Data Structer

COMP 242

آلاء سويلم

chapter 1

Recursion

[1] Hand shake problem

تسليم (كل حدا لازم يسلم على حدا تانى)

$h(2) = 1$, $h(3) = 2$, $h(4) = 3$ ...

base

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

Recursive part

```
Public static int handshake(int n){
        if (n == 2)
            return 1;
        else
            return handshake(n-1) + (n-1);
}
```

? $(n-1) + hand(n-1)$ ← * هيك أحسن أحسن

* هيك أحسن لأنه أقرب للفهم.

[2] عندي طابور وبدي أعرف قد إيش قد امي

```
Sum (int n) {
        if (n == 1)
            return 1;
        else
            return [n] + Sum(n-1);
}
```

هيك بحسب ال sum للعدد اللي بجاه

بستبدلها بـ [1] عشان بس يزيد واحد حتى يجمع

* بجوب دتم.

100 -

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

10

$$1 + Sum(n-1) = 1 + Sum(9)$$
$$1 + Sum(8)$$
$$1 + Sum(7)$$
$$\vdots$$

[3] Factorial

```
Public Static long Fact (int n) {
    if (n == 0)
        return 1;
    else
        return n*Fact (n-1);
}
```

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

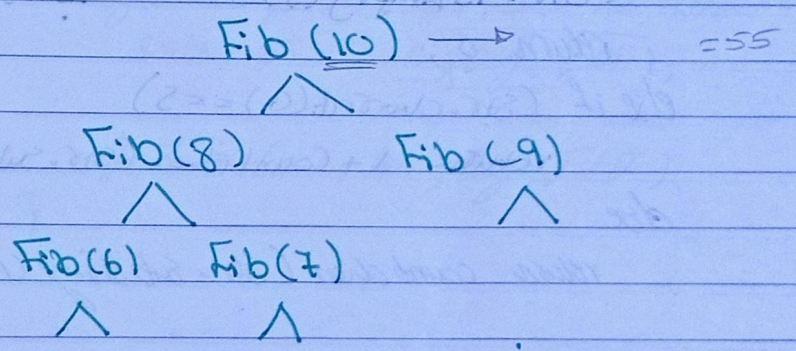
* يا إما بعطي الكود وبده الاتكس أو العكس.

[4] Fibonacci

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

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

```
Public static int Fib(int n){
    if (n==0)
        return 0;
    if (n==1)
        return 1;
    else
        Fib(n-1) + Fib(n-2);
```



5 Reverse

عكس الارقام : 123 ← 321

```
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

بعد عدد الاحرف.

```
Public Static int countchar( string Str , char s){
    if( Str.length() == 0)
        ( Str.isEmpty())
        return 0;
    else if (Str.charAt(0) == s)
        return 1 + Count Char (Str.substring(1) , s);
    else
        return countchar( Str.substring(1) , s);
}
```

بعد استبدالها ب

7 Count String

بعد كلمة جوا جملة.

```
int Count String( 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);
}
```

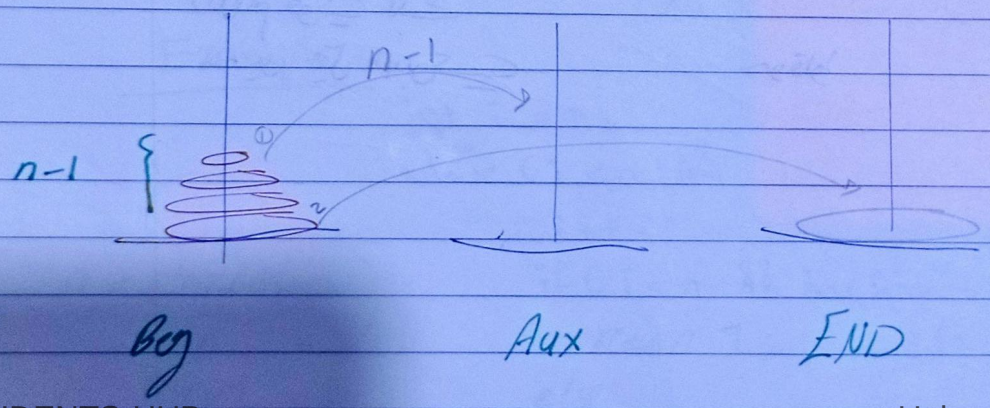
[8] Revers String.

```
public static String Reverce (String str ){
        if (str. is Empty ())
            return str ;
        else
            return Reverce (str.subString (1)) +
                    str . charAt (0);
}
```

nam olleh ← Hello man

[9] Hani Tower.

```
                    حن          بواسطة          إلى
HaniTower (int n , char Beg , char Aux , char END) {
        if (n>0){
                    حن          إلى
(نقل الى فوق) Hani Tower (n-1 , Beg , end , aux);
            System.out.printf ("\n Mov disk %d from %c to %c"
                Disk نقل آخر n , Beg , END);
                                حبا فوق
            HaniTower (n-1 , Aux , Beg , END );
}
```



Palindrome

```
Pablic Static boolean isPali (int [] arr , int i , int j){
    if (i > j)
        return True ;

    else if (arr [i] != arr [j] )
        return false ;

    else
        return isPali (arr, ++i, --j );
}
```

* Done

بس حلي اسئلة سنوات .

لا حول ولا قوة
إلا بالله العلي
العظيم - ه

اللهم قوني بك
حين يقل صبري

Analysis of Algorithm

chapter 2

(DS - AOA)

Problem Solving :-

Problem → A_1, A_2, A_3 → * لازم يكون ولك نفس ال Hardware & Software.

أهم اشي في الـ Algorithm
Speed (قديش آخذ وقت).
Space.

كشان أعرف أنو الـ Algorithm المفروض أجرب

1:- Experimantel → implement
timing (time();)
Plotting بشوف إذا الرسمة بتطلع فنكشن معروف

2:- Asymptotic → T(N)
All possibelton. جميع الاحتمالات.

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

في عندي :

Best case → 1

Average case → Array i element اي

worest case → N

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

* Best → Worest

في عندي 3 تعريفات :

Big-O → upper bound

Big-Ω → lower bound

Big-Θ → Average bound.

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

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

$\therefore O(n^2)$

* انا بوخذ ذلك order

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

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

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 $f(n) = 2n + 3$

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

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

$2n + 3 \leq 2n + 3n$ ✓

$2n + 3 \leq 5n$

* في الأقفا بنشيل ال constant

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

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

$2n + 3 \geq 1n$ ✓

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

$c_1 g(n) \leq f(n) \leq c_2\, g(n)$

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

Ex $n^2 \log n$.

$f(n) = O(n^2 \log n)$ $\quad f(n) \leq c\, g(n)$

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

$f(n) = \Omega(n^2 \log n)$ $\quad f(n) \geq c\, g(n)$

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

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

Ex $f(n) = n!$ $\quad 1 \leq 1 \times 2 \times 3 \times (n-1) \times n < n^n$

lower $\quad$ upper

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

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

$f(n) = \Theta$ unknown.

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

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

1 $f(n) + g(n) = \text{Max}(f(n), g(n))$ $\quad$ ج.

$(n^2 + n)$

2 $f(n) * g(n) = O(f(n) * g(n))$ $(O(n^3 \log n))$ $\quad$ خ

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

$f(n) = n^2 \log(n)$ $\quad \log n^2 \log(n)$ $\quad \log n(\log n)^{10}$

$g(n) = n(\log n)^{10}$ $\quad \frac{2\log n}{n^2} + \frac{\log n \log n}{n} > \frac{\log n}{n} + \frac{10 \log n \log n}{10n}$

$f(n) = 3n^{\sqrt{n}}$ $\quad 3n^{\sqrt{n}}$ $\quad 2^{\sqrt{n}\log n}$

$g(n) = 2^{\sqrt{n}\log n}$ $\quad 3n^{\sqrt{n}} = (n^{\sqrt{2}})^{\log_2 1}$

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

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

$g(n) = n^{\sqrt{n}}$ $\quad \log(n) \cancel{(\log_2)} < \sqrt{n} \log n$

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

$g(n) = 2^{2n}$ $\quad n \log_2 2$ $\quad \log_2 2^{2n}$

$\underline{n} \cancel{\log_2} < \underline{2n} \cancel{\log_2}$

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

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

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

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

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

```
Sum (A, n) {
    s = 0;                                  ①
    for (i = 0; i < n ; i++)                ③
        s = s + A[i];                       ②
    return s;                               ①
}
```

① $2n+2$

$2n$

Big O (n) $= 2n + 2 + 2n$

$= 4n + 4$

$= \boxed{n}$

Time complexity

$s = 0$ — ①

$i =$ — ①

Array — $\boxed{n}$

Space Complixity.

* اكم variable عندي

وبدهم يزيدوا عندي في ال Ram

```
Add (A, B, n) {
(n+1)   for (i = 0; i < n ; i++)        ①
n(n+1)      for (j = 0 ; j < n ; j++)   ②
n^2             C[i][j] = A[i][j] + B[i][j];
}
```

$C[i][j]$: n^2, $A[i][j]$: n^2, $B[i][j]$: n^2

Time $O(n) = 2n^2 + 2n + 1$

$= n^2$

Space. $3n^2 + 2$

$\boxed{n^2}$

```
Multi (A, B, n) {
n+1        for (i = 0 ; i < n ; i++)
n(n+1)         for (j = 0 ; j < n ; j++)
n x n (n+1)        for (k = 0 ; k < n ; k++)
  n x n x n            C = C + A + B ;
}
```

Time = $O(n^3)$

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

```
for (int i = 0; i < n; i++)          → O(n)
    for (int j = 0; j < n; j++) {    → O(n)
        ~~Start~~ Statment;
}
```

* الي أخذناه في المحاضرة
الي قبل >

Stop. $\boxed{i \geq n}$

```
for (int i = 0; i < n; i++).
    for (int j = 0; j < i; j++) {
        Statment;
}
```

i	j	time
0	0	0
1	1	1
2	2	2
⋮	⋮	
k	⋮	k

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

$$= \frac{n^2+n}{2} = \frac{1}{2} n^2 + \cancel{n}$$

$$* \boxed{O(n^2)}$$

```
P = 0;
for (int i = 1; P <= n; i++) {
    P = P + i;
}
```

P > n Stopeal.

$$P > n$$

$$\frac{k(k+1)}{2} > n$$

$$\sqrt{k^2} > \sqrt{n}$$

$$* \boxed{= O(\sqrt{n})}$$

i	P
1	0+1 = 1
2	0+1+2 = 3
3	1+2+3 = 6
4	1+2+3+4 = 10
⋮	⋮
k	1+2+3+4+ ... k ⟹ $\frac{k(k+1)}{2}$

```
for(int i=0; i≤n; i+=2){
    Statment;
}
```

$i = 0, 2, 4, 6 \ldots k$

Stop in $i > n$

$k > n$

$\therefore O(n)$.

* increment and decrement same time $(O())$

```
for(int i=1; i≤n; i*=2){
    Statment;
}
```

$i = 2^0, 2^1, \ldots 2^k$

Stop in $i > n$

$\log 2^k > n \log$

$\therefore O(\log n)$. $\leftarrow$ $k > \log n$

```
for(int i=n; i≥1; i/=2){
    Statment;
}
```

Stop in $i < 1$

$\log \frac{n}{2^k} < 1 \log$

$\therefore O(\log n)$

i		
n	$\rightarrow$	$\frac{n}{2^0}$
$\frac{n}{2}$	$\rightarrow$	$\frac{n}{2^1}$
$\vdots$		$\vdots$
$\frac{n}{m}$	$\rightarrow$	$\frac{n}{2^k}$

* جمع/طرح $\leftarrow$ $O(n)$

ضرب/قسمة $\leftarrow$ $O(\log n)$.

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

→ log(n)

log n مرات P

```
for (j = 1; j < P; j *= 2) {
    Statment;
}
```

→ log(n)

∴ log(log n).

```
for (i = 0; i < n; i++)
    for (j = 1; j < n; ~~j=j+2~~)
        {
            Statment
        }
```

→ n

→ log n

j = *2

∴ O(n log n).

```
P = 0;
for (int i = 1; P < n; i++) {
    P += i;
    for (int j = n; j > 1; j /= 2) {
        Statment
    }
}
```

$\sqrt{n}$

P = P + i

log n

∴ $O(\sqrt{n} \log n)$.

سواء for أو while

نفس النتيجة

Recursion.

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

* Print Down.

* لما تكون مع for بتبدأ أتجه وأطلع max

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

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

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

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

* بعد ما فهم جوا بعض

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

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

$\vdots$

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

* Assume $n-k = 0$

$n = k$

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

$T(k) = \cancel{1} + n$

$\therefore O(n)$.

```
void Test (int n){
    if (n>0)
        for (int i = 1; i<n; i*=2) ⇒ log n
            System.out.print(i);
        Test (n-1);
    }
```

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

$T(n-1) = T(n-2) + \log(n-1)$ ~ ①

$T(n-2) = T(n-3) + \log(n-2)$ ~ ②

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

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

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

$\vdots$

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

Asume $k-n=0$

$k=n$

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

$= \cancel{T(0)} + \log(1) + \log(2) + \sim + \log(n-1) + \log(n).$

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

$T(n) = \log n!$

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

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

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

$$T(n) = 4T(n/2) + n + 1$$
$$= 4[4T(n/2^2) + n/2 + 1] + n + 1$$
$$= 4^2 T(n/2^2) + 2n + 4 + n + 1$$
$$= 4^3 T(n/2^3) + 4^2 n/2^2 + 4 + 2n + 4 + n + 1$$

$$= 4^3 T(n/2^3) + 7n + \cancel{5}$$

constant

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

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

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

$$\log n = k$$

Big O (2^n)

$\boxed{*}$

* هاد حل السؤال الأخير
في آخر صفحة
في الكتاب كوبليكستي

$T(n) = T(n+1) + 1 \Rightarrow O(n)$

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

$= T(n) + \log n \Rightarrow O(n \log n)$

$= T(n) + n^2 \Rightarrow O(n^3)$

$= T(n) + n\log n \Rightarrow O(n^2 \log n)$

constant

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

$T(n-a) + b \quad O(nb)$

↳ فنش شرط وقيم ممكن فنكتشف

algo

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

مهم جداً

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

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

$T(n-2) = 2T(n-3) + 1$ ②

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

$= 2^2 T(n-2) + 2 + 1$ ~ by ①

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

$= 2^3 T(n-3) + 2^2 + 2^1 + 2^0$ ~ by ②

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

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

$$= 2^n + 2^{n-1} + 2^{n-2} + \cdots + 2^0$$ * بعوض $k=n$

$$\sum_{k=0}^{n} a^k = \frac{1-a^{n+1}}{1-a} \Rightarrow \sum_{i=0}^{n} 2^i = \frac{1-2^{n+1}}{1-2} = \boxed{2^{n+1}-1}$$

$$* \boxed{O(2^n)}$$

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

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

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

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

```
Public static int Test (int n){
        if (n == 0)
            return 0;
        else
            return 2T(n/2) + n;
}
```

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

$T(n/2) = 2T(n/2^2) + (n/2)$ ①

$T(n/2^2) = 2T(n/2^3) + (n/2^2)$ ②

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

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

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

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

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

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

* مهم جداً جداً

```
Test (int n){
    if (n==1)
        return 1;
    else{
        Print (n);
        return 4T(n/2) + n;
    }
}
```

*
Big O
2^n

$$T(n) = \begin{cases} 1 & , n==1 \\ 4T(n/2)+n+1 & , n>0 \end{cases}$$

Chapter 3 : Linked List

Node

| data | next |

Node → Data بربع ال

header → ماسكة الداتا من أول.

↳ location in Memory

```
class Node {
    Privete String name;
    Privete int id;
    Node next ;
```

عشان أعرف على وين يا أستر.

Operations:-

① insert ② delet ③ update ④ Traverse.

```
Public class Node {
    privet roll;
    privet int age;
    Node next ;
```

→ Node () { }

constructer . في

```
Node (int roll, int age){
    this. roll = roll;
    this. age = age;
    Next = null; }
```

+ وبدل getter , setter

وبدل toString ← اذ بدالها بدل printinfo

method بدل فيها شو

بدي اطبع

class SingleLL {

Driver

```
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]));

(while) }
}
```

File بدنا نقرأ

بدي أقطع

* insert في البداية.

```
public static Node insertLL (Node header, int roll, int age) {
    Node first = header;
```

هون انا جبت قيمة ال header

```
    Node newnode = new Node(roll, age);
```

انشا ال Node.

```
    if (first == null) {
        return newnode; header = newnode;
    }
    else {
        newnode.next = first;
        first = newnode; header = newnode;
    }
    return first;
}
```

إذا كانت فاضية.

مش فاضية.

Big O (1)

* constant

عشان أتأكد بعد method تطبع

شغلك

```
printList (header);
```

```
print List (Node header) {
    Node ptr = header;
    while ( ptr != null ) {
        ptr.printInfo();
        ptr = ptr.next;
    }
}
```

الـ method اللي عايزة تطبع المعلومات أو تغير أحد ال toString.

اطبع الداتا

وانتقل على اللي بعده

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;
    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;
```

خلي الـ header ← ptr

ومن بدل رقمين.

return newnode.

next.

اشير على اللي بعده

Array and linked list store collection of data.

Array ⇒ Store all element in one block of memory.

linkedlist ⇒ allocate memery for each element separately and only when necessary.

Disadvantages of Array.

1. Size is fixed. → حجمها ثابت بقدر شي اعدل عليها واذا تغير حجمها ديناميكياً رح تحتاج وقت من الذاكرة
ال Array اللي حجمها كبير في الها عيوب
① 90 أو 80 % بس بنستخدم من الاماكن والباقي بكون فاضي وبحتاج مساحة ع الفاضي.

2. Inserting and deleting elements in to the middle of array is potentially expensive because existing elements need to be shifted over to make room.
اضافة وحذف عناصر من منتصف ال Array مكلف جداً لأن العناصر الموجودة بدها تتحرك ع اليمين Shift وهيك اشي بيخلي.

Array versur linked list :-

مناسبة Array ⇒ insert and delete at end.
Randomly accesing any element وصول عشوائي
searching the list for a particular value. البحث عن قيمة معينة

linked list ⇒ Insert and Delete
Application where sequantial access is required تطبيقات بتحتاج وصول متسلسل
number of element can'n be predicted befor hand. الكائنات اللي ما بعرف اعدادها بعدد العناصر مسبقاً

```
public void insertlast (int data){
    Node ptr = header;
    Node newnode = new Node(data);

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

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 deletePossision (int key){
    Node ptr = head.next;
    Node prev = head;

إذا كانت فاضية  if (head == null){
            System.out.println("Linked List Empty!");
        }
        else{
            while (Ptr != null && Ptr.getdata() != key){
مش على اللي بعده    Prev = Ptr;
                    Ptr = ~~head.ne~~ Ptr.next;
                }

دور بكل اللينك دها لغى  if (Ptr == null){
الداتا.                 System.out.Print("Not found");
                }
                else{
                    Prev.next = Ptr.next;
الحذف ←             Ptr.next = null;
                }
            }
        }
```

double Linked List

Chapter 4

```
class Node {
    private key;
    Node next;
    Node prev;

    Node () {
    }

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

    getter , setter
    to string
```

هاد الاشي الجديد

```
Class DLinkedList {
    DLinkedList () {
        Node header , tailer ;
    }
```

~~next~~ ~~prev~~

header =

*كونستركتر

insert At First.

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

إذا كانت فاضية

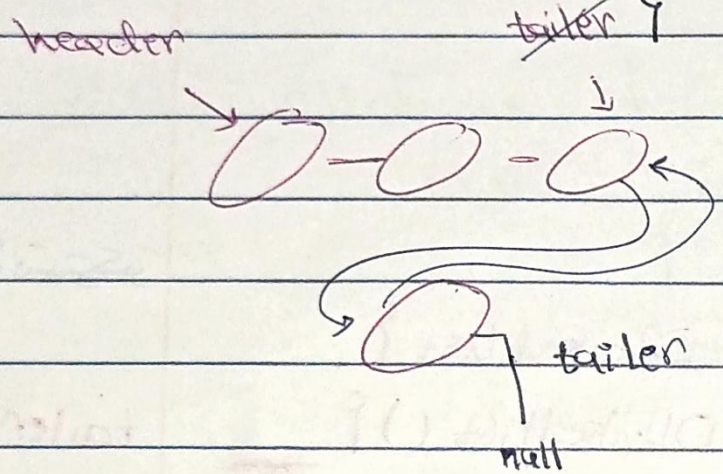
→ insertAtLastD1

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

Sergular: tailer.next = newnode;

header.prev = ← tailer = header = newnode;

إذا كانت Sergular ← ؟



Delete from position:-

```
public void deleteDLElement (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 {
            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;
                ptr.next = null ;
                Ptr.Prev = null ;
            }
```

~~Ptr =~~

prev = ptr;

ptr = ptr.next;

توضيح اسم كدم يكون عندي 3 شغلات

prev

current

next Ptr.

```
current = header
Prev = current
next ptr = current.next;
```

```
Prev.next = next Ptr;
next Ptr.prev = previous ;
current.next = null ;
    .prev = null
```

Prev curr next Ptr

~~Delete at first~~

Delete from first :-

```
public void deletefromfirst ( ) {
        Node ptr ;
        if (header == null) {
        ~o.print(" empty") ;
        }
        else {
            ptr = header.next ;
            header.next = null ;
            Ptr.prev = null ;
            heder = ptr ;
        }
    }
```

⇒

```
else if ( header.next == null )
        header = null ;
```

System.gc (); memory ⇐ شان تنظف ال *

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.net = null ;
            tailer = ptr ;
    } }
```

insert At position :-

```
public void insertAtposision (int data , int element){
    Node current = header;
    Node newnode = new Node (data);
    if (header == null ){
        print (" data not found");
        header = tailer = new node;
    }
    else{ while (current.next != null && current.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){
        count sort (a, p);
        p= p*10;
    }
```

phase.

element

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

number of digit

Range

* هل استقل على الـــــابن ؟ ؟

الحل ← 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;
}
```

* Couting Sort :- no comparssion, Sort according to keys counting the dome having distinct key values.

Ex

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

Range $\Rightarrow 0 \leq a[i] \leq 9$ ← k

Size = k+1

count

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

update/count

3	4	10	10	11	12	12	14	15	17

phase عد.

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

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

ترتيب في b  for (int i=n-1; i>=0; i--){   Big O(n)
            b[--count[a[i]] = a[i];      [a[i] /p %10]
        }
```

المشكلة في الـ Range كبيرة كتير

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

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

* لإيجاد مقدمة Radix

$O(n+k)$

Stack

O(1) insert ← Push عندي عمليتين مهمات ← ١
O(1) delete. ← Pop ٢

Stack over flow ⇒ لما تكون ال stack ممتلئة.
stack under flow ⇒ لما تكون ال stack فاضية. size (-1)

الشي اللي بآخر ← top

Peek () دقي عندي ~~back~~ ← هاد بيجيب اللي على الواجهة.
اكتف والا هاتة من النهاية.

FILO.

* نفس فكرة ال LinkedList بس تغير - مصطلحات

Application of stack:

داعباً يجي سؤال عليهن.

Main uses stack are:

1- Evaluation of Arithmetic. expression

مثال. * E = A - ((B+C) * D)

2- Execute recursive function.

depend on praiority:

1	( )	الأولوية للأقواس
2	^	القوة
3	* /	الضرب & القسمة
4	+ -	الجمع & الطرح

في السلايدات موجود هاد الكلام

infex → Postfix <x> <opp> <y> → <x> <y> <opp>

الوسط → الآخر

* حسب الي قوة عملية والعملية بالاول

infex → Prefix

الاول → الوسط <x> <opp> <y> → <opp> <x> <y>

* حسب الي قوة وعملية والعملية بالآخر

* التركيز على infex → Postfix

-++A/-B*CDEF/GH

Pre → in

قانون ضد العمليات بالاول ذ بدل

HG/FEDC*B-/A++- ← Reverse

<y> <op> <x>

Stack	Expression
C	HG
C/	HGF
C/	HGFEDC (x) (y)
C*	HGFEDC/ (x) B (y)
C-/	HGFE B*C/D A
C-	HGFE A/B*C/D
C+	HGF A/B*C/D-E
C+	HG A/B*C/D-E+F
C-	H A/B*C/D-E+F+G
	A/B*C/D-E+F+G-H → infix

Examples

1) $A+((B-C*D)/E)+F-G/H$ in → Post

<x><opp><y> → <x><y><opp>

CD*

PoP

)
D
*
C
-
B
(
(
+
A
(

⇒

PoP

)
E
/
BCD*-
(
+
A
(

⇒

+
BCD*E/
+
A
(

PoP

⇒

F
+
ABCD*E/+
(

PoP

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

PoP

⇒

GH/
-
ABCD*E/+F+

⇒

ABCD*E/+F+GH/-

Postfix

[2] A+((B-C*D)/E)+F-G/H in → Pre

<x><opp><y> → <opp><x><y>

*CD

)	)	-
D	E	F
*	/	+
C	-*CD	/-*CDE
-	C	+
B	+	A
(	A	C
(	(	
+		
A		
(		

PoP PoP PoP

)	-	
H	F	
/	+	
G	+A/-*CDE	
-	(	
++A/-*CDEF		
(		

PoP PoP /GH PoP

-++A/-*CDEF/GH

Prefix

[3] ABCD*-E/+F+GH/- Post → in

<x><y><opp> → <x><opp><y>

*		-		+	
D		(C*D)		E	
C	PoP	B	PoP	B-(C*D)	PoP
B		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

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

infex

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");
  }
```

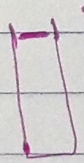
2 Pop

```
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) {
    Point (" ~~over~~ under flow");
  }
  else {
    int x = a [top];
    return x;
  }
}
```

هاد الستاكس
بوخذ آخر قيمة
موجودة في الـ Stack(1)
اللي على الوجه



Stack using Linked List :-

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

* يستخدم لما تكون الـ Stack فاضية.

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

* هي نفسها insert First

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

* هي نفسها delete ~~Last~~ First

4

```
int Peek () {
    if (header == null) {
        Print ln (" Stak empty!.");
    }
    else {
        return header . data;
    }
}
```

بترجع القيمة اللي على
الهيد .

Queue

مبدأ ال Queue ← زي الطابور.

FiFo — اللي بدخل أول بطلع أول

LiLo — اللي بدخل آخر شي بطلع آخر شي

* الإضافة على الآخر enqueue = List.insertFirst()
والحذف من الأول dequeue = List.deleteLast()

* بتحتاج 2 pointer ← front : بأشير على الأول
rear : بأشير على الآخر.

Stacks ⇒ solve problem works on recursion.
Queue ⇒ solve problem have sequantial ~~prossing~~ 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; }
```

← مبني على ال next

← الرير الجديد يصير ال newnode

← الرير نكست تاع ال newdata يصير هو ال front .

* اذا كانت ال Queue فاضية حطي ال rear و ال front بيساوي ال newnode و ال next لل rear حطي = ال front .

2. int deQueue () {

```
Node temp;
if (front == null) {  → rear = null;
    Print(" Empty!! ");
    return 0;
}
else {
    temp = front;
    front = front.next;
    rear.next = front;
    temp.next = null;
}
return temp;
}
```

Node temp = front; وبعدين فوق اقل

Running time

Linked List → - insert O(1)
- delete O(1)
- Find O(N)

Stack → Push O(1)
Pop O(1)

Queue → enQueue O(1)
deQueue O(1)

Stack	Queue
based in the Lifo or filo first in last out or last in first out.	based on the FIFO or LILO, first in first out or last in last out.
insertion and deleting in stacks take place only from one end of the listo called the top	insertion ~~tack~~ takes place at the rear of the list and deletion from front.
insertion called Push Delet called Pop	insertion called enQueue delete called deQueue
we need one pointer which always point to the last element called top	we need ~~to~~ two pointer front → first element inserted in the list and is still present, rear → always point to the last inserted element.
Solving problems work on recursion	solve problems having sequantial processing.

Chapter 4

Tree

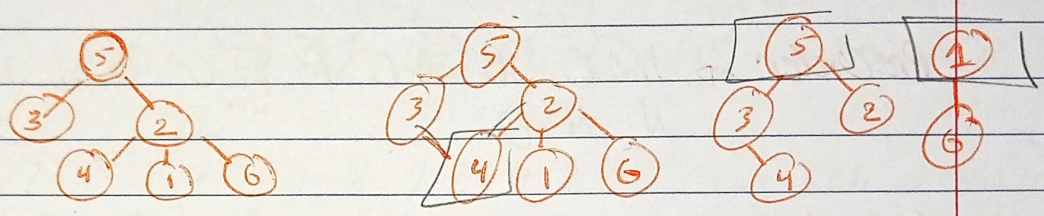
في شروط لل Tree :-

١. ال root هو عبارة عن Node بس فش إله أهل.

٢. كل Node إله أهل واحد.

٣. كل Node إلها صفر أو أكثر من الأولاد.

٤. عشان أوصل لل Node لازم يبقى إلها طريق واحد فقط.



tree.

Not tree
في لل (4) طريقين

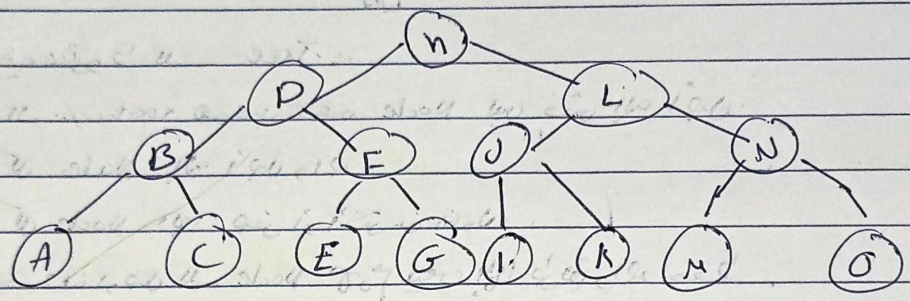
Not tree
في 2 root

مصطلحات مهمة

Ancestor
Proper ancestor
Decsendent
Proper descendent
Subtree
leafe
internal
Siblings
Size
Level
Hight

Tree traversal

1. Pre Order ⇒ root → left → right
2. In Order ⇒ left → root → right
3. Post Order ⇒ left → right → root.



PreOrder ⇒ h D B A C F E G L J I K N M O

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

Post Order ⇒ 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.

root

Pre (H) D B A C F E G L J I K N M O root lift right

Post A C B E G F D I K J M O N L (H) lift righ root.

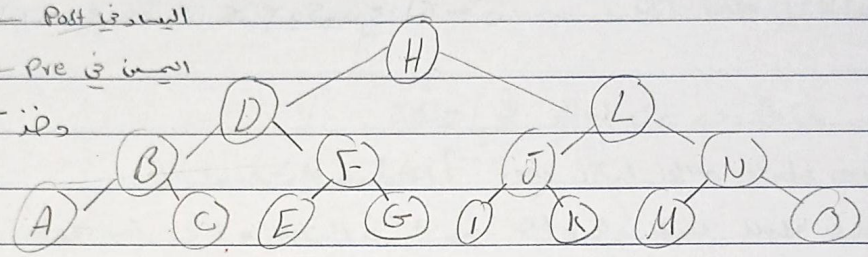
اليسار في Post ← يمين ال tree

اليمين في Pre ← يسار في ال tree

وخذ كل صورة كلمة.

مرة في ال Post

ومرة في ال Pre



*مهم جداً

Prefex (1) - + 9 9 + * 2 3 4 2 root → lift → right

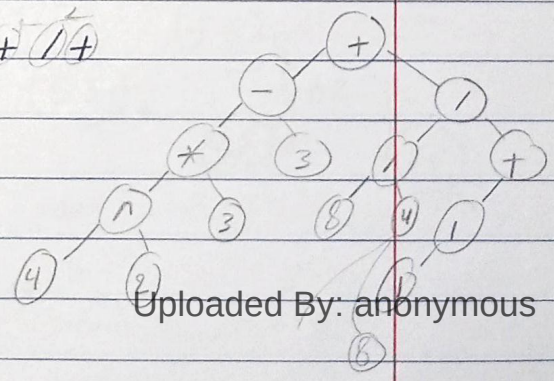
infex 9 + 9 - 2 * 3 + 4 / 2 lift → root → right

Postfix ⇒ ?

Pre ⇒ + - * ^ 4 2 3 3 1 1 8 4 + 11

infix ⇒ ?

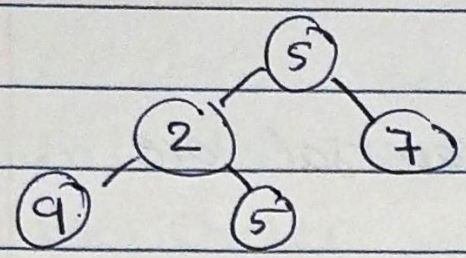
Post ⇒ 4 2 ^ 3 * 3 - 8 4 1 11 + 1 +



Binary tree

Tree عندها على الاقل Node 0

على الاكثر Node 2



Full Binary tree ⇒

Complete Binary tree ⇒ لازم تكون معبية من اليسار لليمين *

واذا كان في نقص لازم يكون في الاخر.

Binary search tree

$O(\log N)$

* كل شي اللي على اليمين بكون أكبر من ال root واللي على اليسار أصغر.

* عشان أضيف Node جديدة بشوف إذا أكبر من ال root روح على اليمين وإذا كان أكبر من ال Node اللي على يمين ال root روح يمين وهكذا حتى ألاقي مكان فاضي.

* بالنسبة للحذف بدي أواجه 3 حالات.

1. احذف Node مالها أولاد يعني leaf ← بحذفها وخلاص.
2. احذف Node عندها ولد واحد ← بحذف ال Node وبخلي ابنها يواجه محلها.
3. احذف Node عندها ولدين ← كل شي أواجه ال Min أو ال Max لل Node اللي بدي احذفها وأحطه مكانه.

~~Min ← أكبر اسم في ال left (أخصص)~~
~~Max ← أكبر اسم في ال right (أخصص).~~

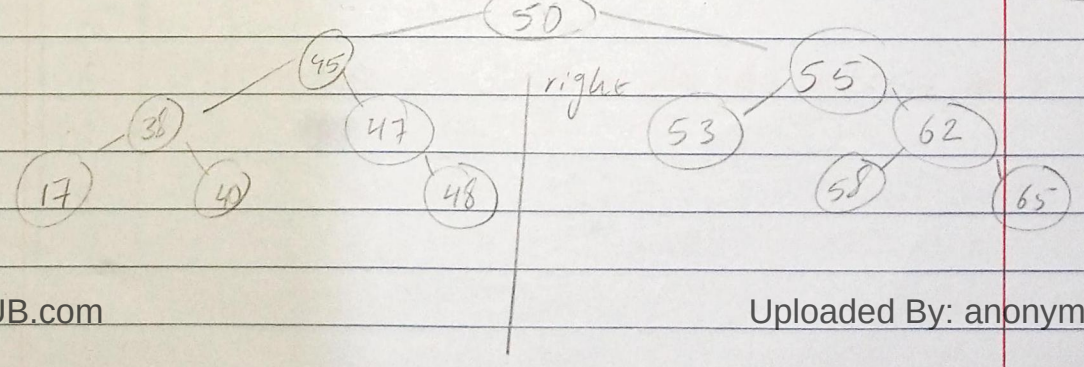
Min ⇒ أصغر اسم في ال right
Max ⇒ أكبر اسم في ال left

* يكون فيه إلها left
* يكون فيه إلها right

Ex

Post ⇒ 17 40 38 48 47 45 53 58 65 62 55 50 (root)

in ⇒ 17 38 40 45 47 48 50 53 55 ~~57~~ 58 62 65



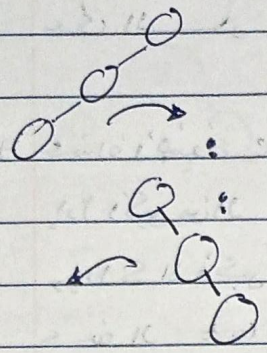
AVL tree

* هي Binary search tree

لكن كمان فيها شرط ⇐ height balanced

(1-, 0, 1)

عشان اعمل ال tree balance ⇐ Rotation نعمل



: Right Rotation ⇐ Rotation

: Left Rotation ⇐ Rotation

Single Rotation → Right
→ left

Double Rotation → left Right
→ Right left.

* لما بعمل insert أو delete كمان لازم اتأكد ان ال Balanced

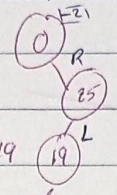
* شرح مثال وكود

شرح القسم الثاني

Insert an integer number cencecuating in an empty AVL tree. *

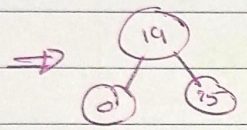
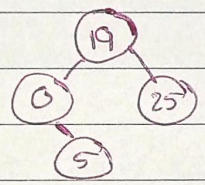
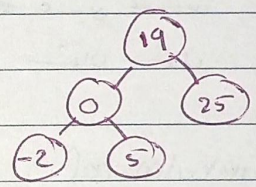
1) 0, 25, 19, 5, -2, 28, 13, -5, 2, 6, 14, 7

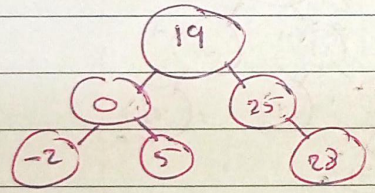
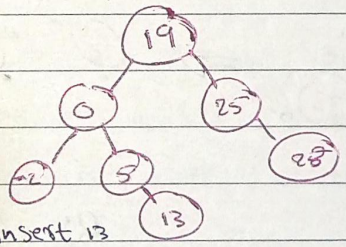
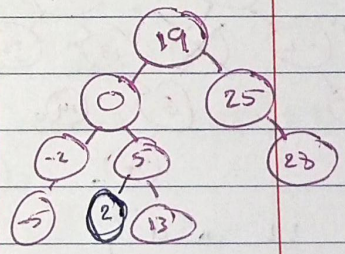
2) Delete node 19 using next successors.

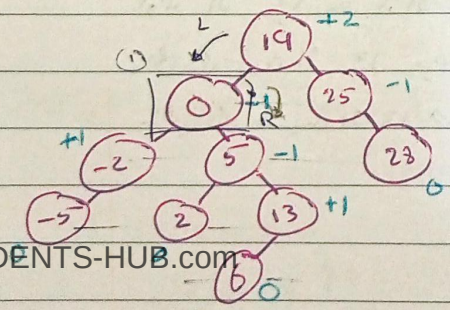
		
insert 0	insert 25	insert 19
Balanced	Balanced	imbalacede
no need to balance	no need to balance	need to balanced Problem : RL

* الحل يكون اول ما ال 19 واعلها راس ويجوبن بعمل لفة لل 0 ←

Solve : RL rotation

		
after double rotation R → L	insert 5 ✓	insert -2 ✓

		
inser 28. ✓	insert 13 ✓	insert -5, 2 ✓



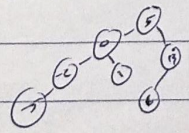
imbalanced

need to balanced

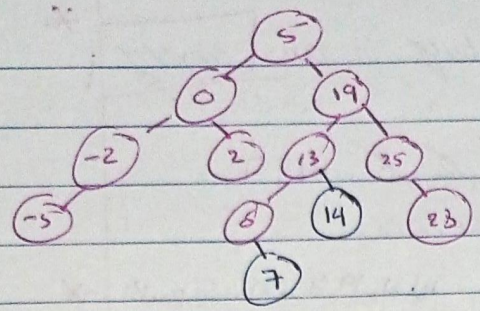
Problem ⇒ LR

Solve ⇒ LR Rotation

* ال 5 بيصير مكان ال 0 وال 2 بيصير على يمين ال 0



insert 6



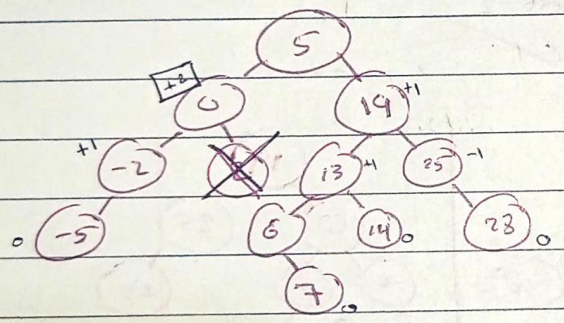
هسا بيصير ال 5 مكان ال 19
والي كان عين ال 5 بصير كان
يمين ال 19

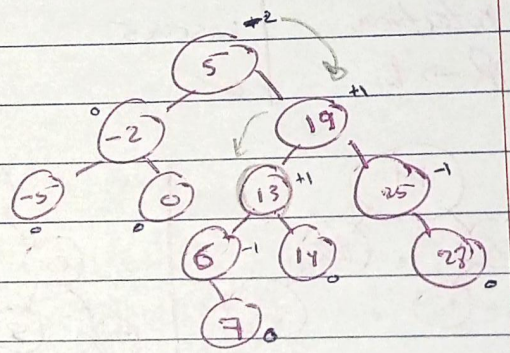
insert 14 → Balanced

insert 7 → Balanced

* كيف أعرف اذا شجرة بين
الي بحدد ال Balanced ← + ← يمين
− ← يسار
0 ← عين عين
شمال شمال.

Delete 2 ⇒ imbalanced

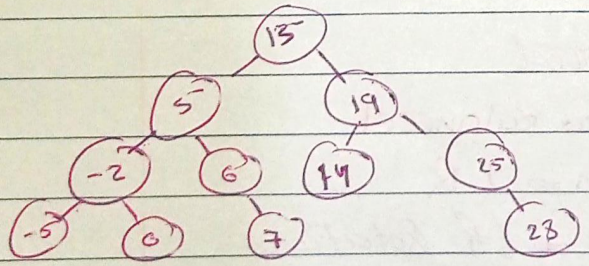




اذا كانت LL
اذا كانت RR

عندي مشكلة في 5 ← RL
اذا ← RL

بوظف ال 13 مكان ال 5
والي كان يمين ال 13 بصير كان عين ال 5
والي كان عين ال 13 // شمال ال 19



Balanced

Tree

```
Node tree {
    object data;
    Node left;
    Node right;

    Node (object data) {
        this.data = data;
    }

class tree {
    Private Node root;

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

    private void insert (Node root, Object data) {
        if (root == null) {
            return newnode (data);
        }
        else if (root.getData() < data) {
            return root.setRight (insert (root.getdata, data));
        else if (root.getdata() > data)
            return root.setLift (insert (root.getdata(), data));
        }
}
```

مصنوع في ال main
أكشف عن ال root
ذاتاً كـ privet

ولعمل method عشان
وحدة Privet وحدة
Public عشان توصل لها
واستدعيها في ال main

Node

Right

أكبر دوس عين

Left.

أصغر دوس شمال

* لعمل recursion لازم
يكون في

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

Private void InOrder (Node root) {
        if (root == null)
PreOrder    InOrder ( root . get left ());
            System.out.println( root . get data() );
            InOrder ( root . get Right ());
    }
```

PostOrder

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

B-Trees

DataBase *

[1] كل ال leave في نفس ال level

[2] ال Node اللي بين ال root وال leafe إلها m أو أعلى الاقل $\frac{m}{2}$ Childe

[3] ? عدد ال element اللي جوا أكثر شي m-1 وأقل شي m/2

[4] كل root إله 2 Child أو يكون فاضي.

* أهم شي الترتيب :-

Example

* Creat a B-tree of order 5

AGFBKDHMJES1RXCLNTUP

* لما يصير أكبر من ال size بعمل split وعملية الانقسام OP للي بالنص

* لما بدي أضيف K بعمل split وبيطلع F فوق ← وأكبر على اليمين أصغر على اليسار. وهكذا الترتيب.

~~AGF~~ (1) ~~ABFG~~ هون

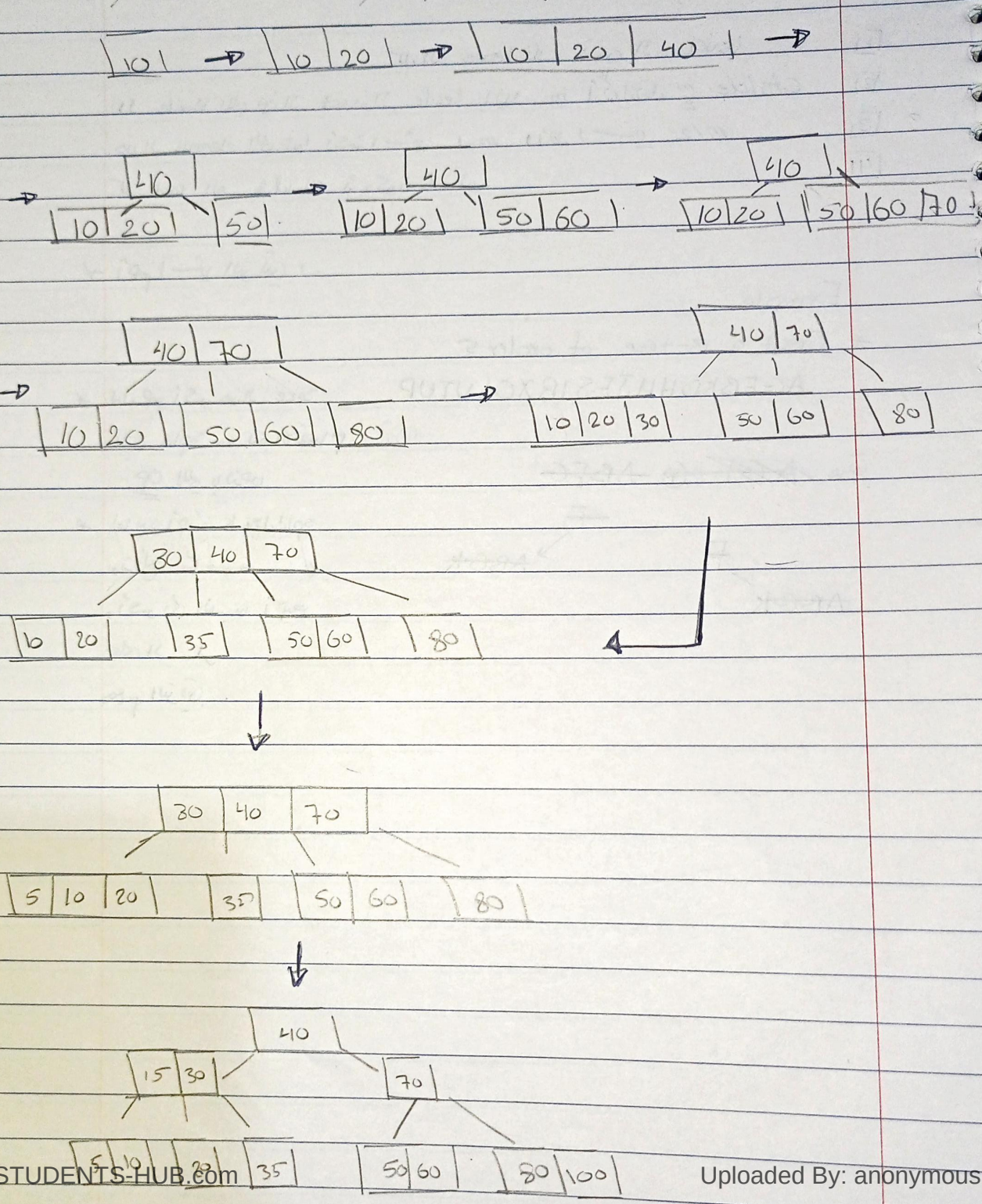
~~F~~

~~F~~ ~~ABGK~~

~~ABGK~~

4

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



Chapter 5

Hashing

* هون كنا بنستعمل ال array

وال array ← Fixed size

- linear search ⇒ $O(N)$
- Binary search ⇒ $O(\log N)$
- if we know index ⇒ $O(1)$

* عشان أوصل لـ $O(1)$ بدها كل رقم في ال index تاعه في ال array

يعني ١ ← بدها في index بس لو كان حجم ال array مثلاً

١٠ يعني من ٠ ← ٩ وبدي أحطّ الرقم ١٠ بدهي عندي

مشكلة.

عشان أحلها بستخدم مفهوم ال Hashing و بستخدم هاي المعادلة

[X % table size]

insert 10 ⇒ 10 % 10 = 0

insert 13 ⇒ 13 % 10 = 3

ملبانة !!

فشو بعمل بروح على اللي بعده لحتى لاقي مكان فاضي

هاي الـ collision اسمها

0	10
1	
2	
3	3
4	4
5	5
6	6
7	13
8	8
9	

* هاي بعد وقت رح تنتهي بعض: out of array size

عشان أحل مشكلة ال collision

في 2 solve

close

1- Separate chaining : array of linked list بستخدم فيها

2- open addressing :

1- Linear probing ⇒ linear search

2- Quadratic probing ⇒ nonlinear search

3- Double hashing ⇒ use two hash function.

أفضل اشي

Separate chaining ⇒ Close.

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

table size 7

h(key) = key % 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

0	21 ← 14 ← 7
1	8 ← 15
2	23
3	
4	
5	
6	13

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

* هاي ما بتنتاك لأنه بستخدم linked list وتعب .

* هاي بس مشكلتها لين بدي اتعامل مع ال Pointer في ال memory ,

* ال Hashing مهم جداً في عملية ال insert
searching

* وتنطبق على ال DUA ، increbtion أو decrebtion.

Example:

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

1- Onion ⇒ O = 111 n = 110 i = 105

hash (onion) = (111 + 110 + 105 + 111 + 110) % 13

= 1

2- Salt ⇒ S = 115 a = 97 l = 108 t = 116

has (salt) = (115 + 97 + 108 + 116) % 13

= 7

0 → okra → potato

1 → onion → carrot

2

3

4 → cabbage

5

6 → mushroom

7 → salt

8

9 → cucumber

10 → tomato → mellon → olive

11 → banana

12 → orange.

open addressing :-

* $h_i(key) = [h(key) + f(i)] \% n$ ⟹ لكل الحالات

↳ linear
Quadratic
double

* $f(0) = 0$ بعض الأنظمة عن النوع

$f(0) \% n$, $f(1) \% n$ ~ $f(n-1) \% \boxed{n}$ → table size

value ⟹ between $\boxed{0 \text{ and } n-1}$

* linear $f(i) = i$
* Quadratic $f(i) = i^2$
* double $f(i) = i * h_p(key)$

بالعادة بنختار ال table size يكون Prime (عدد اولي)

~~حل ال collision~~

لازم بعد حل عملية ال collision أطلع ال load factor

ولازم يكون بين (0.6 , 0.7)

λ = number of element / table size.

وإذا كان أكبر من 0.7 ← undesirable ~~under~~

فبعمل rehash

Type of open addressing :-

[1] linear

Example

insert (18), (26) (35), (9), find(15) (48)
delet (35), (40) find (9), insert (64), (47)
find (35).

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

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

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

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

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

$h_0(35) = ((35 \% 13) + 0) \% 13 = 9 \% 13 = 9$

$h_0(9) = (9 \% 13 + 0) \% 13 = 9 \% 13 = 9$ → collision

$h_1(9) = (9 \% 13 + 1) \% 13 = 10 \% 13 = 10$

$h_0(15) = (15 \% 13 + 0) \% 13 = 2 \% 13 = 2$ (notfound)

$h_0(48) = ((48 \% 13) + 0) \% 13 = 9 \% 13 = 9$

لقى في رقم بس هش 48 ف بروح بدور على اللي بعد

$h_1(48) = (48 \% 13 + 1) \% 13 = 10 \% 13 = 10$

ما لقى 48 وبرد يعمل زي فوق

$h_2(48) = (48 \% 13) + 2 \% 13 = 11$ (Not found)

$h_0(35) = 9$ index of 9 has a value 35

فبعمل Change Status وبتحول من E ← D

$h_0(40) = (40 \% 13 + 0) \% 13 = 1 \% 13 = 1$ (Notfound)

لأنه ال status E مش بعمل Delete

$h_0(9) = (9 \% 13 + 0) \% 13 = 9 \% 13 = 9$ بروح (D)

$h_1(9) = 9 \% 13 + 1 \% 13 = 10$ وبروح لـ h_1

بحول ال status من D ← D

$h_0(64) = (64 \% 13 + 0) \% 13 = 12$

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

$h_0(35) = 9$ found

0	26
1	
2	
3	
4	
5	15
6	
7	
8	47
9	35
10	9
11	64
12	

E Empty
D Delete
O Oqubay

2 Quadratic

$f(i) = i^2$

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

* ملى نفس المشاكل الـ ربع هنا

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

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

$h_0(39) = (39\%13 + 0^2)\%13 = 0$ collision

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

$h_0(26) = (26\%13 + 0^2)\%13 = 0$ collision

$h_1(26) = 1$ collision

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

* كل ما زاد الـ collision بتزيد المشاكل
الحجم أكثر من 70% مشكلة وتشو بعمل
بنشأ Array جديدة ودضاعفها X2
وبتخزن العناصر في مواقع جديدة.

Primary Clustering ← دا ~~معه~~ متجمعة حولين
بعض بسبب كثرة الـ
collision.

عشان اضمن انه الـ table size Prime لازم في هاي
المعادلة

* $f(x) = x^2 + x + 41$

* في بالسلايدات الفرق
بين الـ open & الـ close

Secondary clustering

* تجمعات بس مش بنفس الخلية
المكان (بيكون في تفرعات)
وهو أهون من ال primary cluster

3 Double Hashing:

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

* $f(i) = i * \underline{hp}(key)$.

is prime $\Rightarrow$ $hp = 1 + key \% (\text{table size} - 1)$
$= q - (key \% q)$
$q * (key \% q)$

q is prime #

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

* table size 13
* $hp = 1 + key \% 12$

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

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

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

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

$h_1(18) = (18 \% 13 + (1 * (1 + 18 \% 12)) \% 13 =$
$(5 + (1 * 7)) \% 13 = 12$

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

$h_1(96) = (96 \% 13 + (1 * (1 + 96 \% 12)) \% 13$
$= (5 + 1) \% 13 = 6$

	Double
0	26
1	
2	
3	
4	
5	70
6	96
7	
8	
9	
10	
11	
12	18

Exercise 4-

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

1- insert ~~29~~ 29.

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

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

1- $h_0(29) = (29\%7 + 0)\%7$

$= 1\%7 = 1$ collision

$h_1(29) = (29\%7 + 1)\%7$

$= 2\%7 = 2$ collision

$h_2(29) = (29\%7 + 2)\%7$

$= 3\%7 = 3$ collision

$h_3(29) = (29\%7 + 3)\%7$

$= 4\%7 = 4$ inserted.

* عنا مشكلة (كثرة ال colision) وشو بدل ال array بضاعف ال array 2x
بصير عندنا حجم ال array = 14 ف اقرب Prime# هو ال 17
وبدل insert جديدة باستخدام ال Double hash

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	

* حل مشكلة ال String

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

Array

بوخذ ال ASCII Code للحرف
ونعمل كل هاي ال Method

```
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;

    if(hashVal<0)
        hashVal += table_size;

    return hashVal;
}
```

بدل ما اضرب بـ 32
كل مرة shifting

عشان ما يصير
over flow.
ونرجع لو صغيرة
ال positive

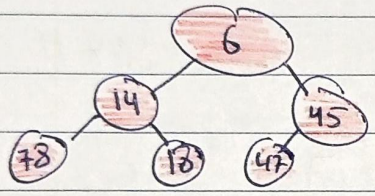
HEAP

Chapter 6

Binary Heap

1. Complete binary Tree → مليانة من فوق لتحت ومن اليسار لليمين

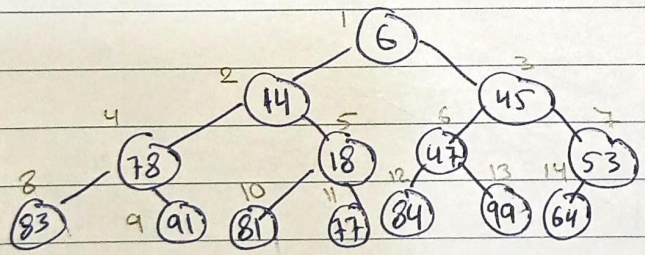
2. Min-heap ordered → الأولاد لازم يكونوا أكبر أو يساوي الأب



→ مو مهم اللي ع اليمين أكبر أو اللي ع اليسار وكمان مش مهم التكرار.

Minimum element is in the root

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



→ Tree هيك بعبي ال بس برمجيا

Parent (i) = i/2

left (i) = 2i

Right (i) = 2i+1

Insert

insert into next available solt

Bubble up until it's heap ordered

وبين في فاضي بالمصفوفة

وبعدين نشوف اذا هي

heap ordered ← اذا لا بدل Swape

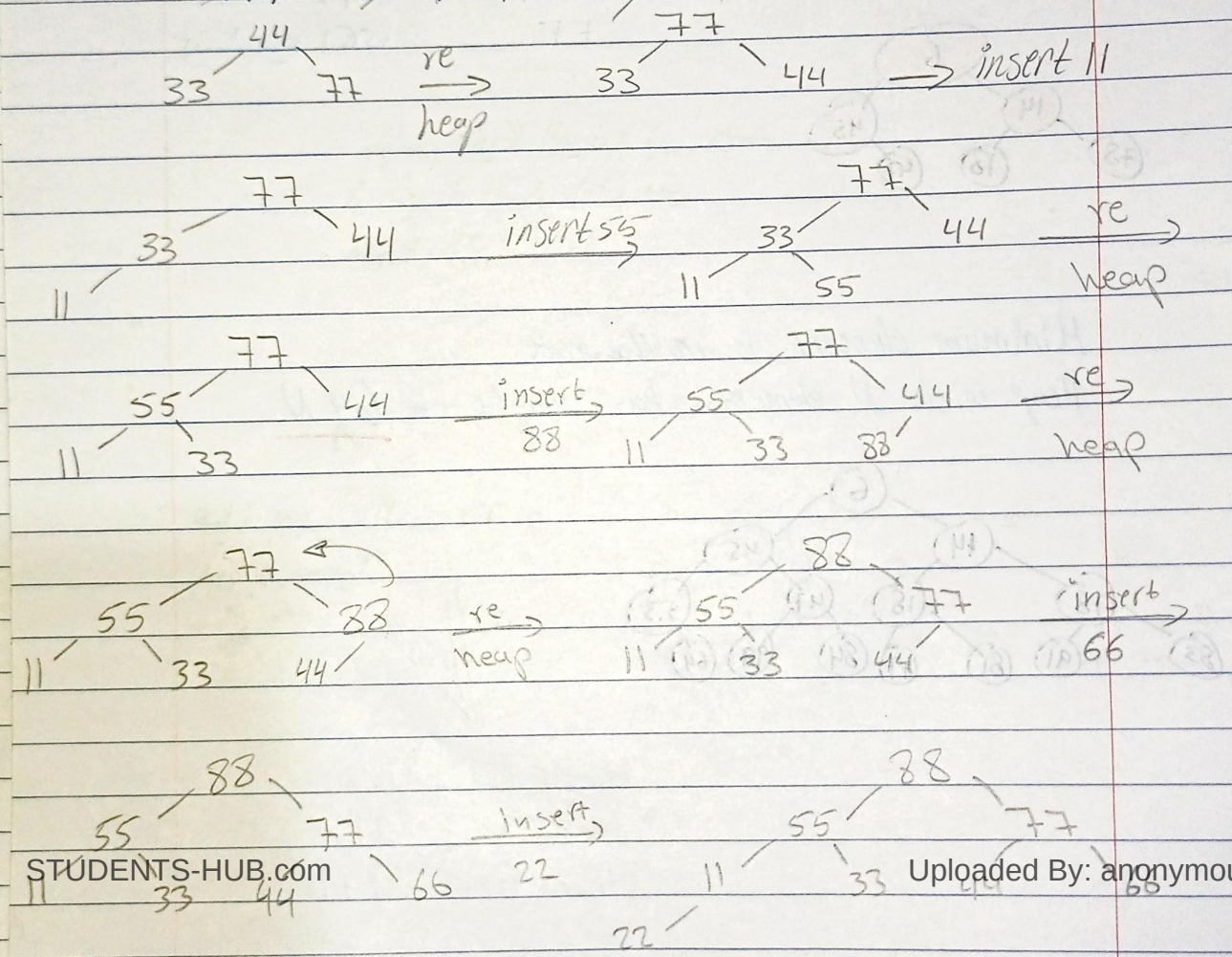
(اكبر من الاب) ← اكبر من الاب

Time Complexity $O(\log N)$

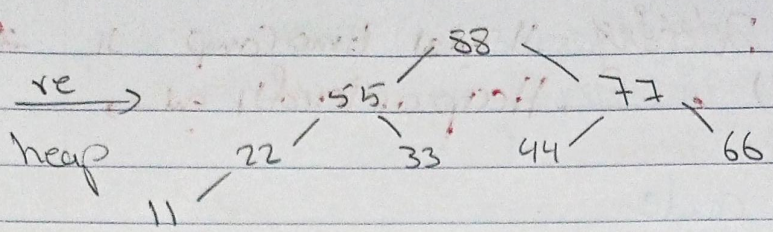
Example

Bulid **Maximum** heap tree?

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



re heap



Done

Delete minimum element from heap.

exchange root with rightmost leaf.

* أخر ابن على اليمين

Bubble root down until it's heap ordered.

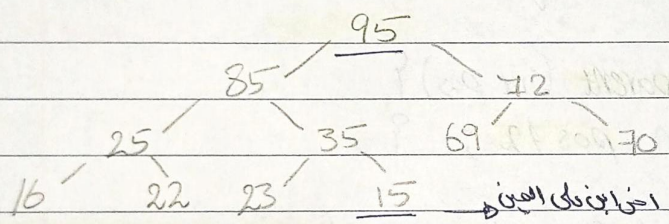
* بتحقق اذا الاب اصغر من الابناء.

* وبنقص من حجم الداتا 1

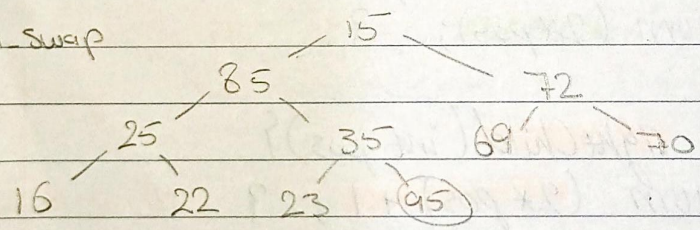
Example

Delete root 95

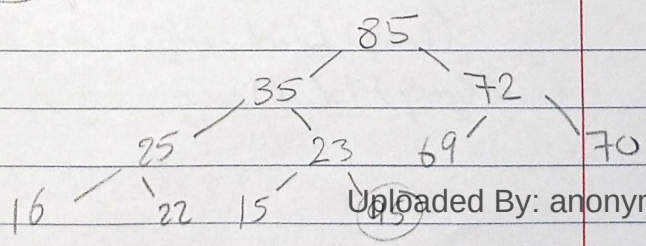
maximum Heap



1_ Swap



2_ Heap ordered



Done

* الـ time Comp لعملية الـ Deleting $O(\log N)$
وعملية الـ Heap ordered $O(n \log N)$ ←

Min Heap code

```
public class MinHeap {
    privet int [] Heap;
    privet int size;        → عشان اتتبع size
    privet int maxsize;     → حجم كل الأوراي
    privet static final int FRONT = 1;
```

راجع ! — خالد

Constructor →

```
    public MinHeap (int maxsize) {
        this.maxSize = maxsize;
        this.size = 0;
        Heap = new int [this.maxSize +1];
        Heap [0] = Integer.MIN_VALUE; }

    privet int parent (int pos) {
        return pos/2; }

    privet int leftChild (int pos) {
        return (2*pos); }

    privet int rightChild (int pos) {
        return (2*pos) +1; }
```

```
public void insert (int element){
    if (size >= ~~ele~~ maxSize){
        return; }  → insert اطلع فنش حبال تعمل
    Heap[++size] = element;
    int current = size;  → موشتو لل size
```

```
    while (Heap[current] < Heap[Parent(current)]){  ← بقارن الاب مع الابن
        swap(current, parent(current));  ← ميشود
        current = parent(current); }
    }
```

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

```
~~public~~ privet void minHeapify(int pos){
    if (! isleaf (pos)){  ← فش إله اولاد
        if (Heap[pos] > Heap[leftChild(pos)]  or  ||
            Heap[pos] > Heap[rightChild(pos)]){
        if (Heap[leftChild(pos)] < Heap[rightChild(pos)])
        {
            swap(pos, leftChild(pos));  اللي على اليسار اصغر
            minHeapify(leftChild(pos));  اعمل swap
        }
    else{  ← اللي على اليمين اصغر
            swap(pos, rightChild(pos));
            minHeapify(rightChild(pos)); }
    }
}
```

إله اولاد فش داعي اعمل اشي

```
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 --] ;
    min Heapify (FRONT);
    return popped;
}
```

يبدل اول element مع
اخر element

Heap Sort

1. Bulid 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. اعلى اشي
3. Repeat above steps while size of heap is greater than 1.

Time Complexity :

Heapify ⇒ $O(\log N)$

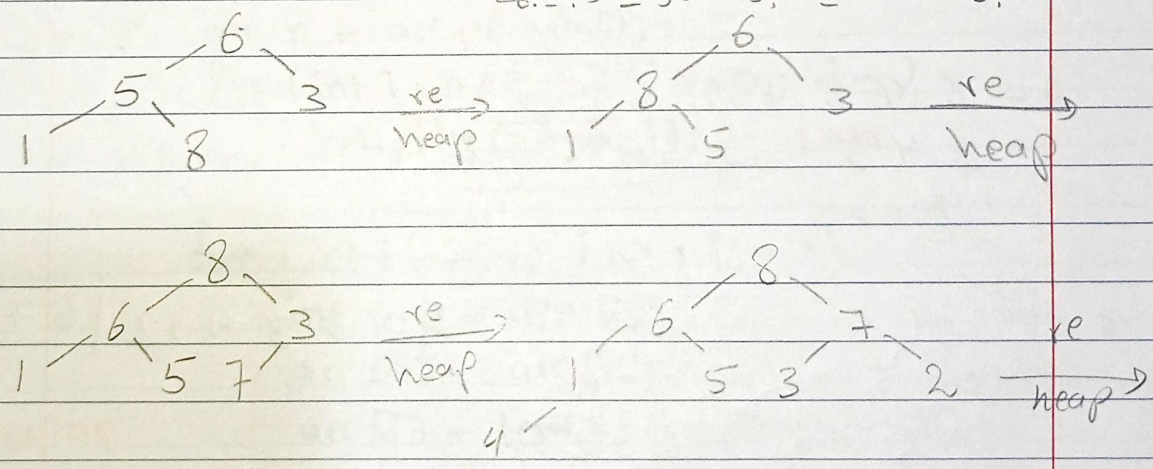
Bulid Heap ⇒ $O(N)$

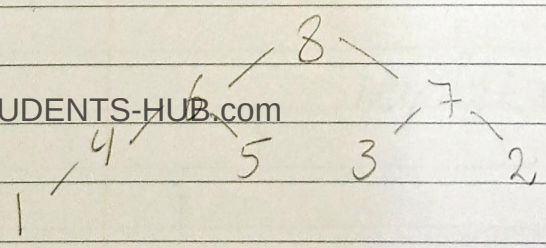
Heap sort ⇒ $O(N \log N)$.

Example

A = [6 5 3 1 8 7 2 4] full Array. size = 8

1. (Heapify) بدل كود ال
بنوخذ ال Array و ال size.
وبقى الداتا اللي جوا الاراي وبعيبها





⇒ 8 6 7 4 5 3 2 1

Front

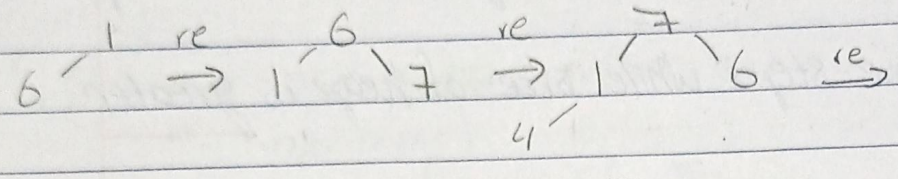
A = [8 6 7 4 5 3 2 1]

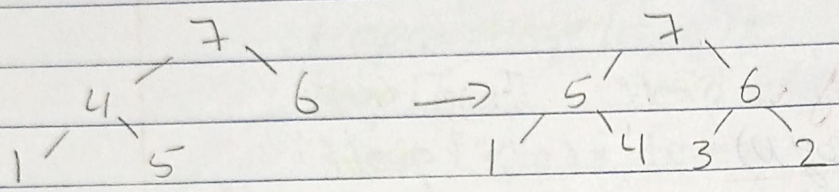
هس بدل remove لل 8

A = 1 6 7 4 5 3 2 8

size = 7

وبعدين بدل Build لل Heap



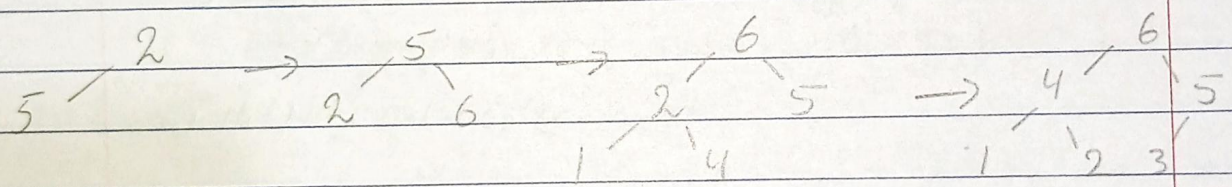


A = 7 5 6 1 4 3 2 8 1

remove 7

2 5 6 1 4 3 7 8 1

size = 6

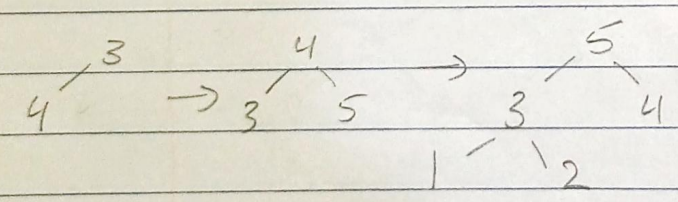


A = 6 4 5 1 2 3 7 8

3 4 5 1 2 6 7 8

remove 6

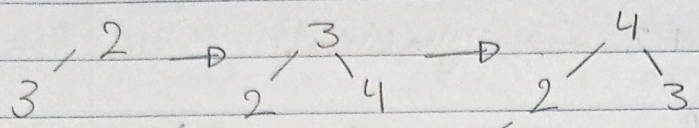
size = 5



A = 5 3 4 1 2 6 7 8

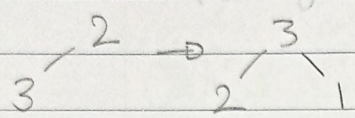
A = 5 3 4 1 2 | 6 7 8 |

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



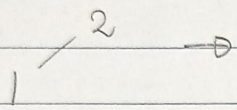
A = 4 2 3 1 | 5 6 7 8 |

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 |

remove 2

A = 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] = temp;
```

swap

```
        heapify (arr, i, 0);
    }
}
```

index

```
public void heapify(int arr[], int n, int i){
    int largest = i          → root
    int l = 2i + 1           → left Child
    int r = 2i + 2           → right Child.

    if(l < n && arr[l] > arr[largest])
        largest = l;

    if(r < n && arr[r] > arr[largest])
        largest = r;

    if (largest != i) {
        int swap = arr[i];
        arr[i] = arr[largest];
        arr[largest] = swap;

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

الابن اليسار أكبر من الـ root

الابن اليمين أكبر

swap أعمل

Radix Sort

هدفي شحليتين في ال Sorting بشكل عام

1- No extra space ← اي ما أحتاج Array جديدة

2- Time Complexity.

Radix Sort → $O(N \log N)$ with space

Selection and Bubble Sort → $O(N^2)$ No space

Heap Sort → $O(N \log N)$ No space

Quick Sort

هاد النوع مبني على ال Pivot

المحور

الداتا بتتكون حول محور معين

من هدول الاربعة ←

bad 1- First element pivot

bad 2- Last " "

bad 3- random " "

good 4 median " "

Example:

$$A = [\overset{L}{7}\ 6\ 10\ 5\ 9\ 2\ 1\ 15\ \overset{h}{7}]$$

Pivot first element ⇒ Pivot = A[L];

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

بغط ال pivot مكان بحيث يصير

كل الداتا الاصغر منه على اليسار

والاكبر على اليمين.

يسار	7	يمين
7>		7<

divide & conquer

* نقسم ونجمع

7	6	10	5	9	2	1	15	7

Start (s) — swap — end (e)

Pivot = 7

* 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.

* وبعدين نعمل swap

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

No swap because $s > e$ (crose)

swap between element end with pivot.

Pivot

2	6	7	5	1	7	9	15	10

Pivot = 2

less — proper place — great

* هسا بوخذ كل جزء لحال.
اليمين لحال واليسار لحال

Algorithim

```
Partion (a, l, h){
    int start, end, pivot;
    pivot = a[l];
    while (start < end){
        while (a[start] <= pivot){
            start ++; }
        while (a[end] > pivon){
            end --; }
        if (start < end){
            swap(a[start], a[end]);
        }
        swap(a[l], a[end]);
    }
    return end;
}
```

لما تلاقي رقم أكبر من ال Pivot وقف

اذا لقيت رقم أصغر وقف

* اذا صار تقاطع ⇐ بدل بين ال end مع ال index قلع البداية

عشان أعرف وين مكان ال Pivot تماماً

```
Qsort (a, l, h){
    int loc = Partion (a, l, h);
    Qsort (a, l, loc-1);
    Qsort (a, loc+1, h);
}
```

أخذ الاراي (اليسار)

(اليمين)

* Best Case ⇒ $O(N \log N)$

* Worest Case ⇒ $O(N^2)$

حاولت ال Pivot ←

```
اول   Pivot = a[L];
اخر   Pivot = a[h];
      Pivot = randome(a);
```

فنكشن

```
      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];
}
```

بدل الرقم اللي بالداخل مع اللي قبل الاخير

رجعلي ال Pivot

Insertion Sort

بتشتغل شغل لعب الشدة.

Exercise

	0	1	2	3	4	5
A =	5	4	10	1	6	2

sort | unsort

الفكرة ← بقسم الكراي لـ
1- sorted list
2- unsorted list

temp [4]

index 1 بياخد
يعني بنطرح 4 جواه

وبقارن اللي جوا ال temp مع اللي موجود في sorted list. ←

* هل اللي بال list sort أكبر من اللي بال temp ؟
إذا أكبر بعمل shift to the right.
وبقارن باقي العناصر اللي ضلت مع ال temp
إذا لقيت عنصر أصغر من ال temp بوقف.

A = 4 ? 5 | 10 1 6 2

sort | unsort

temp [10]

وبرجع بقارن 10 مع 5 & 4
مش أكبر من 10 بدخل عليها

A = 4 5 10 | 1 6 2

sort | unsort

temp [1]

هل 1 < 10 ← shift
هل 1 < 5 ← shift
هل 1 < 4 ← shift

4 5 ? 10 6 2
4 ? 5 10 6 2
? 4 5 10 6 2
1 4 5 10 | 6 2

وصلت index 0 وفش جواه اشي بحط ال 1

sort | unsort

1 4 5 ? 10 2
1 4 5 6 10 | 2

sort | unsort

temp [6]

هل 6 < 5 ← 5 بنطرح ال 6

A = 1 4 5 6 10 | 2

sort | unsort

temp $\boxed{2}$

1	4	5	6	?	10	shift ← 2<10 هل
1	4	5	?	6	10	shift ← 2<6 هل
1	4	?	5	6	10	shift ← 2<5 هل
1	?	4	5	6	10	shift ← 2<4 هل
1	2	4	5	6	10	insert ← 2<1 هل

A = 1 2 4 5 6 10

sorted هي

Algorithim

بدي 2 while ← تمشي من 1 ← n

و مرة من n ← 1

decrement

```
for (i=1; i<n; i++){
    temp = A[i];
    j = i--;
    while (j>=0 && A[j] > temp){
shifting بعد  A[j+1] = A[j];
            j--; }
    A[j+1] = temp;
}
```

Time Complexity ⇒ Worst $O(N^2)$

Best $O(N)$

Shell sort

قائمة على مبدأ ال movement

* تقليل عدد الحركات اللي بعمل فيها shifting.

array ال

A = 23 29 15 19 31 7 9 5 2 | gap = n/2 |

i — No swap — j

23 29 15 19 31 7 9 5 2

i — swap — j

23 7 15 19 13 29 9 5 2

i — swap — j

23 7 9 19 31 29 15 5 2

i — swap — j

23 7 9 5 31 29 15 19 2

i — swap — j

A = 23 7 9 5 2 29 15 19 21

i — j

A = 2 7 9 5 23 29 15 19 31

هيك جاهزة. ⇐ phase 0

هسا ال phase الثانية مبنية على ال gap القديم تقسيم 2

| New gap = Old gap /2 |

gap = 9/2
= 4

بقارن إذا إن ال i أصغر من ال j بعمل swap وبعدين i++ & j++ بطلع Backward & forward

وبتشارطلع Backward لما يتحقق هاد الشرط

| i - gap >= 0 |

i - gap >= 0

4 - 4 = 0

∴ بطلع Backward على ال index 0 هسا بقارن ال i مع Backward.

* Newgap = 4/2 = 2

A = 2 7 9 5 23 29 15 19 31

i — j — No swap

2 7 9 5 23 29 15 19 31

i — j — swap and No need to look Backward

2 5 9 7 23 29 15 19 31

j — i — j — No swap

| 2-2 = 0 |

need to look Backward

~~2 5 9 7~~

2 5 9 7 23 29 15 19 31

look ← i j No need swap

2 5 9 7 23 29 15 19 31

look ← i j swap i and j

2 5 9 7 15 29 23 19 31

look ← i j swap i and j

2 5 9 7 15 19 23 29 31

look ← i 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

i j No swap No look Back word

2 5 9 7 15 19 23 29 31

look ← i j No swap and need to look Back word

2 5 9 7 15 19 23 29 31

look ← i j swap

2 5 7 9 15 19 23 29 31

← i j No swap

2 5 7 9 15 19 23 29 31

← i j No swap

2 5 7 9 15 19 23 29 31

← i j No swap

2 5 7 9 15 19 23 29 31

← i j No swap

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 ;
        }
}
```

هان إذا كانت أ أكل من ب نفذ السوا .

Swap و بنفس الوقت بشوف إذا كان يتطلع Backworld .

Time Complexity ⇒ $O(N^2)$.

Question

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

* عندي Data حجمها كبير والرام عندنا قليلة
وبدي اوتب هاي الداتا استخدم ل ؟

منهان طلعت عندي فكرة

External Sort.

اقسم وبعدين Sort
اي نوع سورت بدي ياه
بس اذا كان عندي اقل من 10 بعدين insertion Sort
واذا كانت الداتا كبيرة استخدم Quick sort.

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

بعدين Divid الـ file

1G	1G	1G	1G	1G

بوخذ كل 1G وبحطه ← Load in to Memory
Sort use quick sort

5G → 1G (F1) → Sort
→ 1G (F2) Sort
→ 1G (F3) Sort
→ 1G (F4) Sort
→ 1G (F5) Sort

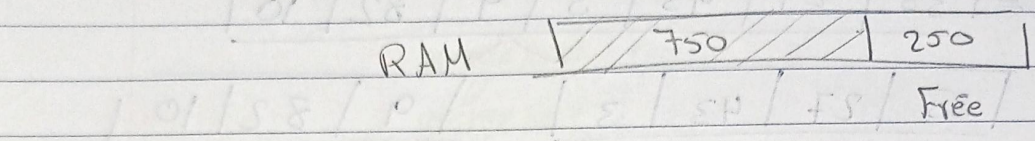
سميناه external
لأنه أنا استفدت بمساحات
خارجية اللي هي H.D
و عبرت اعمل loaded
على الـ Memory.

طيب رتبت كل واحد لحال بس أنا بدي كل الداتا تكون مرتبة !؟

* بوخذ من كل 1G ← 150 MG

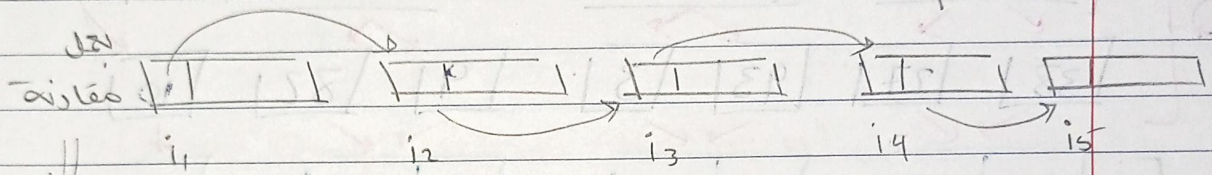
$5 \times 150 = 750$ ميغابايت

وهدول ال 750 بحطهم loaded كل 1G من ال RAM



* وبعدين بدل Merge sort ← K-way Merge sort

بدي اقسم لعدة شغلات



compare ⇒ وبترتيبهوا ال Free 250

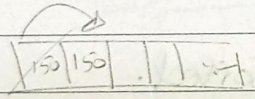
ولما تتعبى ال 250 بطلع OUT 1

وبرد بفضي ال 250 وهكذا.

وفي مرات اشي بخلص قبل مثلاً خلصت i4 قبل i3

فشو بدل ← بدل load another 150 MG

وبضل بطلع عندي out مجموعها 250 M



عدد ال out اللي هي ال Files = $\frac{5 \times 1024}{250}$

هاي الفايلات بتطلع sorted

out 1
out 2
out 3
⋮

Merge file it's 5G sorted.

Merge Sort

فكرته انه بقسم الداتا وبخليها مرتبة
بعدين عند هاي ال 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 |

Conquer

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

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

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

Sorted.

Algorithim

```
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 Complixity $\Rightarrow O(n \log n)$.

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

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

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

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

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

$\vdots$

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

Let

$2^k = n$

$\log_2 2^k = \log n$

$\boxed{k = \log n}$

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

$\boxed{T(n) = n \log n}$