D&AA (3001) SOLUTION - AUTUMN END SEM (2016)

Satarupa Mohanty

1\. a// Ans b. I & II

b// Ans c. t1 > t2

c// Steps: 1. Divide the problem into number of subproblems.
2. Conquer subproblems by solving them recursively.
3. Combine the solution of subproblems.

DAC(P)
{ if P is small.
    return (sol(P))
  else
    divide the problem 'P' in small parts
        $P_1, P_2, \ldots P_K;\ K > 1$
    return (combine (sol(P_1), ... sol(P_K)))
}

~~d//~~ $T(n) = \begin{cases} g(n) \\ T(n_1) + T(n_2) + \cdots + T(n_K) + f(n) \end{cases}$

d//

Insertion sort()	O(nlogn)
Heap sort()	O(logn)
Max_Min ()	$O(n^2)$
Binary search()	O(n)

e//

BFS	DFS
1. BFS stands for "Breadth first search"	DFS stands for "Depth first search"
2. BFS starts traversal from the root node and then explore the search in the level by level manner	DFS starts the traversal from root and explore the search as far as possible in a depth wise manner.
3. BFS uses queue	3. DFS uses stack.
4. BFS is slower and requires more memory.	4. DFS is faster and requires less memory.

f// The max-heapify( ) procedure can be applied for a node of the complete binary tree provided that the children node already maintain the max-heap property.

If the max-heapify( ) will be called from 1 to "length (A)/2 then they pre-condition wouldn't be ~~satif~~ satisfied at that time.

g//

Tape 1	Tape 2	Tape 3	Tape 4
40	50	60	70
100	110	120	200
250	400		

The average time of retrieving 10 files from 4 tapes is

~~40 + (40 + (100))~~

$$\frac{(40) + (40+100) + (40+100+250) + (50) + (50+110) + (50+110+400) + (60) + (60+120) + (70) + (70+200)}{10}$$

$$= \frac{1920}{10} = 192$$

h// a) True

b) False

c) True

d) False

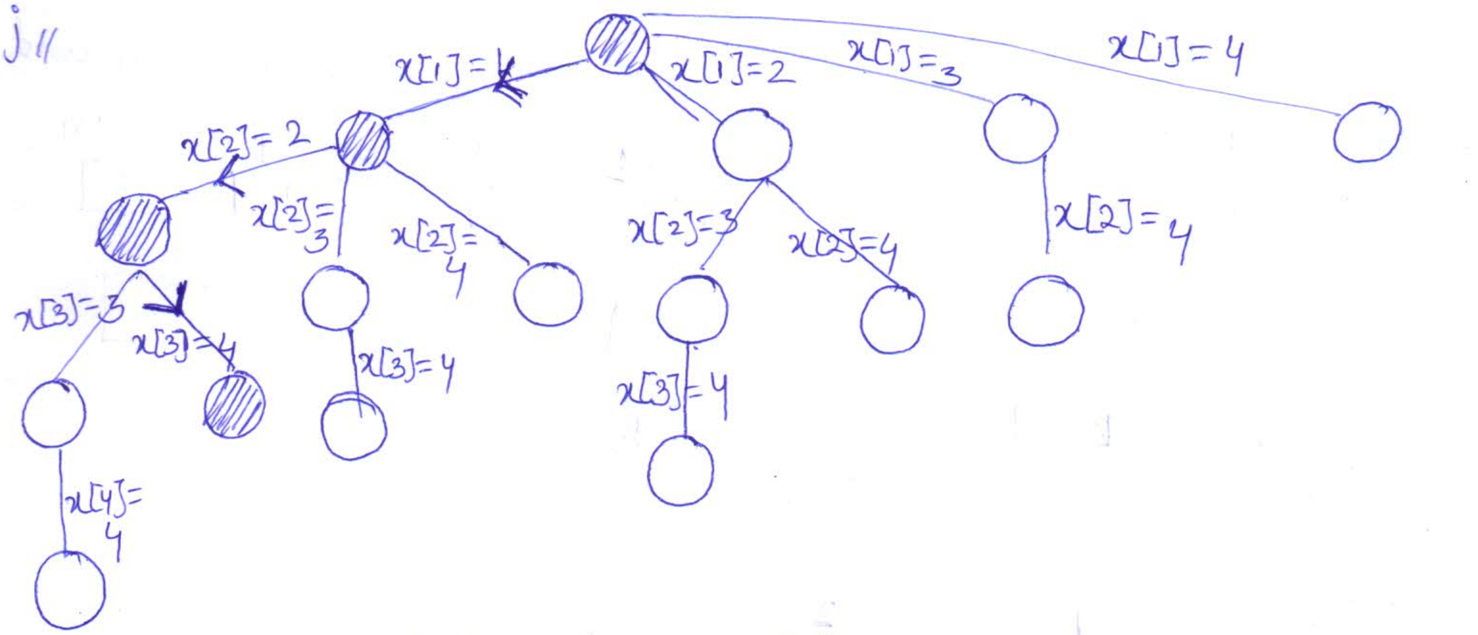
Page-3.

i) A problem is said to satisfy the Principle of Optimality if the subsolutions of an optimal solution of the problem are themselves optimal solutions for their subproblems.

A greedy algorithm is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with a hope of finding a global optimum. In many problems, a greedy strategy doesn't in general produce an optimum solution, however it may yield locally optimal solutions. Greedy alg iteratively makes greedy choice, reducing each given problem into smaller one.

In dynamic programming to solve a given problem it solve different part of the problem (subproblems), then combine the solution of the subproblems to reach an overall solution. Similar to the greedy approach, in every stage of problem solving it makes the choice but unlike greedy the choice is made by looking at the future. It usually described as a recurrence.

j)



Solution = {1 2 4}

Q 2. a// Notation description with graph. [3+1]

b// i// $T(n) = 2T(n/2) + n\log n$ [2

$a=2 \quad b=2 \quad f(n) = n\log n.$

I $\quad h(n) = f(n)/n^{\log_b a} = \dfrac{n\log n}{n^{\log_2 2}} = \dfrac{\cancel{n}\log n}{\cancel{n^1}} = \log n = O(\log n)$

II $\quad u(n) = O\left((\log n)^{1+1}/1+1\right) = O\left((\log n)^2/2\right) = O\left((\log n)^2\right)$

III $\quad T(n) = n^{\log_b a}\{T(\text{base}) + u(n)\}$

$= n^{\log_2 2}\{\text{const} + O((\log n)^2)$

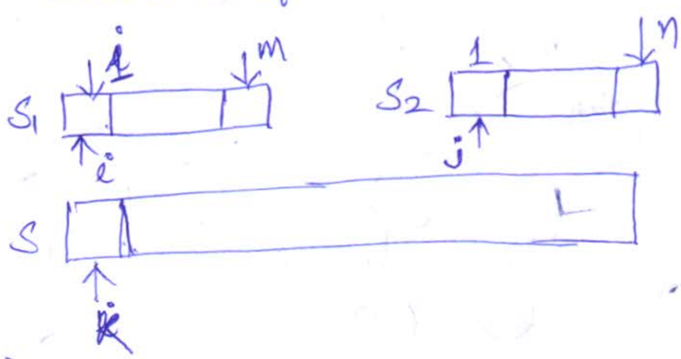
$= O(n\log^2 n)$

iii) $T(n) = 2T(n/3) + T(2n/3) + cn$ [2

Using recursion tree answer is

$T(n) = O(n\log_{3/2} n)$

3/ a// Let S_1 ~~cond~~ and $S2$ contains number of elements as m and n respectively.




```
Union ( S1[],S2[ ], m, n, )
{
    i = j = k = 1
 While((i <= m) && (j <= n)) do
 {
     if( S1[i] < S2[j]) then
     {     S[k] = S1[i]
             k++;
             i++;
     }
```

```
        elseif ( S2[j] < S1[i] ) then
        {       S[K] = S2[j];
                K++;
                j++;
         }
        else
         {      S[K] = S1[i]
                K++;
                i++;
                j++;
          }
   }
   if ( i <= m )
       for t = i to m do
       {    S[K] = S1[t];
               t++;
       }
   else  for t = j to n do
         {   S[K] = S2[t];
               t++;
          }
   ~~for t= 1 to K~~
  }
```

b// Max-min Procedure — 2

Deriving time complexity — 2

4 a// Max_heapify (A, i) procedure — 2

Max_heapify (A, i) procedure — 2

i// When the ith element is greater than its children (As recursive part will not be executed)

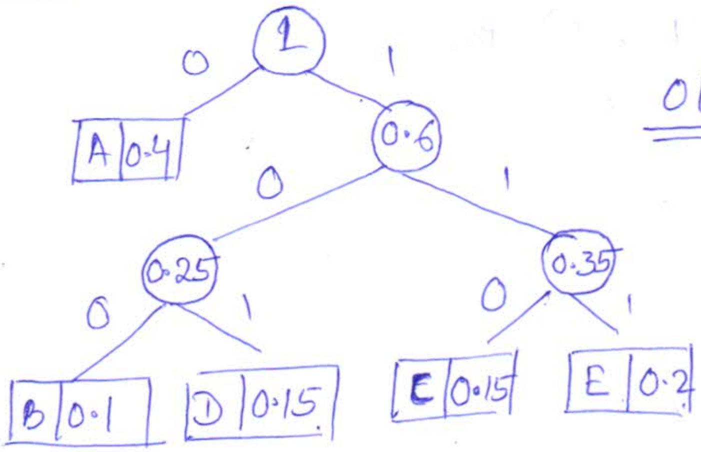
Ans - $T(n) = O(1)$

ii// When i is greater than (Heapsize[A]/2)

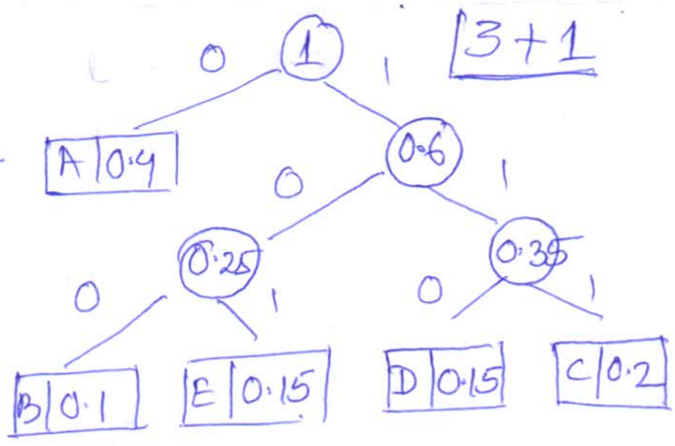
Ans $T(n) = O(1)$ (As there will be no left and right child to compare with)

Page-6

b//



OR



3+1

A = 0
B = 100
C = 110
D = 101
E = 111

BACCDEEA = 10001101101011111110

A = 0
B = 100
C = 111
D = 110
E = 101

BACCDEEA = 10001111111101011010

5/a// 3+1

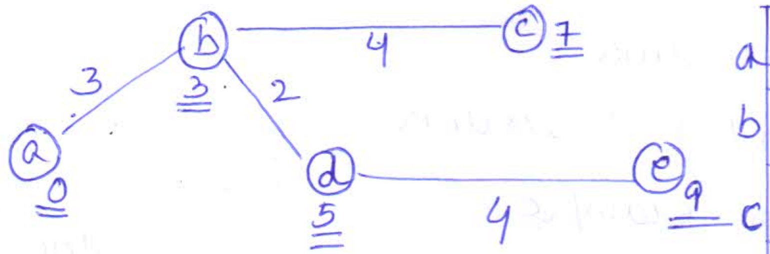
i	Task	Profit	Deadline
1	T3	30	5
2	T9	25	3
3	T7	23	2
4	T2	20	2
5	T4	18	3
6	T5	18	4
7	T8	16	7
8	T1	15	7
9	T6	10	5

Total Profit = 147

1	2	3	4	5	6	7
T7	T2	T9	T5	T3	T8	T1

No all task can't be completed in the scheduled time.
The tasks those will be left out are T4, T6

b/ Shortest Path from 'a' to all other vertex:



	dist[]	Pred[]
a	0	-1
b	3	a
c	7	b
d	5	b
e	9	d

6/a/ $(A) \times ((B \times C) \times D)$ $\underline{1+3}$

$(A) \times (B \times (C \times D))$

$(A \times B) \times (C \times D)$

$((A \times B) \times C) \times (D)$

$(A \times (B \times C)) \times (D)$

	1	2	3	4
1	0	30	66	114
2	x	0	90	130
3	x	x	0	72
4	x	x	x	0

	1	2	3	4
1	x	1	2	3
2	x	x	2	2
3	x	x	x	3
4	x	x	x	x

$((A \times B) \times C) \times D$

b/ $A^K[i,j] = \text{Min}\{A^{K-1}[i,j],\ A^{K-1}[i,K] + A^{K-1}[K,j]\}$

$\underline{1+3}$

A^1:

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

A^2:

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

A^3:

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

A^4:

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

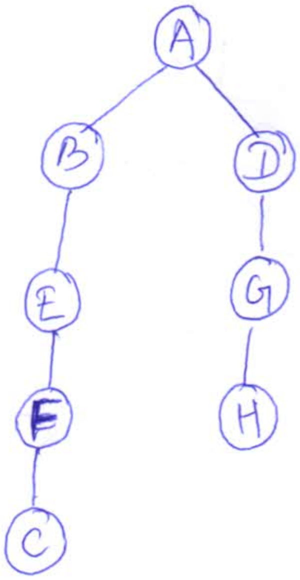
7 a// Union ( ) Procedure [1]

Weighted - union ( ) Procedure [1]

Illustration with example [2]

b// [4]

Node	Start	Finish	Predecessor	Color
A	0	15	-1	~~W~~ ~~G~~ B
B	1	8	~~-1~~ A	~~W~~ ~~G~~ B
C	4	5	~~-1~~ F	~~W~~ ~~G~~ B
D	9	14	~~-1~~ A	~~W~~ ~~G~~ B
E	2	7	~~-1~~ B	~~W~~ ~~G~~ B
F	3	6	~~-1~~ E	~~W~~ ~~G~~ B
G	10	13	~~-1~~ D	~~W~~ ~~G~~ B
H	11	12	~~-1~~ G	~~W~~ ~~G~~ B



DFS tree

8/9/1 $g_0(m) = \max\{g_1(m),\ g_1(m-w_1)+P_1\}$

$g_0(7) = \max\{g_1(6)+1,\ g_1(7)\}$
$= \max\{18+1,\ 24\} = 24$ (1st item not considered)

$g_1(6) = \max\{g_2(4)+6,\ g_2(6)\}$
$= \max\{0+6,\ 18\} = 18$ (2nd item not considered)

$g_2(6) = \max\{g_3(1)+18,\ g_3(6)\}$
$= \max\{0+18,\ 0\} = 18$ (3rd item considered)

$g_2(4) = \max\{g_3(-1)+18,\ g_3(4)\}$
$= \max\{\infty,\ 0\} = 0$ (3rd item not considered)

$g_1(7) = \max\{g_2(5)+6,\ g_2(7)\}$
$= \max\{18+6,\ 18\} = 24$ (2nd item considered)

$g_2(5) = \max\{18+g_3(0),\ g_3(5)\}$
$= \max\{18+0,\ 0\} = 18$ (3rd item considered)

$g_2(7) = \max\{18+g_3(2),\ g_3(7)\}$
$= \max\{18+0,\ 0\} = 18$ (3rd item considered)

x			
	0	1	1

Total Profit = 24

8/b/i N-queen Algorithm [2]

Tracing or drawing space state diagram [2]

—— o ——