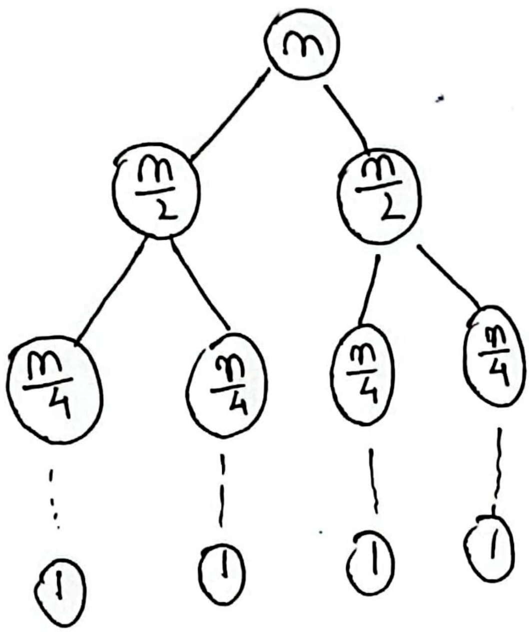
1. We know that,

Brute force Algorithm take $O(n.n) = O(n^2)$ time to find maximum sum subarray problem. It checks all possible subarray combinations. But divide and conquer only consider sub problems.



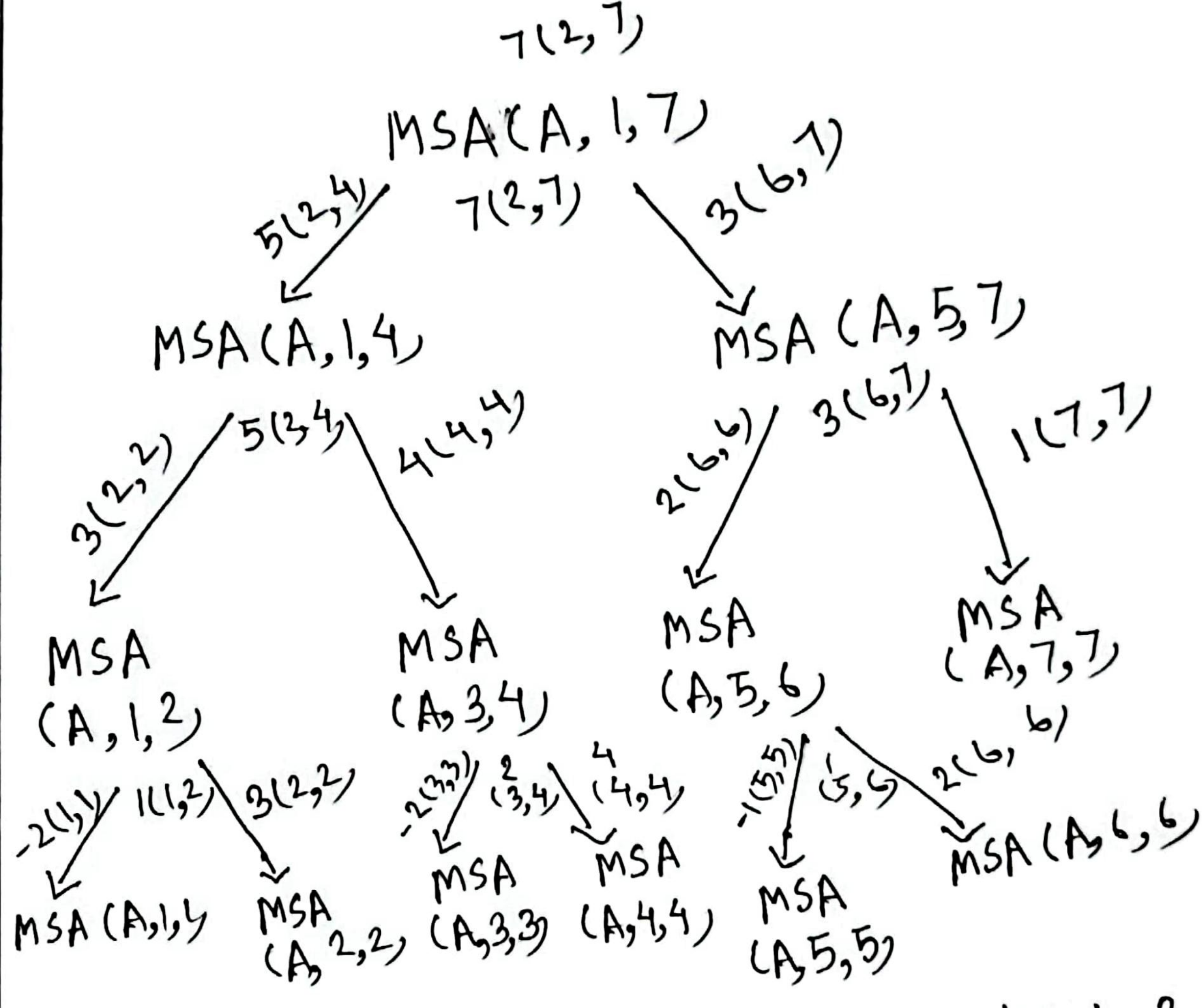
Divide and conquer time complexity $= O(n\log n)$

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

So, Divide and conquer Approach improve the time complexity

2. $A = \{-2, 3, -2, 4, -1, 2, 1\}$

1	2	3	4	5	6	7
-2	3	-2	4	-1	2	1



1	2	1
-1	2	1
5	6	7

3

1	3	-2	2
-2	3	-2	4
1	2	3	4

5

3	5	2	4	-1	1	2
-2	3	-2	4	-1	2	1
1	2	3	4	5	6	7

7

3.

L:

1	2	3	4	5	6
1	5	8	9	10	15

↑ i

R:

1	2	3	4	5	6
4	6	7	11	13	14

j ↑

A:

1	2	3	4	5	6	7	8	9	10	11	12

k ↑

Step-1: L[i] < R[j], A[k] = L[i] i++, k++

A:

1	2										
1											

↑ k

L:

1	5	8	9	10	15

↑ i

Step-2: L[i] > R[j], A[k] = R[j] k++, j++

A:

1	4										

↑ k

R:

4	6	7	11	13	14

↑ j

Step-3: L[i] < R[j], A[k] = L[i] i++, k++

A:

1	4	5									

k ↑

L:

1	5	8	9	10	15

↑ i

Step-4: L[i] > R[j], A[k] = R[j] k++, j++

A:

1	4	5	6								

↑k

R:

4	6	7	11	13	14

↑j

Step-5: L[i] > R[j], A[k] = R[j] k++, j++

A:

1	4	5	6	7							

↑k

R:

4	6	7	11	13	14

↑j

Step-6: L[i] < R[j], A[k] = L[i], k++, i++

A:

1	4	5	6	7	8						

↑k

L:

1	5	8	9	10	15

↑i

Step-7: L[i] < R[j], A[k] = L[i], k++, i++

A:

1	4	5	6	7	8	9					

↑k

L:

1	5	8	9	10	15

↑i

Step-8: L[i] < R[j], A[k] = L[i], k++, i++

A:

1	4	5	6	7	8	9	10				

↑k

L:

1	5	8	9	10	15

Step-9: $L[i] > R[j]$, $A[k] = R[j]$, $k{+}{+}$, $j{+}{+}$

A:

1	4	5	6	7	8	9	10	11			
									↑k		

R:

4	6	7	11	13	14
				↑j	

Step-10: $L[i] > R[j]$, $A[k] = R[j]$, $k{+}{+}$, $j{+}{+}$

A:

1	4	5	6	7	8	9	10	11	13		
										↑k	

R:

4	6	7	11	13	14
					↑j

Step-11: $L[i] > R[j]$, $A[k] = R[j]$ $k{+}{+}$

A:

1	4	5	6	7	8	9	10	11	13	14	
											↑k

R:

4	6	7	11	13	14
					↑j

Step-12: $A[k] = L[i]$

A:

1	4	5	6	7	8	9	10	11	13	14	15
											↑k

4. There are departure times for 8 trains for a railway platform.

Trains	t1	t2	t3	t4	t5	t6	t7	t8
Start	1000	840	850	1700	800	1300	1500	1200
End	1030	1030	1040	2000	835	1800	1650	1380

Now sorting activities by their finish time

	1	2	3	4	5	6	7	8
Trains	t5	t2	t1	t3	t8	t7	t6	t4
Start	800	840	1000	850	1200	1500	1300	1700
End	835	1030	1030	1040	1380	1650	1800	2000

$S(1,8)$ $\{t5, t1, t8, t7, t4\}$

$\downarrow t5$

$S(2,8)$ $\{t1, t8, t7, t4\}$

$\downarrow t1$

$S(5,8)$ $\{t8, t7, t4\}$

$\downarrow t8$

$S(6,8)$ $\{t7, t4\}$

$\downarrow t7$

$S(8,8)$ $\{t4\}$

We at first choose t5. Then we choose t1 as there wasn't 10 min safety break between t2 and t5. Then we choose t8, t7 and t4.

$\therefore$ So the final result is:

$\{t5, t1, t8, t7, t4\}$

[800, 835], [1000, 1030], [1200, 1380], [1500, 1650], [1700, 2000]

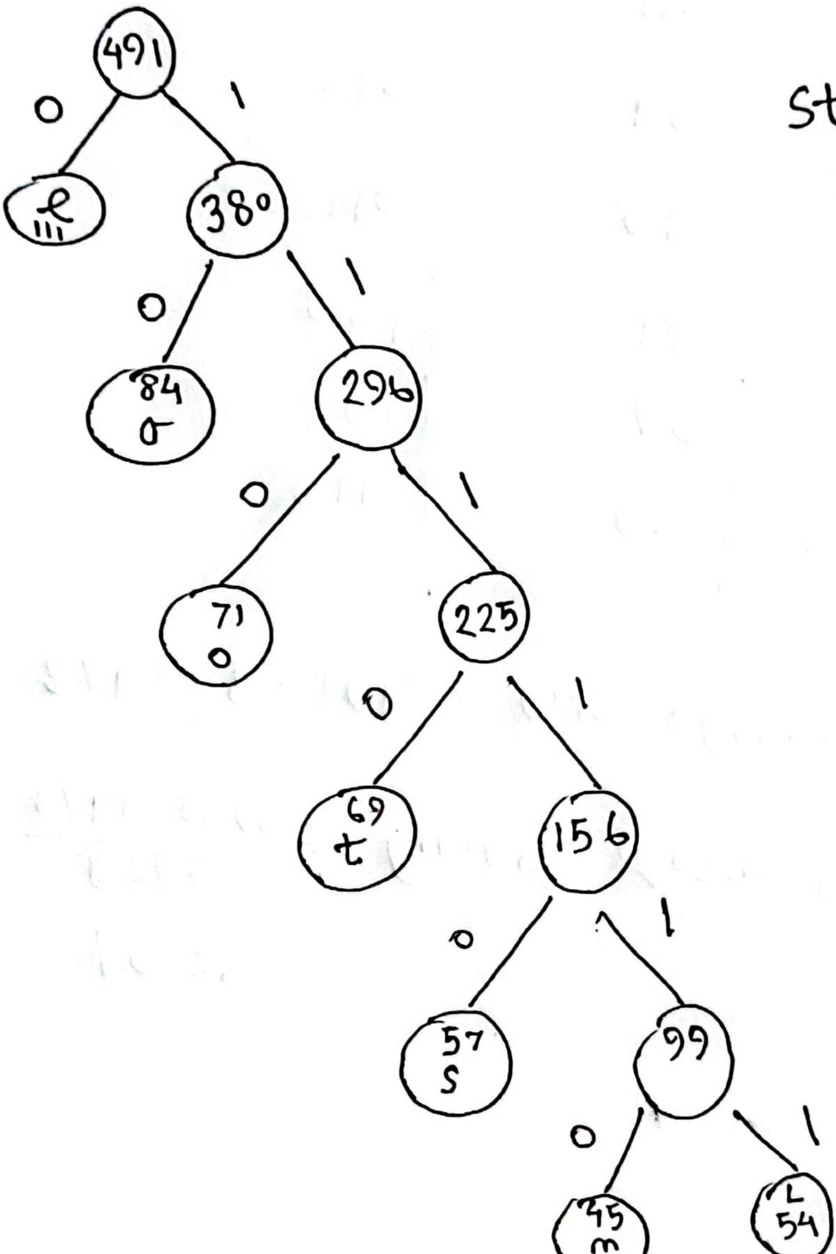
(Ans)

5.

i)

Character	a	e	l	m	o	s	t
Frequency	84	111	54	45	71	57	69

Total Char = 491



Stolen → ~~1111~~0

s → 11110

t → 1110

o → 110

l → 111111

e → 0

m → 111110

ii) Total size = 491×8

$= 3928$

char	frequency	Code
a	84	000
e	111	001
l	54	010
m	45	011
o	71	100
s	57	101
t	69	110

message size $= 491 \times 3 = 1473$

percentage saving $= \dfrac{3928 - 1473}{3928} \times 100\%$

$= 62.5\%$

6. Coins = {1, 2, 3, 4, 7}

Minimum number of coin to make 15

0

0	1
0	1

0	1	1
0	1	2

0	1	1	1
0	1	2	3

0	1	1	1	1
0	1	2	3	4

0	1	1	1	1	2
0	1	2	3	4	5

0	1	1	1	1	2	2
0	1	2	3	4	5	6

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

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

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

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

0	1	1	1	1	2	2	1	2	2	2	2
0	1	2	3	4	5	6	7	8	9	10	11

0	1	1	1	1	2	2	1	2	2	2	2	3
0	1	2	3	4	5	6	7	8	9	10	11	12

0	1	1	1	1	2	2	1	2	2	2	2	3	3
0	1	2	3	4	5	6	7	8	9	10	11	12	13

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

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

Coin ~~used~~

minimum number of coin = 3

coin 7 = 2

coin 1 = 1

7. Let, We have a rod of Length 5. The Prices for cutting rod at different lengths are:

Length:	1	2	3	4	5
Price:	2	6	8	9	10

If we think greedily, we will choose length 5 as it has the highest price 10. But if we sell length 2 and length 3 cut we will get (6+8) = 14 price. Which is greater than greedily choice. So, in this case greedy approach failed to find optimal solution.

8.

Let I am a Internet service provider with bandwith capacity 5. I have 4 customers and their demand bandwith and payment willing are:

	1	2	3	4
Bandwith (Bi)	3	4	2	5
Price: (Pi)	8	10	6	11

We can solve it by 0/1 knapsack

	0	1	2	3	4
0	0	0	0	0	0
1	0	0+0=0 or 0 N	0+0=0 or 0 N	0+0=0 or 0 N	0+0=0 or 0 N
2	0	0+0=0 or 0 N	0+0=0 or 0 N	0+0=0 or 0+6=6 T	0+6=6 or 0 N
3	0	0+0=0 or 8+0=8 T	8+0=8 or 0 N	8+0=8 or 0+6=6 T	8+0=8 or 0 N
4	0	0+0=0 or 0+8=8 T	0+8=8 or 0+10=10 T	0+10=10 or 0+6=6 T	0+10=10 or 0 N
5	0	0+0=0 or 0+8=8 T	0+8=8 or 0+10=10 T	0+10=10 or 8+6=14 T	8+6=14 or 0+11=11 T

∴ We can do maximum profit 14.

Q.9. i)

	1	2	3	4	5
Weight :	3	3	2	3	3
Value :	150	180	170	120	210

w \ i	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0+0=0 or 0 N	0+0=0 or 0 N	0+0=0 or 0 N	0+0=0 or 0 N	0+0=0 or 0 N
2	0	0+0=0 or 0 N	0+0=0 or 0 N	0+0=0 or 0+170=170 T	0+0=0 or 0+170=170 N	0+170=170 or 0 N
3	0	0+0=0 or 0+150=150 T	0+150=150 or 0+180=180 T	0+180=180 or 0+170=170 T	0+180=180 or 0+120=120 T	0+180=180 or 0+210=210 T
4	0	0+0=0 or 0+150=150 T	0+150=150 or 0+180=180 T	0+180=180 or 0+170=170 T	0+180=180 or 0+120=120 T	0+180=180 or 0+210=210 T
5	0	0+0=0 or 0+150=150 T	0+150=150 or 0+180=180 T	180 or 180+170 =350 T	180+170 =350 or 170+120= 290 T	180+170=350 or 170+210=380 T
6	0	0+0=0 or 0+150=150 T	150 or 150+180 =330 T	150+180 =330 or 180+170 =350 T	180+170=350 or 180+120 =300 T	180+170=350 or 180+210=390 T
7	0	0+0=0 or 0+150=150 T	0+150=150 or 150+330= 330 T	150+180 = 330 or 350 T	180+170 =350 or 300 T	180+170=350 or 180+210=390 T

Highest Profit = 390

ii)

object:	1	2	3	4	5
Weight:	3	3	2	3	3
Value :	150	180	170	120	210
P/W :	50	60	85	40	70

objects	Value	Weight	Remaining weight
3	170	2	7-2=5
5	210	3	5-3=2
2	120	2	2-2=0
	500		

∴ Total profit = 500

10.
```
int Catalon(int n)
{
    if (n==0)
        return 1;

    int memo[100];

    if (memo[n]!=0)
        return memo[n];

    int c=0;
    for (int i=0; i<n; i++)
    {
        c += Catalon(i) * Catalon(n-i-1);
    }
    memo[n]=c;
    return c;
}
```

Catalon(3) = 5

11.

Algorithm (n, m)

for (i=1; i<=n; i=i+1) $\longrightarrow n+1$

Print (i) $\longrightarrow n$

for (j=1; j<=m; j=j+1) $\longrightarrow m+1$

for(i=1; i<=n; i=i+2) $\longrightarrow m\left(\frac{n}{2}+1\right)$

print (i*j) $\longrightarrow \frac{mn}{2}$

$T(n) = n+1+n+m+1+\frac{mn}{2}+m+\frac{mn}{2}$

$= 2n+2+2m+mn$

$= mn+2m+2n+2$

$\therefore$ Time complexity $= O(mn)$