \rightarrow left \rightarrow right

left \rightarrow root \rightarrow right

Pre $\Rightarrow \text{①} + \text{②} * \text{③} \wedge \text{④} 2 3 3 \text{⑤} \text{⑥} 8 4 \text{⑦} + \text{⑧}$

infix $\Rightarrow ?$

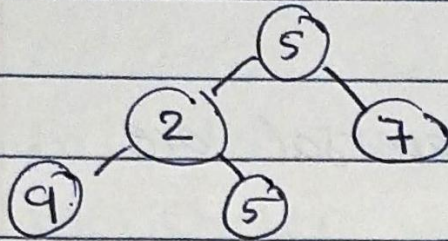
Post $\Rightarrow 4 2 \wedge 3 * 3 \text{⑤} 8 4 1 \text{⑥} + \text{⑦}$



Binary tree

Node 0 دے دیا ہے Tree

Node 2 دے دیا ہے



Full Binary tree \Rightarrow

Complete Binary tree \Rightarrow ہر گونہ مکمل ہونا چاہیے *
وہاں کن فی رتہ ہے ہر گونہ مکمل ہونا چاہیے۔

Binary search tree

$$O(\log N)$$

* كارتم الي على اليمين يكون
أكبر من ال root ، الي
على اليسار أصغر .

* عندنا شجرة Node هدية بتسوف
إذا أكبر من ال root دوح على اليمين
وإذا كان أكبر من ال Node الي على
يمين ال root د دوح على اليمين وهكذا
حتى لاقي مكان خاص .

* بالنية لكارت بوي اوله 3 حالة .

1. ال Node مالها اولاد يعني leaf ← كارتها وعدادي .
2. ال Node عندها ولا واحد ← جذع ال Node ويلي انشا اولاد عليها .
3. ال Node عندها ولدتين ← كارتم اوله ال Min أو ال Max
لا Node الي بوي ال Node فيها واولادها
مكانه .

Min ← أصغر شيء في ال left (أصغر)

Max ← أكبر شيء في ال right (أكبر)

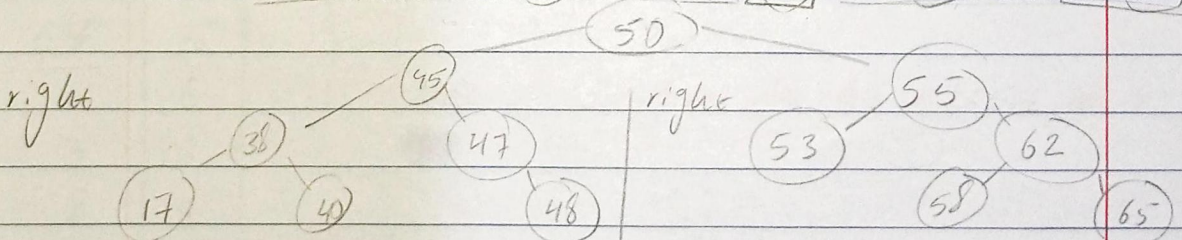
Min ⇒ ال right (أصغر شيء في ال right)

Max ⇒ ال left (أكبر شيء في ال left)

* يكون في ال left

* يكون في ال right

Ex Post ⇒ 17 40 38 48 47 45 53 58 65 62 55 50
in ⇒ 17 38 40 45 47 48 50 53 55 58 62 65



AVL tree

Binary search tree

height balanced \leftarrow 'Red' (Pisces)

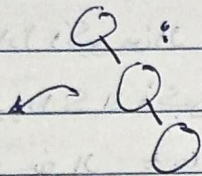
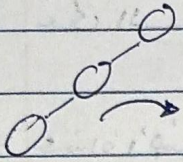
 $(1, 0, 1)$

balance ← tree $\mathcal{O}(\log n)$

Rotation dz

Right Rotation

Left Rotation



Single Rotation \rightarrow Right
 \searrow Left

Double Rotation \rightarrow Left Right
 \searrow Right Left

* delete or insert data is balanced

$\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$

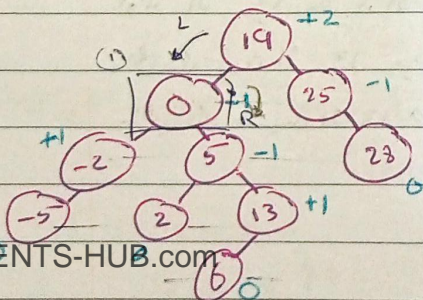
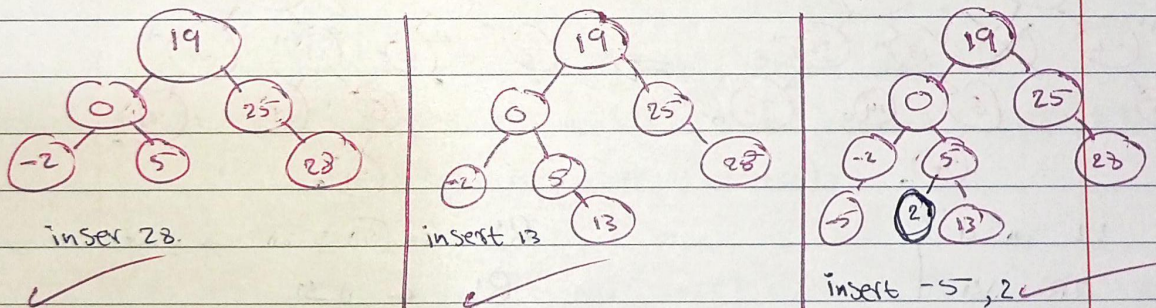
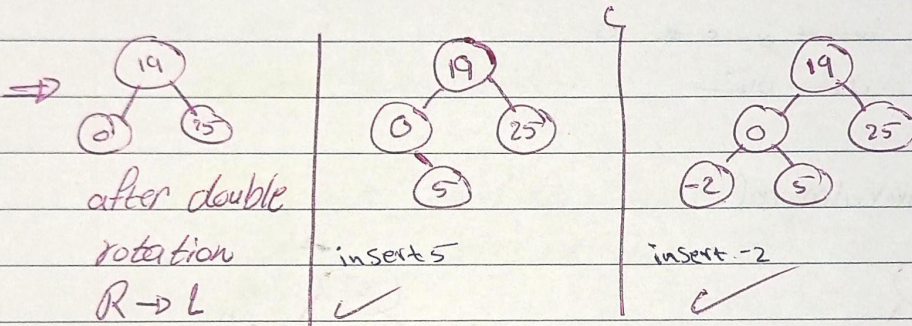
Insert an integer number consecutively in an empty AVL tree.

- 1) 0, 25, 19, 5, -2, 28, 13, 2, 6, 14, 7
- 2) Delete node 19 using next successors.

insert 0	insert 25	insert 19
Balanced	Balanced	imbalanced
no need to balance	no need to balance	need to balanced
		Problem: RL

* الـ 19 يكون اذلة الـ 25، الـ 19
والـ 25 يكون اذلة الـ 0
لا

Solve: RL rotation



imbalanced

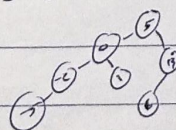
need to balanced

Problem \Rightarrow LR

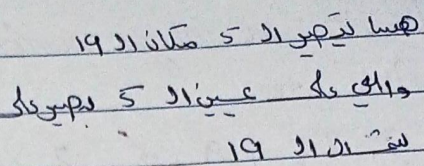
Solve \Rightarrow LR Rotation

Uploaded By: anonymous

* الـ 5 يكون مكان الـ 0، والـ 2 يكون مكان الـ 6



يعني الـ 5



insert 7 \rightarrow Balanced

* لیف آلودگی کے ماحول

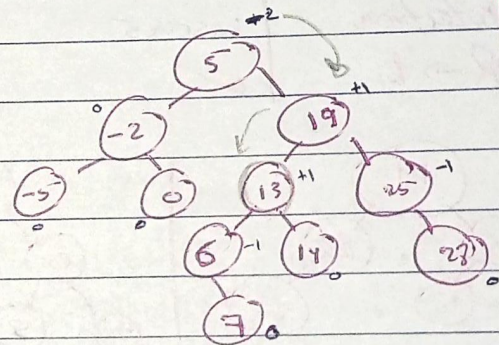
Dr \rightarrow + \leftarrow Balance Sheet \rightarrow Cr

Q. 2.4

[illegible]

۱۲۵

LL $\overline{aa} \overline{cc}$
RR JJ



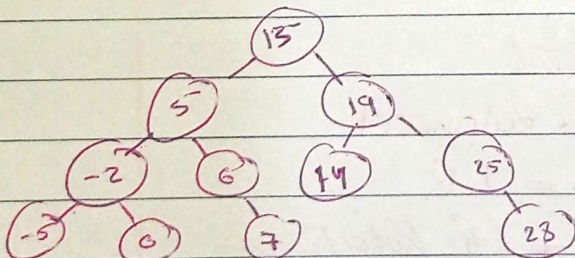
R2 ← (5) ثانية من م

$R_L \leftarrow \Delta S$

5 91 2 3 13 91 291

دینی کے لئے 13 دھرم کی عین الہی

دانشی های ۱۳ و ۱۴ و ۱۵ و ۱۶ و ۱۷ و ۱۸ و ۱۹



Tree

```
Node tree {  
    Object data;  
    Node left;  
    Node right;  
}
```

```
Node (Object data) {  
    this.data = data;  
}
```

```
class tree {  
    private Node root;
```

Main میں
root میں
private آپ کی

```
    public void insert (Object data) {  
        root = insert (root, data);  
    }
```

Public میں تو ہے

```
    private void insert (Node root, Object data) {
```

Main میں

```
        if (root == null) {  
            return new Node (data);  
        }
```

```
        else if (root.getData() < data) {
```

Right

```
            return root.setRight (insert (root.getRight(),  
                                            data));
```

```
        else if (root.getData() > data) {
```

Left

```
            return root.setLeft (insert (root.getLeft(),  
                                          data));
```

وہ recursion ہے *


```
Public void InOrder() {  
    InOrder (root);  
}
```

```
Private void InOrder (Node root) {  
    if (root == null)  
        return;  
    InOrder (root.getLeft());  
    System.out.println (root.get data());  
    InOrder (root.getRight());  
}
```

PreOrder

Post Order

```
public boolean delete (int data) {  
    if (root == null) {  
        return false;  
    }  
    else if (!  
        return  
    }  
    else {  
        return delete (root, data);  
    }  
}
```


4. أَهْمُ السِّمَاتِ الْوَلَدِيَّةِ -

* Create a B-tree of order 5

* لاپس اکر مینا size

~~\bar{w}_3 ACF (F) ABFG~~

CP LL NP

split zu Konjunktion

وَمِنْ أَمَلِ حَوْفِ ←

وَأَكْرَمَ عَلَى الْبَيْتِ أَهْلَهُ

کای الی

الحمد لله

4

10 20 40 50 60 70 80 30 35 5 15 60

10 | → 10 | 20 | → 10 | 20 | 40 | →

→ 40 | → 40 | → 40 |
 10 | 20 | 50 | 10 | 20 | 50 | 60 | 10 | 20 | 50 | 60 | 70 |

→ 40 | 70 | → 40 | 70 |
 10 | 20 | 50 | 60 | 80 | 10 | 20 | 30 | 50 | 60 | 80 |

30 | 40 | 70 |
 10 | 20 | 35 | 50 | 60 | 80 |

↓

30 | 40 | 70 |
 5 | 10 | 20 | 35 | 50 | 60 | 80 |

↓

40 | 70 |
 15 | 30 | 5 | 10 | 20 | 35 | 50 | 60 | 80 | 100 |

Hashing

Fixed size \leftarrow array

- Linear search $\Rightarrow O(N)$
- Binary search $\Rightarrow O(\log N)$
- if we know index $\Rightarrow O(1)$

"this array" → 0 to 100 index! 50 ap ← 1 500

١٥ ربيع الثاني ١٤٠٥ هـ وليلي 'السيف' الرقم ١٥ ربيع الثاني
مسألة.

→ Address of first, hashing all pages from the table
[X % table size]

insert 13 $\Rightarrow 13 \% 10 = 3$!! alle

insert 13 $\Rightarrow 13 \% 10 = 3$

فَسُوْهُ لَ بَرُوْحَكَ اِلَى اَعْوَدِهِ اَلَا فِيْ مَكَانٍ قَابِلٍ

collision $1991-2015$

0	10
1	
2	
3	3
4	4
5	5
6	6
7	13
8	8

* out of array size: تجاوز سعة المصفوفة

collision situation → solve

close

2 solve

3

1 - Separate chaining : array of Linked list

2 - open addressing :

1 - Linear probing → Linear search

2 - Quadratic probing → nonlinear search

3 - Double hashing → use two hash function.

→ 1 step

Separate chaining → Close.

Ex

23, 13, 21, 14, 7, 8, 15

table size 7

$h(key) = key \% \text{table size}$

$h(23) = 23 \% 7 = 2$

$h(13) = 13 \% 7 = 6$

$h(21) = 21 \% 7 = 0$

$14 \% 7 = 0$

$7 \% 7 = 0$

$8 \% 7 = 1$

$15 \% 7 = 1$

7 ← 14 ← 21 ← 0

15 ← 8 ← 1

23 ← 2

3

4

5

13 ← 6

* time Complexity $> O(1)$
but less than $O(N)$.

نقطه

* Linked List : ماتریس نیست و به سبب این

، memory و Pointer در آن استفاده می شود

insert 11 and 2 \leftarrow 3 in the Hashing 11 *
searching

decreption. și increption, DVA și d-k și r, *

Example 4.

Use the hash function to load the following commodity items into a hash table of size 13 using separate chaining.

1. Onion $\rightarrow O = 111$ $n = 110$ $i = 105$

$$\text{hash}(\text{onion}) = (111 + 110 + 105 + 111 + 110) \% 13$$

$$= 1$$

2- Salt $\Rightarrow S = 115$ $a = 97$ $l = 108$ $t = 116$

$$\text{hrs (salt)} = \frac{(115 + 97 + 108 + 116)}{7} \% 13$$

$0 \rightarrow 0Kra \rightarrow Potgto$

1 \rightarrow Onion \rightarrow Carrot

2 2019 10-26-2019

3 $2.9, 2.0, 9.0, 2.0$

4 → cabbage

5

6 → mushroom

7 \rightarrow salt

8

9 → Cucumber

b → tomato → melon → olive

11 \rightarrow banana

12 \rightarrow orange.

open addressing :-

$$* h_i(\text{key}) = [h(\text{key}) + f(i)] \% n \rightarrow \text{كل الاصل}$$

↳ linear

Quadratic

double

$$* f(0) = 0 \quad \text{حالة الـ 0 في الـ 0} \\ f(0) \% n, f(1) \% n, \dots, f(n-1) \% n \rightarrow \text{table size}$$

value \Rightarrow between **0 and $n-1$**

$$* \text{linear } f(i) = i$$

$$* \text{Quadratic } f(i) = i^2$$

$$* \text{double } f(i) = i * h_p(\text{key})$$

بالطريقة بفاتر الـ table size يكون Prime (عدد أولي)

~~في حال وجود collision~~

load factor \Rightarrow collision \Rightarrow load factor

وكان يكون بين (0.6, 0.7)

λ = Number of element / table size.

undesirable under

← وإذا كان أكبر من 0.7
rehash

Type of open addressing :-

I Linear

Example

insert (18), (26) (35), (9), find (15) (-48)

delete (35), (40) Final (9), insert (64), (47)

Lind (35).

$f(i) = i$, table size = 13.

$$h(\text{key}) = \text{key} \% \text{table size}$$

$$h_i(\text{key}) = (h(\text{key} + f(i))) \% 13$$

$$h_0(18) = (18 \% 13 + 0) \% 13 = 5 \% 13 = 5$$

$$\text{hd}(26) = (26 \% 13 + 0) \% 13 = 13 \% 13 = 0$$

$$10(35) = ((35 \% 13) + 0) \% 13 = 9 \% 13 = 9$$

$$h_0(9) = (9 \% 13 + 0) \% 13 = 9 \% 13 = 9 \rightarrow \text{collision}$$

$$\hookrightarrow h_1(9) = (9 \cdot 13 + 1) \cdot 13 = 10 \cdot 13 = 10$$

$$ho(15) = (15 \% 13 + 0) \% 13 = 2 \% 13 = 2 \text{ (not found)}$$

$$ho(48) = ((48 \% 13) + 0) \% 13 = 9 \% 13 = 9$$

(لَقَدْ نَزَّلَ رَقْمًا بَيْنَ يَدَيْهِ ۖ فَفُتِحَ الْبُورُ عَلَى الْوَلِيِّ بِحُورِ

$$h_1(48) = (48 \cdot 13 + 1) \% 13 = 10 \% 13 = \underline{10}$$

ما لقي 48 و در اهل زیر فوق

$$h_2(48) = (48 \div 13) + 2 \div 13 = 11 \text{ (Not found)}$$

$ho(35) = 9$ index of 9 has a value 35

D ← E indicating change status

$$h_0(40) = (40 \% 13 + 0) \% 13 = 1 \% 13 = 1 \quad \text{(Not found)}$$

Delete dari status E jika

$$hd(9) = (9\%13 + 0) \% 13 = 9\%13 = \underline{9} \quad \text{برج (د)}$$

$$h_1(9) = 9 \cdot \frac{1}{13} + 1 \cdot \frac{1}{13} = \underline{\underline{\frac{10}{13}}}$$

$D \leftarrow D$ in status $\downarrow \downarrow$

$$h_0(64) = (64 \cdot 0.13 + 0) \cdot 0.13 = 12$$

$$h_0(47) = (47\% \cdot 13 + 0) \% \cdot 13 = 8$$

TS-HUB.com (35) = 992 Found

یوح (د)

و یوحنا ۱۱

E Empty

D Delete

O qubay

[2] Quadratic

$$p(i) = i^2$$

$$h_i(\text{key}) = (h(\text{key}) + i^2) \% \text{table size}$$

insert(13), (39), (26), (52)

* كل نفس العنصر الى مكان واحد

$$h_0(13) = (13 \% 13 + 0^2) \% 13 = 0$$

$$h_0(39) = (39 \% 13 + 0^2) \% 13 = 0 \quad \text{collision}$$

$$h_1(39) = (39 \% 13 + 1^2) \% 13 = 1$$

$$h_0(26) = (26 \% 13 + 0^2) \% 13 = 0 \quad \text{collision}$$

$$h_1(26) = 1 \quad \text{collision}$$

$$h_2(26) = (26 \% 13 + 2^2) \% 13 = 4$$

* كلما زاد ال Collision بتزايد المشكل

الحجم اكبر من 70 % مشكلة فتسولج

نبدأ Array جديدة ونبدأ بها x2

وننقل العناصر الى مواقع جديدة.

Primary Clustering

دالة مكملة حولين

و من بسبب اثره ال

Collision.

عشان افهم انه ال table size هو Prime

المعادلة

$$P(x) = x^2 + x + 41$$

* في بالسلاسل الفرق

بين ال open و ال close

Secondary clustering

* کھانا میں سے بعض کھانا
انسان (بکون کے کھانا)

Primary cluster اور ان کے

31 Double Hashing:-

$$h_i(\text{key}) = [h(\text{key}) + f(i)] \% \text{table size}$$

* $f(i) = i * h_p(\text{key})$

is prime $\Rightarrow \begin{cases} h_p = 1 + \text{key} \% (\text{table size} - 1) \\ = q - (\text{key} \% q) \\ q \neq (\text{key} \% q) \end{cases}$

q is prime #

Example:-

insert (26, 70, 18, 9, 47, 35, 96, 64)

* table size 13

* $h_p = 1 + \text{key} \% 12$

Double

$$h_i(\text{key}) = (h(\text{key}) + i * h_p(\text{key})) \% \text{table size}$$

$$h_0(26) = (26 \% 13 + 0) \% 13 = 0$$

$$h_0(70) = (70 \% 13 + 0) \% 13 = 5$$

$$h_0(18) = ((18 \% 13) + 0) \% 13 = 5 \text{ collision}$$

$$h_1(18) = (18 \% 13 + (1 * (1 + 18 \% 12))) \% 13 =$$

$$(5 + (1 * 7)) \% 13 = 12$$

$$h_0(96) = (96 \% 13 + 0) \% 13 = 5 \text{ collision}$$

$$h_1(96) = (96 \% 13 + (1 * (1 + 96 \% 12))) \% 13 =$$

$$(5 + 1) \% 13 = 6$$

0 26

1

2

3

4

5 70

6 96

7

8

9

10

11

12 18

Exercise 2-

if the hash table after using linear probing is as shown below?

1- insert ~~25~~ 29.

2- is there a problem in hashing table? if there is state it and give a solution.

$$h_0(29) = (29 \% 7 + 0) \% 7 = 1 \% 7 = 1 \quad \text{collision}$$

$$h_1(29) = (29 \% 7 + 1) \% 7 = 2 \% 7 = 2 \quad \text{collision}$$

$$h_2(29) = (29 \% 7 + 2) \% 7 = 3 \% 7 = 3 \quad \text{collision}$$

$$h_3(29) = (29 \% 7 + 3) \% 7 = 4 \% 7 = 4 \quad \text{inserted.}$$

0	
1	15
2	37
3	25
4	29
5	
6	13
7	

2x array collision (تصادم) في الجدول (collision) في الجدول *
 في الجدول array = 14 في الجدول Prime # هو 17
 Double hash في الجدول insert الجدول

table size = 17

$$hp = 1 + key \% 16$$

$$h_0(15) = (15 \% 17 + 0) \% 17 = 15$$

$$h_0(37) = (37 \% 17 + 0) \% 17 = 3$$

$$h_0(25) = (25 \% 17 + 0) \% 17 = 8$$

$$h_0(29) = (29 \% 17 + 0) \% 17 = 12$$

$$h_0(13) = (13 \% 17 + 0) \% 17 = 13$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
			37					25				29	13		15	

$$* \text{hash} = \sum_{i=0}^{\text{keysize}-1} \text{key}[\text{keysize}-i-1] \cdot 32^i$$

String الـ 32^i *
Array

بجاء الـ ASCII code الـ

Method الـ الـ الـ الـ

Public Static int hash (String key, int table-size) {

int hashVal = 0;

for (int i=0; i < key.length(); i++)

hashVal = hashVal << 5 * key.charAt(i);
hashVal %= table-size;

32 الـ الـ
shifting الـ

if (hashVal < 0)

hashVal += table-size;

return hashVal;

}

⇒ الـ الـ الـ
over flow.

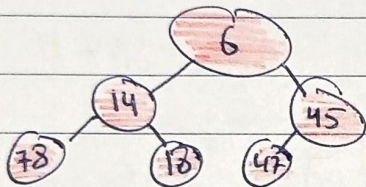
الـ الـ الـ
positive الـ

HEAP

Chapter 6

Binary Heap

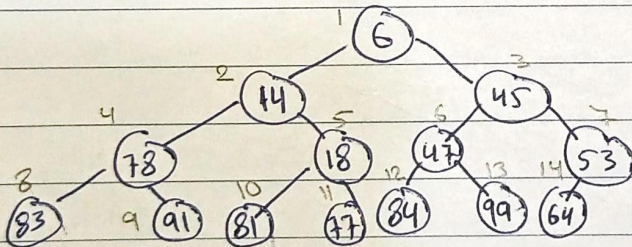
1. Complete binary Tree. → خاصية من فوق لادنا ومن اليسار لليمين
2. Min-heap ordered. → الاولاد كاد من يكونوا اكبر او يساوي الاب



موقعهم الذي في اليسار اكبر
او الذي في اليمين اكبر
من الاب

Minimum element is in the root

Heap with N elements has height = $\log_2 N$



Tree ال Tree
ليس بترتيب

$$\text{Parent}(i) = i/2$$

$$\text{left}(i) = 2i$$

$$\text{Right}(i) = 2i+1$$

Insert

insert into next available slot

Bubble up until it's heap ordered

وین (3) با (4) دبیته
ولی یه 5 > 4 و 5 > 3

Swap 4 & 3 ←

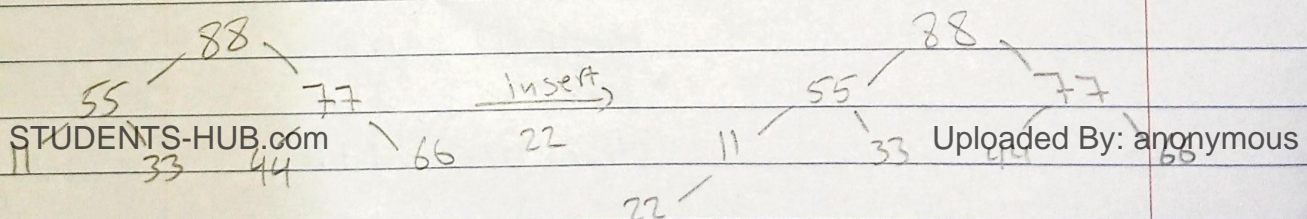
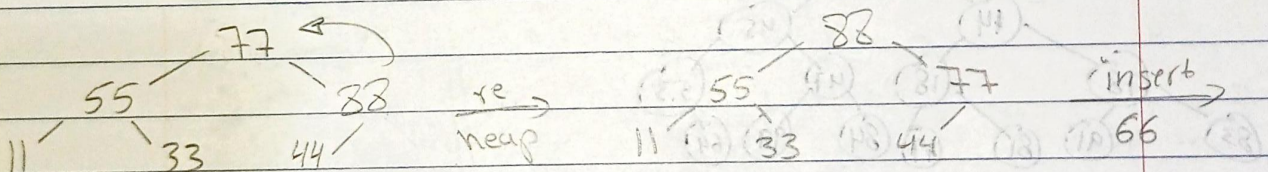
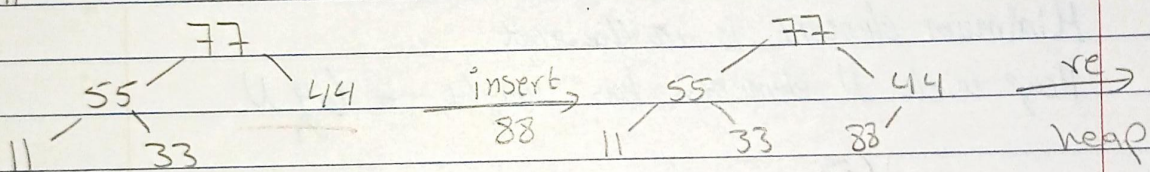
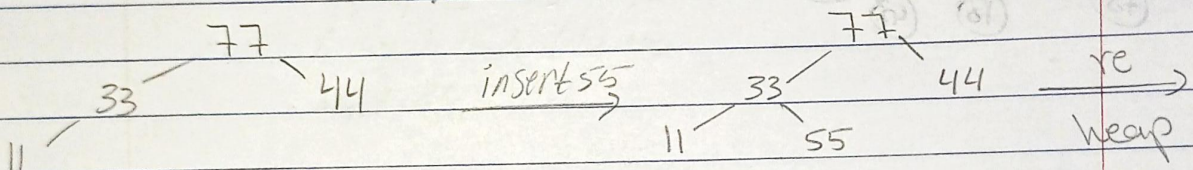
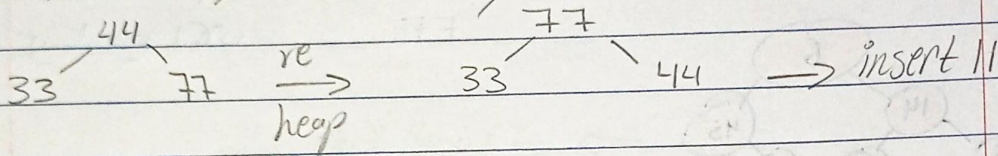
heap ordered
(5 > 4 > 3)

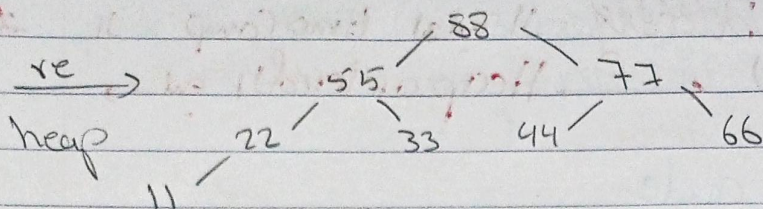
Time Complexity $O(\log N)$

Example

Build Maximum heap tree?

44, 33, 77, 11, 55, 88, 66, 22





Done

Delete minimum element from heap.
exchange root with rightmost leaf.

Bubble root down until it's heap ordered.

* آخر ابن اليمين

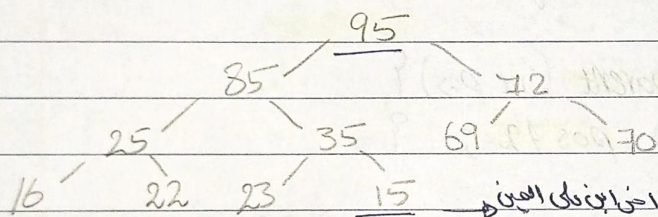
* يتحقق اذا الابن اليمين من

الابناء

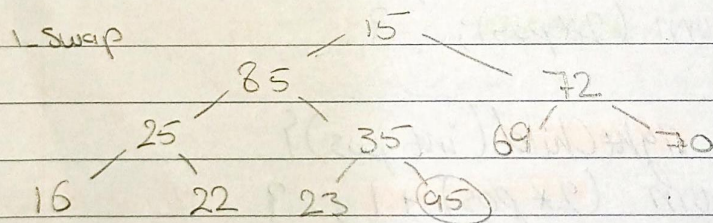
* وبقوة من حجم الارتفاع

Example

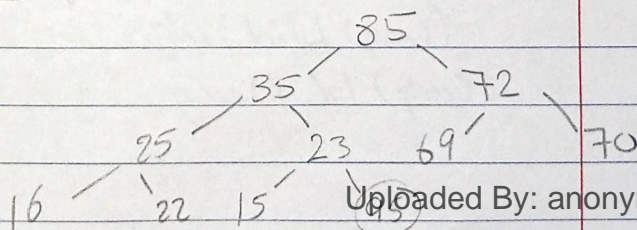
Delete root 95



maximum
Heap



2 - Heap ordered



$O(\log N)$ Deleting \rightarrow $O(n \log N)$ \leftarrow Heap ordered, \rightarrow $O(\log N)$ time Comp \rightarrow \times

Min Heap code

```
public class MinHeap {
```

```
    private int[] Heap;  $\rightarrow$  size of array
```

```
    private int size;  $\rightarrow$  current size of heap
```

```
    private int maxSize;  $\rightarrow$  max size of heap
```

```
    private static final int FRONT = 1;
```

Constructor \rightarrow `public MinHeap(int maxSize) {`

```
    this.maxSize = maxSize;
```

```
    this.size = 0;
```

```
    Heap = new int[this.maxSize + 1];
```

```
    Heap[0] = Integer.MIN_VALUE; }
```

```
    private int parent(int pos) {
```

```
        return pos / 2; }
```

```
    private int leftChild(int pos) {
```

```
        return (2 * pos); }
```

```
    private int rightChild(int pos) {
```

```
        return (2 * pos) + 1; }
```



```

public void insert (int element) {
    if (size >= ele maxsize) {
        return ; } → insert فنى 3 ك ال نزل
    Heap [++size] = element ;
    int current = size ; → مؤشر لا size

```

```

    while (Heap [current] < Heap [Parent (current)]) {
        → تبادل الاب مع الابن ← Swap (current, parent (current));
        current = parent (current); }
    }

```

```

public void minHeap () {
    for (int pos = (size/2); pos >= 1; pos--) {
        minHeapify (pos); }
}

```

```

public private void minHeapify (int pos) {
    → فنى اله اولاد ← if (!isLeaf (pos)) {
        or
        if (Heap [pos] > Heap [leftChild (pos)] ||
            Heap [pos] > Heap [rightChild (pos)]) {
            if (Heap [leftChild (pos)] < Heap [rightChild (pos)]) {

```

```

                swap (pos, leftChild (pos));
                → اغل ال على ال يمين ار ال يمين ← minHeapify (leftChild (pos));
            }

```

```

        } else {
            swap (pos, rightChild (pos));
            minHeapify (rightChild (pos)); }
    }
}

```



```

private boolean isLeaf (int pos) {
    if (pos >= (size / 2) && pos <= size) {
        return true;
    }
    return false;
}

```

```

private void swap (int fpos, int spos) {
    int temp;
    temp = Heap[fpos];
    Heap[fpos] = Heap[spos];
    Heap[spos] = temp;
}

```

```

public int remove () {
    int popped = Heap[FRONT];
    Heap[FRONT] = Heap[size - 1];
    minHeapify (FRONT);
    return popped;
}

```

بدل اول
 element
 آخر element

Heap sort

1. Build a max heap from the input data.
2. At this point, the largest item is sorted at the root of the heap. Replace it with the last items of the heap followed by reducing the size of heap by 1.
3. Repeat above steps while size of heap is greater than 1.

Time Complexity :

Heapify $\Rightarrow O(\log N)$

Build Heap $\Rightarrow O(N)$

Heap sort $\Rightarrow O(N \log N)$.

Example

A = [6, 5, 3, 1, 8, 7, 2, 4]

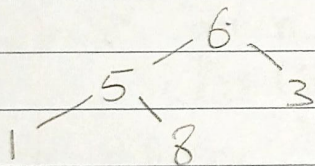
full Array.

size = 8

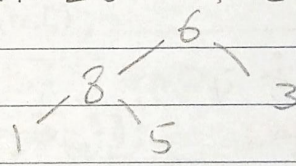
1. (Heapify) نزل 5 و 1

size 1, Array

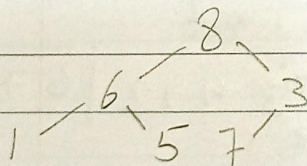
وبقى الـ 6 الى جوا الـ 8 و 7



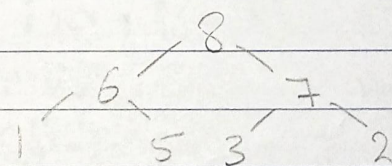
re
heap



re
heap

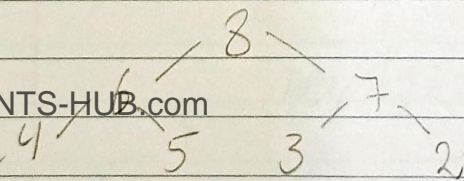


re
heap



re
heap

4

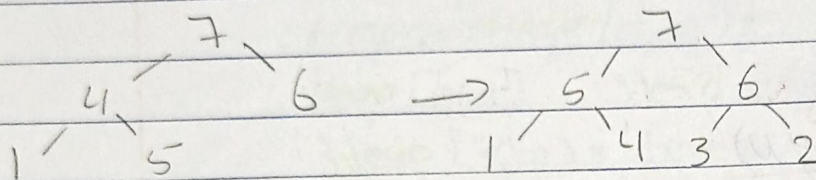
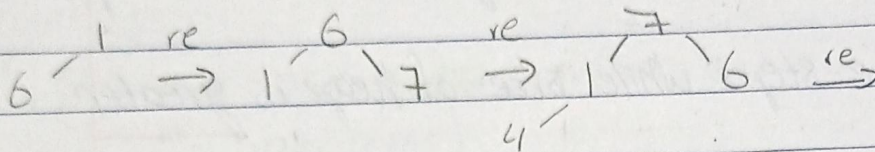


Front
 $A = [8, 6, 7, 4, 5, 3, 2, 1]$

8, remove 1st element

$A = [1, 6, 7, 4, 5, 3, 2]$ (8)
 size = 7

Build Heap

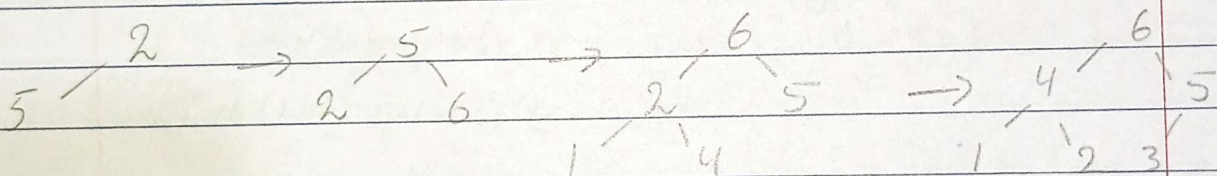


$A = [7, 5, 6, 1, 4, 3, 2, 8, 1]$

remove 7

$[2, 5, 6, 1, 4, 3, 7, 8, 1]$

size = 6

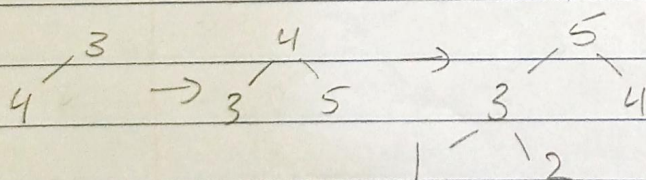


$A = 6, 4, 5, 1, 2, 3, 7, 8, 1$

$3, 4, 5, 1, 2, 6, 7, 8, 1$

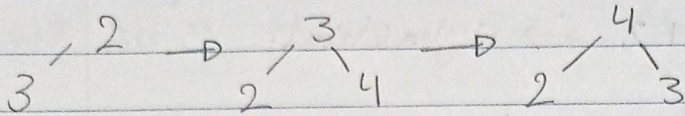
remove 6

size = 5

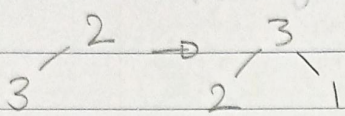


$A = 5, 3, 4, 1, 2, 6, 7, 8, 1$

A = 5 3 4 1 2 | 6 7 8 |
 2 3 4 | 5 6 7 8 | remove 5



A = 4 2 3 1 | 5 6 7 8 |
 2 3 1 | 4 5 6 7 8 | remove 4



A = 3 2 1 | 4 5 6 7 8 |
 2 1 | 3 4 5 6 7 8 |

remove 3

A = 2 1 | 3 4 5 6 7 8 |
 1 | 2 3 4 5 6 7 8 |

remove 1

A = 1 2 3 4 5 6 7 8

is sorted. ∞

```

public void sort (int arr []) {
    int n = arr.length();
    for (int i = n/2 - 1; i >= 0; i--)
        heapify (arr, n, i);
    }
  
```

→ Data

```

    for (int i = n-1; i > 0; i--) {
        int tmp = arr [0];
        {
            arr [0] = arr [i];
            arr [i] = tmp;
        }
    }
  
```

swap

heapify (arr, i, 0);

index up


```
public void heapify(int arr[], int n, int i) {
```

```
    int largest = i → root
```

```
    int l = 2*i + 1 → left child
```

```
    int r = 2*i + 2 → right child
```

```
    ← الذي أكبر if (l < n && arr[l] > arr[largest])  
        largest = l;
```

```
    ← الذي أكبر if (r < n && arr[r] > arr[largest])  
        largest = r;
```

```
    if (largest != i) {
```

```
        int swap = arr[i];
```

```
        arr[i] = arr[largest];
```

```
        arr[largest] = swap;
```

Swap ↑

```
        heapify(arr, n, largest); → Recursive
```

```
    }
```


Radix Sort

- هدفی شخیص فی ال Sorting سبب ال عام
1. No extra space ← اسے ما احتاج Array مرتبة
 2. Time Complexity

Radix Sort $\rightarrow O(N \log N)$ with space
 Selection and Bubble Sort $\rightarrow O(N^2)$ No space
 Heap Sort $\rightarrow O(N \log N)$ No space go

Quick Sort

- bad bad bad good
1. First element pivot
 2. last // //
 3. random // //
 4. median // //

حد النوع مبيع في ال Pivot

الحدود

الانما يتكون حد من حد معين

من ههول الاربعة ←

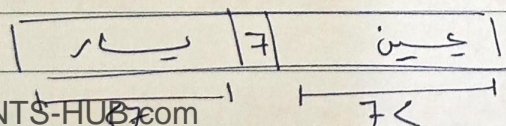
Example:

$A = [7 \ 6 \ 10 \ 5 \ 9 \ 2 \ 1 \ 15 \ 7]$

Pivot First element \Rightarrow Pivot = $A[0]$;

حبا عمل Put pivot in proper (place) so that all data less than Pivot goes on left, and all data great than pivot goes to Right.

بف ال Pivot مكان حية ديس
 ال الانما اة صيف حيه في اليسار
 وال اكبر في اليمين



divide & conquer

7 | 6 | 10 | 5 | 9 | 2 | 1 | 15 | 7 |

↑

↑

swap

↑↑

Pivot = 7

Start

s

end

* Move start to right ⇒ till find a number greater than or equal to the Pivot

* Move end to left ⇒ till find a number less than the Pivot.

7 | 6 | 7 | 5 | 9 | 2 | 1 | 15 | 10 |

↑
s

↑
s

↑
e

↑
e

swap

7 | 6 | 7 | 5 | 1 | 2 | 9 | 15 | 10 |

↑
s

↑
e

↑
e

↑
s

swap ٧ و ٢

No swap because

$s > e$ (cross)

Swap between element end with pivot.

Pivot 2 | 6 | 7 | 5 | 1 | 7 | 9 | 15 | 10 |

Pivot = 2

less

greater

proper place

هنا هو المكان الجيد

اليمين واليسار

Algorithm

Partion (a, l, h) {

int start, end, pivot;

pivot = a[l];

while (start < end) {

لما نلاقي رقم أكبر من ال
Pivot وقف

while (a[start] <= pivot) {

start ++; }

while (a[end] > pivot) {

إذا لقيت رقم أصغر وقف
end --; }

if (start < end) {

Swap(a[start], a[end]);
}

Swap(a[l], a[end]);

* اظهر تقاطع
بدل بين ال
مع ال index
كل ال الباقي

return end;

}

عنشان اعرف
مكان ال Pivot

Qsort (a, l, h) {

int loc = Partion (a, l, h);

Qsort (a, l, loc - 1);

الجزء الاكبر (اليمين)

Qsort (a, loc + 1, h);

الجزء الاصغر (اليسار)

}

* Best Case $\Rightarrow O(N \log N)$

Worst Case $\Rightarrow O(N^2)$

ول 81 Pivot = a[L]; ← Pivot ال 581
ول 81 Pivot = a[h];
Pivot = randome (a);

فكرتي
Pivot = Median3 (a, L, h);

int Median3 (int a[], int L, int h) {

int center = (L+h)/2;

if (a[L] > a[center])

swap(a, L, center);

if (a[L] > a[h])

swap(a, L, h);

if (a[center] > a[h])

swap(a, center, h);

Swap(a, center, h-1);

return a[h-1];

} Pivot ال 581

بدل الرقم الى اليمين
مع اليمين قبل الاخير

Insertion Sort

تستعمل سخل لعب السندة .

Exercise

A = 5 | 4 | 10 | 1 | 6 | 2

Sort Unsort

temp 4

index 1

يقع 4 في 4

وبقارن الى جوار ال sorted list.

هل الى جوار list sort اكبر من الى جوار temp .
إذا أكبر اجل shift to the right

وبقارن باقى العناصر الى قبله مع ال temp .
إذا لقيت أكبر من ال temp .

A = 4 | 5 | 10 | 1 | 6 | 2

Sort Unsort

temp 10

وبرجع بقارن 10 مع 5 و 4
من اكبر من 10 يتدخل في

A = 4 | 5 | 10 | 1 | 6 | 2

Sort Unsort

4 5 10 1 6 2

4 5 10 1 6 2

4 5 10 1 6 2

1 4 5 10 6 2

Sort Unsort

1 4 5 10 6 2

1 4 5 6 10 2

Sort Unsort

1 4 5 6 10 2

temp 1

shift ← 10 10

shift ← 5 10

shift ← 4 10

index 0

جاء 1 في 0

temp 6

shift ← 10 6

6 10 5 10

A = 1 4 5 6 10 | 2

Sort

unSort

temp | 2

1 4 5 6 ? 10

shift ← 2 < 10

1 4 5 ? 6 10

shift ← 2 < 6

1 4 ? 5 6 10

shift ← 2 < 5

1 ? 4 5 6 10

shift ← 2 < 4

1 2 4 5 6 10

lets 2 < 1

A = 1 2 4 5 6 10

sorted

Algorithm

n ← 1 in while 2 while 5

1 ← n in temp

decrement

for (i = 1; i < n; i++)

temp = A[i];

j = i - 1;

while (j >= 0 && A[j] > temp)

shifting A[j+1] = A[j];

j--;

A[j+1] = temp;

}

Time Complexity ⇒ worst $O(N^2)$

Best $O(N)$

Shell sort

حركة على مسافات

* تقليل عدد المقارنات التي أجعل فيها shifting

حجم array

$$A = \underline{23} \quad 29 \quad 15 \quad 19 \quad \underline{31} \quad 7 \quad 9 \quad 5 \quad 2 \quad \left| \text{gap} = \left\lceil \frac{n}{2} \right\rceil \right|$$

i No swap j

$$23 \quad 29 \quad 15 \quad 19 \quad 31 \quad 7 \quad 9 \quad 5 \quad 2$$

i swap j

$$23 \quad 7 \quad \underline{15} \quad 19 \quad 13 \quad 29 \quad \underline{9} \quad 5 \quad 2$$

i swap j

$$23 \quad 7 \quad 9 \quad 19 \quad 31 \quad 29 \quad 15 \quad 5 \quad 2$$

i swap j

$$23 \quad 7 \quad 9 \quad 5 \quad 31 \quad 29 \quad 15 \quad 19 \quad 2$$

i swap j

$$A = \underline{23} \quad 7 \quad 9 \quad 5 \quad 2 \quad 29 \quad 15 \quad 19 \quad 31$$

$$i - \text{gap} \geq 0$$

$$A = 2 \quad 7 \quad 9 \quad 5 \quad 23 \quad 29 \quad 15 \quad 19 \quad 31 \quad 4 - 4 = 0$$

phase 0 = جايزة

هذا ال phase الثانية مبدية

في ال gap القديم تقسم 2

$$\text{New gap} = \text{Old gap} / 2$$

Backward ::

index 0

هذا يقارن ال i

Backward.

$$* \text{New gap} = 4 / 2 = 2$$

$$A = \underline{2} \quad 7 \quad 9 \quad 5 \quad 23 \quad 29 \quad 15 \quad 19 \quad 31$$

i No swap j

$$2 \quad 7 \quad 9 \quad 5 \quad 23 \quad 29 \quad 15 \quad 19 \quad 31$$

i swap and No need to look Backward

$$\text{STUDENTS-HUB.com} \quad 5 \quad 9 \quad 7 \quad 23 \quad 29 \quad 15 \quad 19 \quad \text{Uploaded By: anonymous}$$

oo i No swap j

$$2 - 2 = 0$$

need to look Backward

~~2 5 9 7~~

2 5 9 7 23 29 15 19 31

look \leftarrow \overline{i} \overline{j} No need swap

2 5 9 7 23 29 15 19 31

look \leftarrow \overline{i} \overline{j} swap \overline{i} and \overline{j}

2 5 9 7 15 29 23 19 31

look \leftarrow \overline{i} \overline{j} swap \overline{i} and \overline{j}

2 5 9 7 15 19 23 29 31

look \leftarrow \overline{i} \overline{j} No swap

A = 2 5 9 7 15 19 23 29 31

phase 2 ✓

new gap = $2/2 = 1$

A = 2 5 9 7 15 19 23 29 31

\overline{i} \overline{j} No swap No look Back word

2 5 9 7 15 19 23 29 31

look \leftarrow \overline{i} \overline{j} No swap and need to look Back word

2 5 9 7 15 19 23 29 31

look \leftarrow \overline{i} \overline{j} swap

2 5 7 9 15 19 23 29 31

\leftarrow \overline{i} \overline{j} No swap

2 5 7 9 15 19 23 29 31

\leftarrow \overline{i} \overline{j} No swap

2 5 7 9 15 19 23 29 31

\leftarrow \overline{i} \overline{j} No swap

2 5 7 9 15 19 23 29 31

\leftarrow \overline{i} \overline{j} No swap

2 5 7 9 15 19 23 29 31

\leftarrow \overline{i} \overline{j}

Uploaded By: anonymous

A = 2 5 7 9 15 19 23 29 31

Sorted ✓

Code

```
void shellSort (A[]) {  
    int i, j, increment;  
    int temp;  
    int N = A.length;  
    for (gap = N/2; gap > 0; gap /= 2)  
        for (j = gap; j < N; j++) {  
            temp;  
            for (i = j - gap; i >= 0; i -= gap)  
                if (A[i] < A[i + gap]) {  
                    temp = A[i + gap];  
                    A[i + gap] = A[i];  
                    A[i] = temp;  
                }  
            else {  
                break;  
            }  
        }  
    }
```

هنا إذا كانت أقل من الجا
المرحلة

Swap ، بنفس الوقت يتسوف إذا
كانت يتطلع Backward

Time Complexity $\Rightarrow O(N^2)$.

Question

Suppose we have 5 GB of data using only 1 GB of Ram, what is the Best sorting algorithm could you use?

* عندی Data حصہ کیا اورام میں کیا جائے
وہی ترتیب سے ترتیب سے کیا جائے

↓
میں یہاں پہلے ہی فیکٹ

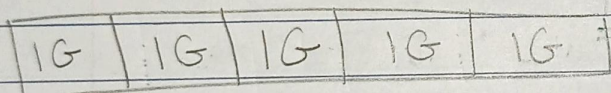
External Sort.

قسم و بدل Sort
ایک نوع کی ترتیب دینا
پس اگر اس کی ترتیب دینا
وہاں اس کی ترتیب دینا
Quick sort

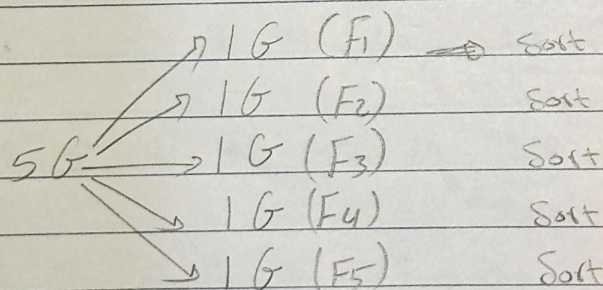
Example

5 GB of data (file) sorted in H.D.

File کو Divide کرنا



Load in to Memory
Sort use quick sort



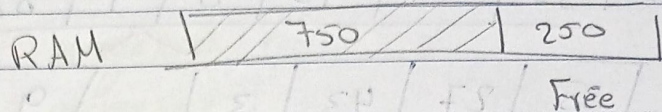
external
کہہ آنا اس کی ترتیب دینا
خارجیہ کی ترتیب دینا
وہاں اس کی ترتیب دینا
loaded
Uploaded By: anonymous

طِبِّ رَبِّهِ الْوَاحِدِ بِسْمِ آتَا بِدِي إِلَى الدَّائِمَا دَكُونُ حَرِيَّةَ !

* روفت من لال 1G ← 150 MG

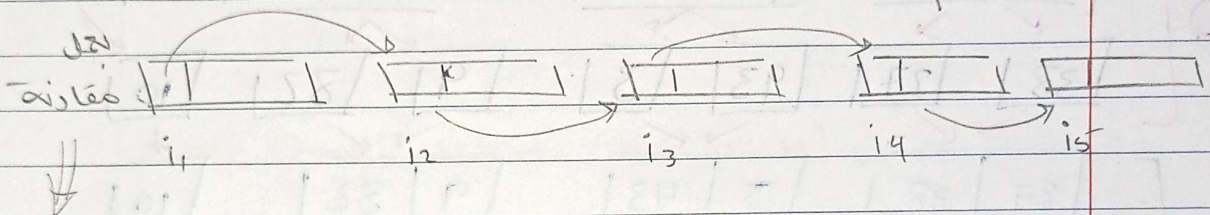
$$6150 - 750 = 5400$$

RAM 11GB 1G is loaded. 750 is used.



K-way Merge sort

* Merge sort و بزرگن آید *

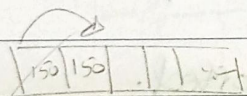


Compare \Rightarrow

وَبَيِّنُوا إِلَى النَّاسِ مَا كَانُوا لَا يَفْقَهُونَ ۖ وَلَا يَتْلُوا الْقُرْآنَ عُذُوًّا وَلَا نِيءًا ۚ

وفي مرات ١٥ ياءين قبل "هـ" ١٤ قبل ١٣

load another 150 MG $120 \leftarrow 120 \text{ gms}$



ویدیا ریڈیوس ۲۵۰ م

5 X 1024 Files ٥١ ٢٤ الى Out ٥١ ٢٤
250

کای الفایلیں - یطرح sorted

out 1
out 2
out 3

Merge File it's 5G
sorted.

Merge Sort

فلتره انه يقسم الـ array ويطبقها مرتبة
 Array ←

A = | 38 | 27 | 43 | 3 | 9 | 82 | 10 |

Divide

| 38 | 27 | 43 | 3 | | 9 | 82 | 10 |

| 38 | 27 | | 43 | 3 | | 9 | 82 | | 10 |

| 38 | | 27 | | 43 | | 3 | | 9 | | 82 | | 10 |

| 27 | 38 | | 3 | 43 | | 9 | 82 | | 10 |

| 3 | 27 | 38 | 43 | | 9 | 10 | 82 |

Conquer

| 3 | 9 | 10 | 27 | 38 | 43 | 82 |

Sorted.

Algorithm

```
void merge sort (int low, int high) {  
    if (low < high) {  
        int mid = (low + high) / 2;
```

Recursion

```
    {  
        merge sort (low, mid);  
        merge sort (mid + 1, high);  
        merge sort (low, mid, high);  
    }  
}
```

Time Complexity $\Rightarrow O(n \log n)$.

$$T(n) = \begin{cases} c & , n=1 \\ 2T(n/2) + cn & , n>1 \end{cases}$$

$$T(n) = 2T(n/2) + cn$$

$$T(n/2) = 2T(n/4) + cn/2$$

$$T(n) = 2(2T(n/4) + cn/2) + cn$$
$$= 2^2 T(n/4) + 2cn$$

$$= 2^k T(n/2^k) + kcn$$

$$T(n) = k T(1) + cn \log n$$

$$T(n) = n \log n$$

let

$$2^k = n$$
$$\log_x 2^k = \log n$$

$$k = \log n$$