(8 Pages) No. 147027

VASAVI COLLEGE OF ENGINEERING

(AUTONOMOUS)

IBRAHIMBAGH, HYDERABAD - 500 031.

Invigilator's Signature and date

Roll No. / Hall ticket No. 1602-21-733-013

MAIN ANSWER BOOK

B.E/M.E/M.Tech......IV......Semester, Branch...CSE.......Mid Term test No...I......

Subject...Design & Analysis of Algorithms... Name...Gayathri Belide...

INSTRUCTIONS :

1) Fill in the Particulars mentioned above before you answer.
2) Write your answers on both the sides of the paper.
3) Write your answers for Part-A questions at one place, next answers for Part-B Questions at one place followed by answers for Part-C Questions

Q. No.	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	Total
Award	1	2	1	1	2	1	3	3	3	3	3	3	29

(1) $f(n) = 3n^2 + 5n - 4$

We know that $f(n) = \Omega(g(n))$ if $f(n) \geq cg(n)$ $\forall n > n_0$

So, $3n^2 + 5n - 4 = \Omega(n^2)$ $\forall n > 0$

for $n = 1$ $f(n) = 4 > (1)^2$

for $n = 3$ $f(n) = 38 > (9)$

So, for any n, $f(n) > g(n)$ so, $f(n) = 3n^2 + 5n - 4 \geq \Omega(g(n))$

(2) $\Rightarrow$ In accounting method amortized ~~value~~ cost for each ~~so~~ case is assigned some value less than average cost, we calcule $P(i)$ for all $i \in (1, n)$ and we show than $P(n) < 0$. Hence amortized cost can not be less than average cost

(3) $f(n) = O(g(n))$ if $f(n) \leq cg(n)$ $\forall n > n_0$

consider $c = 1$, $n_0 = 1$

$f(n) - g(n) = 20n+1 > 0 \quad \forall n > -1$

$f(n) \geq g(n) \quad \forall n > -1$

$\therefore f(n) = n^3 + 20n + 1 = O(n^3)$

(4) Algorithm minmax(a, i, j)

```
{ if (i==j) ~~then~~ max:=min:= a[i] }
  elseif (i=j+1)   ~~max then~~
     if (a[i] < a[j]) then
          min:=a[i]
          max:=a[j]
     else
        min:a[j]
        max:=a[i]
  else
     mid= i+j/2
     minmax(a, i, mid)
     max1=max
     min1 = min
     minmax (a, mid+1, j)
     if(max1 ≥ max) then max := max1
}    if (min1 < min) then min:=min1
```

Approach

1) if there is a single element then it is both min & min

2) if there are 2 elements, min & max are assigned accordingly

3) Else, the problem is divided into 2 halves. And max and min are assigned accordingly

(5) for optimal placement, they should be ascending order

4, 5, 6, 8, 9, 12, 15, 16, 18, 20

$$MRT = \frac{4 + (4+5) + \text{-----} \quad (4+5+\text{---}20)}{10}$$

$$= \frac{4(10) + 5(9) + 6(8) + 8(7) + 9(6) + 12(5) + 15(4) + 16(3) + 18(2) + 20}{10}$$

Mean retrieval time =

2

(6)
```
Algorithm Greedy ( )
{ solution := ϕ
  for (i=0; i<n; i++)
  { a = select();
     if (feasible (solution, a)) then
          solution: = ~~solution~~ Union(solution, a)
  }
} return solution;
```

(8)
```
Algorithm Knapsack(int m,n)
{ // p[1:n], w[1:n] are arranged in decreasing order of Pi/wi

  for(i=0 to n)
     x[i]:=0
  U:= m
  while (U>0 and i<n)
  { if (w[i] < U) then
        U = U - w[i];
  }     x[i]:=1
  if(i<n)     x[i]= U/w[i];
  return x;
}
```

Given $m=28$, $n=7$

$P = \{9, 5, 2, 7, 6, 16, 3\}$ $W = \{2, 5, 6, 11, 1, 9, 1\}$

$\frac{P_1}{W_1} = 4.5$ $\frac{P_2}{W_2} = 1$, $\frac{P_3}{W_3} = 0.33$, $\frac{P_4}{W_4} = \frac{7}{11} = 0.636$ $\frac{P_5}{W_5} = 6$

$\frac{P_6}{W_6} = 1.77$ $\frac{P_7}{W_7} = 3$

Arrangement in decreasing order of P/w

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

$W5 = 1 < m$

$x5 = 1$, $m = m-1$

$m = 27$

$W1 = 2 < 27(m)$

$x1 = 1$, $m = 27-2$

$m = 25$

$W7 = 1 < 25$

$x7 = 1$, $m = 25-1$

$m = 24$

$W6 = 9, < 24$

$x6 = 1$, $m = 24-9$

$m = 15$

$W2 = 5 < 15$

$x2 = 1$, $m = 15-5$

$m = 10$

$W4 = 11 > 10$

$x4 = 10/11$

$x3 = 0$

Solution = $\{1, 1, 0, \frac{10}{11}, 1, 1, 1\}$

profit = ?

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

$a = 4 \quad b = 2 \quad K = 2 \quad P = 0 > -1$

$\log_b a = \log_2 4 = 2$

$\log_b a = K$

$P > -1$, $T(n) = \theta(n^K \log_n^{P+1})$

$T(n) = \theta(n^2 \log n)$

10) ⇒ Job scheduling with deadline completing jobs (or) set of jobs within their given their deadlines so as to gain maximum profit This is solved using greedy approach.

⇒ firstly sort the jobs in decreasing order of their profit

Given 4 jobs

$(P1, P2, P3, P4) = (100, 10, 15, 27)$

Jobs in decreasing of ~~jobs~~ profits = $(100, 27, 15, 10)$ = $\overset{2}{(1},\overset{1}{4},\overset{2}{3},\overset{1}{2})$

Jobs done	Assigned slots	Jobconsidered & deadline	Job ~~considered~~ Status	Profit
{∅}	{∅}	1 (2)	Assigned to [1,2]	0
~~{1}~~ {1}	~~[1,2]~~ [1,2]	4 (1)	Assigned to [0,1]	100
{1,4}	[1,2], [0,1]	3 (2)	reject	127
{1,4}	[1,2], (0,1)	2 (1)	reject	127

Maximum profit = 127

Unfinished jobs = {2, 3}

1 in schedule [1,2]
2 not done
3 not done
4 in schedule [0,1]

11ⓐ Given $f(n) = O(g(n))$

this implies $f(n) \le c_1 g(n)$ — ① $f(n) +$

Given $d(n) = O(h(n))$

this impliest that $d(n) \le c_2 h(n)$ — ②

① + ② ⇒ $f(n) + d(n) \le c_1 g(n) + c_2 h(n)$

Now replace c_1 & c_2 ~~for~~ by some constant K such that $K > c_1$, $K > c_2$

$f(n) + d(n) \le Kg(n) + K(h(n))$

$f(n)+d(n) \leq k(g(n)+h(n))$

$\therefore f(n)+d(n) = O(g(n)+h(n))$

11(b) Algorithm findvowels(a, n)

```
{ S:=[]
   K:=0
   for i in (0,n):
       if (a[i] = 'a' or a[i] = 'e' or a[i] = 'i' or a[i] = 'o' or a[i] = 'u') then
           S[K]:=i          // array of indexes whor vowel is pres
           K=K+1
           a[i]='v'
   return S
}
```

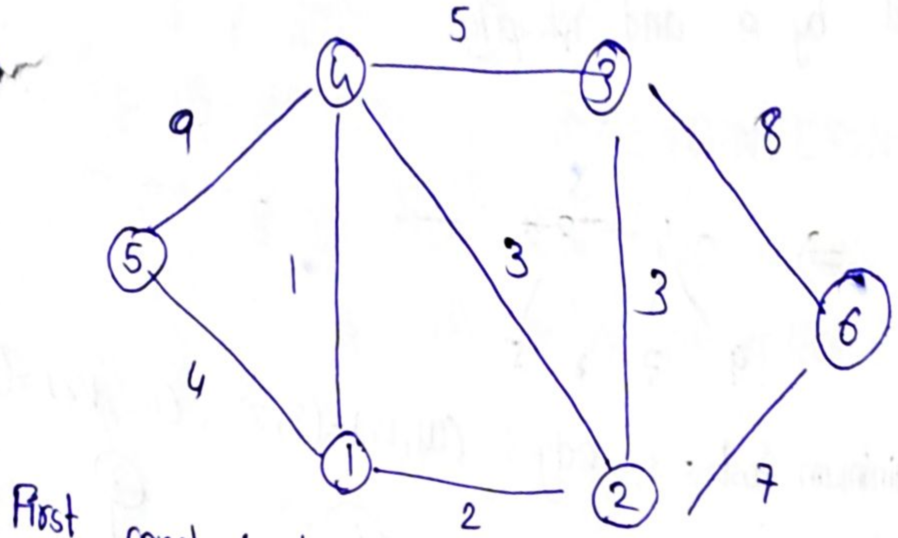
Timecomplexity

Step	count	frequency	Total operations
S:=[]	1	1	1
K:=0	1	1	1
for loop	1	n+1	n+1
if condition	5	n	5n
S[K]:=i	1	n	n
K++	1	n	n
a[i]='v'	1	n	n
return S	1	1	1

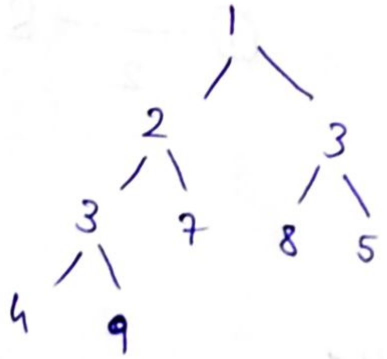
$T(n) = 9n+4$

$T(n) = O(n)$

12(a)

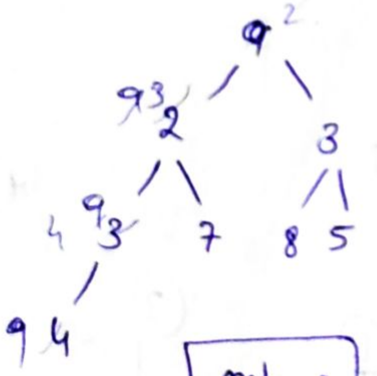


First construct tree with nodes as weights
1, 2, 3, 3, 7, 8, 5, 4, 9

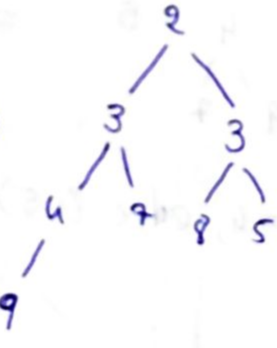


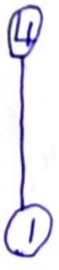
This is a min heap

Select minimum value ie 1, so Select edge (1, 4)
Remove 1, replace by 9 and reheapify



⇒





$\boxed{\text{cost} = 1}$

Select next minimum value = 2, select edge $(\overset{u}{1}, \overset{v}{2})$

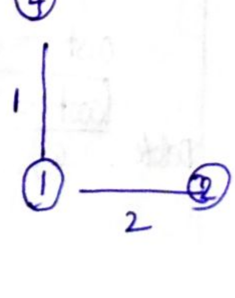
j = find(1) = 4

K = find(2) = 2

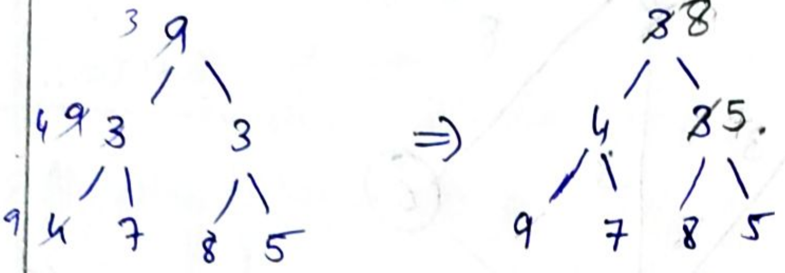
∴ j ≠ K so add edge (1, 2)

cost = 1+2

$\boxed{\text{cost} = 3}$



Delete 2, replaa it by 9 and reheapify



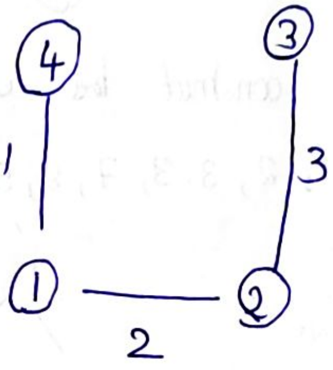
consider next minimum cost =3, edges $(u1,v1)=(3,2)$, $(u2,v2)=(4,2)$

$j1 = find(u1) = 3$

$k1 = find(v1) = 1$

$j1 \neq k1$

so, add (3,2) $\boxed{cost = 3+3 = 6}$



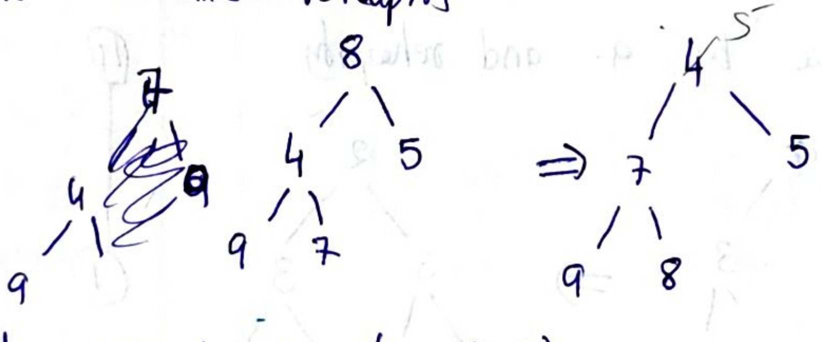
Now consider $(u2,v2) = (4,2)$

$j1 = find(u2) = 1$

$k1 = find(v2) = 1$

$j1 = k1$, so (4,2) cannot be added

Remove 3's and reheapify



consider mincost = 4, edge (1,5)

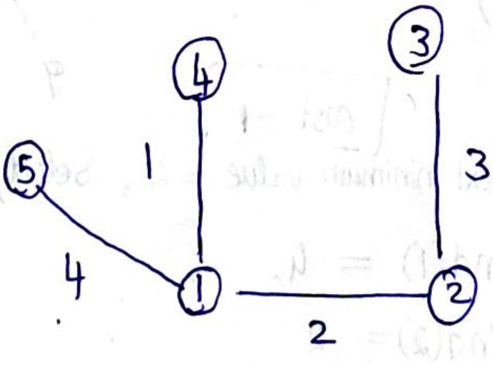
$j = find(1) = 4$

$k = find(5) = 5$

$j \neq k$ ⇒ so add (1,5)

cost = 6+4

$\boxed{cost = 10}$



Delete 4, and reheapify

VASAVI COLLEGE OF ENGINEERING
(AUTONOMOUS)
IBRAHIMBAGH, HYDERABAD - 500 031

(4 Pages)

No. 223118

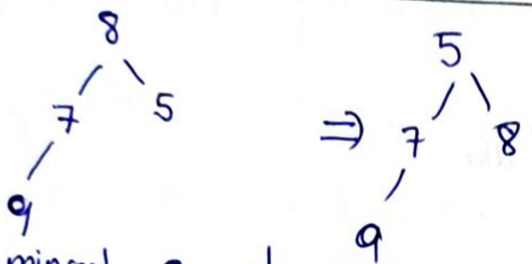
Additional Answer Book

Roll No./ Hall Ticket Number : 1602-21-733-013

Date:_______ Course:_______ Branch:_______ Semester:_______

Subject :_______

Invigilator's Signature with Date



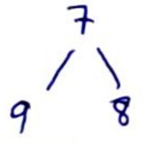
Next mincost = 5, edge = (4,3)

j = find(4) = 1

K = find(3) = 1

j = K ⇒ Don't add (4,3)

Delete 5, and reheapify



Consider mincost = 7, edge = (2,6)

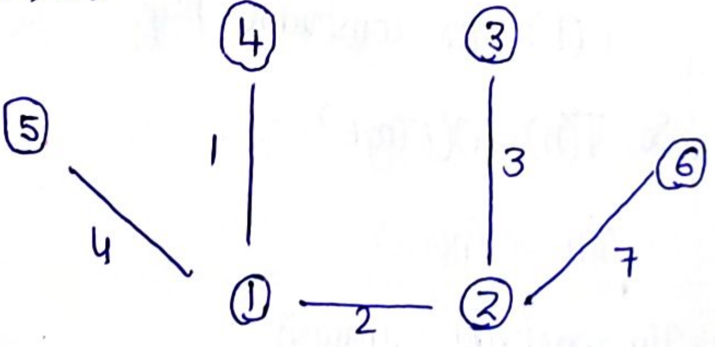
j = find(2) = 1

K = find(6) = 6

j ≠ K , so add (2,6)

cost = 10+7

$\boxed{\text{cost} = 17}$



Final tree

Next mincost = 8, edge (3, 6)

j = find(3) = 2

K = find(6) = 2

j = K , so (3,6) cannot be added

Stop the process as we have got 6 vertices

$\boxed{\text{minimum cost} = 17}$

Algorithm Kruskal (V, E} Cost

{//t [1:n, 1:2] set of edges in final minimum spanning tree

```
while (i<n-1)
{ Delete minimum cost edge from minheap
  Reheapify
  let (u,V) be mincost edge
  if (parent (u) ≠ parent(v)) then
    //add (u,v) to t
      t[i,1] = u
      t[i,2] = v
    mcost := mcost + cost[u,V]
  } union(u,v)
}
```

Time complexity

⇒ O(log E) for ~~constructin~~ selecting minimum cost edge

O(E) for constructing heap

So, $T(n) = O(E \log E)$

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

12(b) Timecomplexity analysis :

Worst case : $(n ~~+~~ + (n-1) + (n-2) + \dots 1$ (when elements are already sorted)

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

$C_w(n) = T(n) = O(n^2)$

Average ~~case~~ time complexity $C_A(n)$:

⇒ In the first partition call number of comparisions (max) Is
$n+1$, and the quicksort is called for two Subarrays. Say
'K' be value returned by partition.

Two subarrays are of sizes $n-K$, $K-1$. On average case
Returned 'j' may be any index, so probability of getting any index $=\frac{1}{n}$

$$C_A(n) = n+1 + \frac{1}{n}\Sigma(C_A(n-K) + C_A(K-1))$$

multiply with n,

$$nC_A(n) = n(n+1) + \sum_{1\le K\le n}(C_A(n-K) + C_A(K-1)) \quad -①$$

Replace n by $n-1$

$$(n-1)C_A(n-1) = (n-1)(n) + \Sigma(C_A(n-K-1) + C_A(K-1)) \quad -②$$

① − ② ⇒

$$nC_A(n) - (n-1)C_A(n-1) = 2n + 2[C_A(0) + C_A(1) + \ldots C_A(n-1)]$$

$$\frac{C_A(n)}{n+1} = \frac{C_A(n-1)}{n} + \frac{2[\cancel{C_A(0) + C_A(1) + \ldots} C_A(n-1)]}{n+1}$$

Repeatedly substituting for $C_A(n-1)$ .. and so on

$$\frac{C_A(n)}{(n+1)} = 2 + \Sigma\frac{2}{K}$$

$$\Sigma \, 1/K < \int 1/K \, dk = \log_e n$$

$$C_A(n) = (n+1)[2 + \log_e n)$$

$$\therefore C_A(n) = \log(_e n)$$

~~55, 25, 48, 32, 65, 72, 28, 36~~

~~↑ pivot ↑ p ↑ p ↑ Q~~

a[i] ≤ v

a[j] ≥ v

~~P<Q, swap P, Q 55, 25, 48, 36, 65, 72, 28, 32~~

~~↑ P → ↑ P ↑ Q~~

~~55 25, 48, 36, 65, 72, 28, 32~~

~~↑ pivot P ↑ P Q~~

~~P<Q, swap (P,Q) 55 25 48 36 65~~

~~55 25 48 32 65 72 28 36~~

~~↑ Pi ↑ i j j i~~

~~55 25 48 32 28 72 65 36~~

~~↑ pivot i j j~~

~~55~~

~~i > j, swap (pivot, j) 28 25 48 32 55 72 65~~

~~j~~

---

55 25 48 32 65 72 28 36

↑ P ↑ i i ↑ j

55 25 48 32 36 72 28 65

i i j j

55 25 48 32 36 28 72 65

i j i

55 25 48 32 36 28 72 65

j i

i>j, swap pivot and j

28 25 48 32 36 |55| 72 65

This partition call returns j = 6,

VASAVI COLLEGE OF ENGINEERING
(AUTONOMOUS)
(4 Pages) IBRAHIMBAGH, HYDERABAD - 500 031

No. 223136

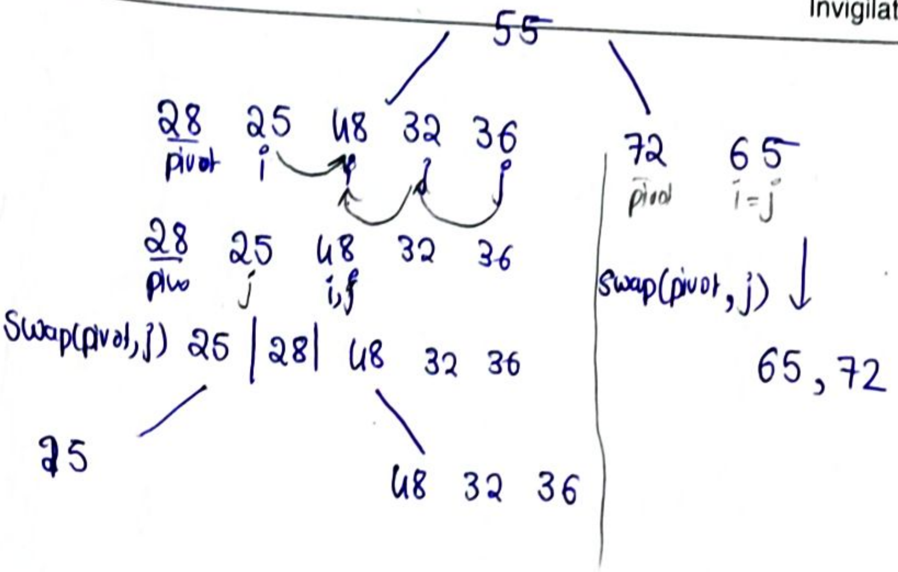
Additional Answer Book

Roll No./ Hall Ticket Number : 1602-21-733-013

Date:______ Course:______ Branch:______ Semester:______

Subject : ______

Invigilator's Signature with Date



⑦ Algorithm minconsumption (a,n)

```
{
    sort (a,n)          // descanding order     } ⇒ sort = nlog n
    miles := 0                                   } 1 ⇒ Assignment
    for i in (0,n):                              } n+1 ⇒ i value
        miles := miles + a[i] * pow(2,i)         } 2*n } multiply & add
    return miles;                                } 1     return
}
```

Timecomplexity

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

$T(n) = n\log n + 3n + 3$

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