Analysis of Algorithms

Write a code that finds the maximum sum of sub vectors

Exp : 5, 2, -1, ,-3, -5, 6, 4, -1, 2, 5, -3, 1, -4, 5

Algorithm#1:

```
Max = -maxInteger;
 for(i=1; i<=n; i++)
   for( j=1; j<=n; j++){
      sum = 0;
        for(k=i; k<=j; k++)
          sum += A[k];
          if(max < sum)
             max = sum;
   }
```

Time = O(n^3).

Algorithm#2:

```
sum(A[i…j] = sum A[i…j]+ A[j+1];
Max = -maxInteger;
for(i=1; i<=n; i++){
  sum = 0;
  for( j=1; j<=n; j++){
    sum += A[k];
    if(max < sum)
    max = sum;
  }
}
```

Time = O(n^2).

Algorithm#3:

```
A[5, 2, -1, ,-3, -5, 6, 4, -1, 2, 5, -3, 1, -4, 5];
C[5,7,6,3,-2,  4, 8, 7, 9, 14, 11, 12, 8, 13]  ;
 Count[Ci … Ci-1] ➔ sum [Ai … Aj] ;
```

```
c[o]=0;
for(i=1; i<=n; i++)
  c[i] = c[i-1] + A[i];
max =0;
for(i=1; i<=n; i++)
  for( j=1; j<=n; j++){
     sum = sum c[j] – c[i-1];
     if(max < sum)
     max = sum;
}
```

Time = O(n^2).

Algorithm#4:

```
 maxSum (L,U);
if( L > U)
    return 0;
if (L == U)
 return(max(0,A{L}));
 m = (L+U)/2;
  // find max crossing to left
    sum =0;
  for(i=m; i>=L; i--){
   sum += A[i];
      maxToLeft = max(maxToLeft, sum);
    }
//find max crossing to right
sum = 0;
maxToRight =0;
 for(i=m+1; i<=U; i++){
      sum += A[i];
      maxToRight = max(maxToRight, sum);
   }
maxCrossing = maxToLeft + maxToRight;
maxA = maxSum(L,m);
maxB = maxSum(m+1, u);
return (max(maxCrossing, maxA, maxB));
}
```

$$\mathbf{T(n)} = \begin{cases} C & , \ n = 0 \\ 2T\left(\frac{n}{2}\right) + n & , \ n > 1 \end{cases} \quad \rightarrow \quad \mathbf{Time = O(}n \ log n\mathbf{).}$$

Algorithm#5:

```
A[5, 2, -1, ,-3, -5, 6, 4, -1, 2, 5, -3, 1, -4, 5];
sum [5,7,6,3,0,  6, 10, 9, 11, 16, 13, 14, 10, 15] ;
Max [5,7,7,7,7,  7, 10, 10, 11, 16, 16, 16, 16, 16] ;

sum = 0;
max = 0;
  for(i=0; i<=n; i++){
   sum = max(o.sum + A[i]);
   max = max(max , sum);
}
```

Time = O(n).