Department of Electronics and Electrical Communication Engineering, IIT Kharagpur

EC60074, Design and Analysis of Algorithms

Tadem Sai Pavan
21MM61R13

The Course Solitary Assignment

March 30, 2022

Problem 1

Solution.

a) (Θ)=g

b) f(O)=g

c) f(Θ)=g

d) f(Θ)=g

e) f(Θ)=g

f) f(Θ)=g

g) f(Ω)=g

h) f(Ω)=g

i) f(Ω)=g

j) f(Ω)=g

k) f(Ω)=g

l) f(Θ)=g

m) f(Ω)=g

n) f(Θ)=g

o) f(Ω)=g

p) f(O)=g

q) f(O)=g

Problem 2

Solution.

- lets x, y, z be the single digit numbers in base b.
- Now x, y and z can be atmost $(b-1)$. eg. in base 10 max single digit is 9.
- lets call the 3 single digit summation as Sum, now $Sum = x + y + z$
- $Sum = x + y + z = (b-1) + (b-1) + (b-1) = 3b - 3 = 3(b-1)$

- consider the case of $b = 10$,then the Sum $=3(10-1) = 27$ which is maximum sum for x,y,z maximum single digits (i.e. x=9,y=9,z=9)
- Here it is clear that Sum is 2 digits for max possible base 10 three single digits.
- i.e for any combination of single digit x,y,z the sum is less than b^2

Problem 3

Solution.

a)

$$T(n) = 3T(n/2) + O(n)$$
$$T(n/2) = 3T(n/4) + O(n/2)$$
$$T(n/4) = 3T(n/8) + O(n/4)$$
$$T(n/8) = 3T(n/16) + O(n/8)$$

So the generalized expression is $T(n/2^(k-1)) = 3T(n/2^(k)) + O(n/2^(k-1))$ on solving all above recursive expressions we can get the following expressing
$T(n) = (3^k) * T(n/2^k) + (3^0 + 3^1 + 3^2 + 3^3....3^k) * O(n/2^(k-1))$
but O(n/P) =O(n) for all P
so the expression becomes
$T(n) = (3^k) * T(n/2^(k+1)) + (3^0 + 3^1 + 3^2 + 3^3....3^k) * O(n)$
here $n/2^k = 1$ so k=$log_2 n$

b)

$$T(n) = T(n-1) + O(1)$$
$$T(n-1) = T(n-2) + O(1)$$
$$T(n-2) = T(n-3) + O(1)$$
$$T(n-3) = T(n-4) + O(1)$$

So the generalized expression is $T(n-k) = T(n-(k+1)) + O(1)$
on solving all above recursive expressions we can get the following expressing
$T(n) = T(n-(k+1)) + k * O(1)$
$o(1)$ is for a constant so we can write the time complexity as k*c here k=n-1 so Θ(n)=(n-1)*c => Θ(n)

Problem 4

Solution.

a) T(n) = 5T(n/4) + n By comparing with Master theorem,a=5,b=4,d=1.so $log_b a$ = 1.1609 so d<$log_b a$ so
$T(n) = \Theta(n^l og_b a)$
$T(n) = \Theta(n^1.1609)$

b) $T(n) = 8T(n/2) + n^3$ By comparing with Master theorem,a=8,b=2,d=3.so $log_b a = 3$ so d=$log_b a$ so
$T(n) = \Theta((n^d) * logn)$
$T(n) = \Theta((n^3) * log3)$

c) $T(n) = T(n-1) + n^c$ By comparing with Master theorem,a=8,b=2,d=3.so $log_b a = 3$ so d=$log_b a$ so
$T(n) = \Theta((n^d) * logn)$
$T(n) = \Theta((n^3) * log3)$

d) $T(n) = T(n-1) + n^c$

$$T(n) = T(n-1) + n^c$$
$$T(n-1) = T(n-2) + n^c$$
$$T(n-2) = T(n-3) + n^c$$
$$T(n-3) = T(n-4) + n^c$$

So the generalized expression is $T(n-k) = T(n-(k+1)) + n^c$
on solving all above recursive expressions we can get the following expressing
$T(n) = T(n-(k+1)) + k * n^c$
here $k = n-1$ so Θ(n)=(n-1)*nc => $\Theta(n^{(}c+1))$

e) $T(n) = T(n-1) + n^c$

$$T(n) = T(n-1) + n^c$$
$$T(n-1) = T(n-2) + n^c$$
$$T(n-2) = T(n-3) + n^c$$
$$T(n-3) = T(n-4) + n^c$$

So the generalized expression is $T(n-k) = T(n-(k+1)) + n^c$
on solving all above recursive expressions we can get the following expressing
$T(n) = T(n-(k+1)) + k * n^c$
here $k = n-1$ so Θ(n)=(n-1)*nc => $\Theta(n^{(}c+1))$

f) $T(n) = T(\sqrt{n}) + 1$

The given expression is not in the form of standard. We cannot directly apply the masters theorem,So we need to do some operations to bring into the standard format.

1.Lets assume n=2p

2.So the T(n) becomes $T(2^p) = T(2^(p/2)) + 1$

3.$T(2^p) = S(p), then T(2^(p/2)) = S(p/2)$

4.$S(p) = S(p/2) + 1$, Now, we can easily apply Master's Theorem.

5.On comparing with the masters expression $a = 1, b = 2, d = 0$.So $log_b a$=0 i.e. d=$log_b a$ case.

$S(p) = \Theta((p^d) * logp)$

$S(p) = \Theta((logp)$

6.By using $n = 2^p$ we can get $T(n) = \Theta(loglog_2 n)$

Problem 5

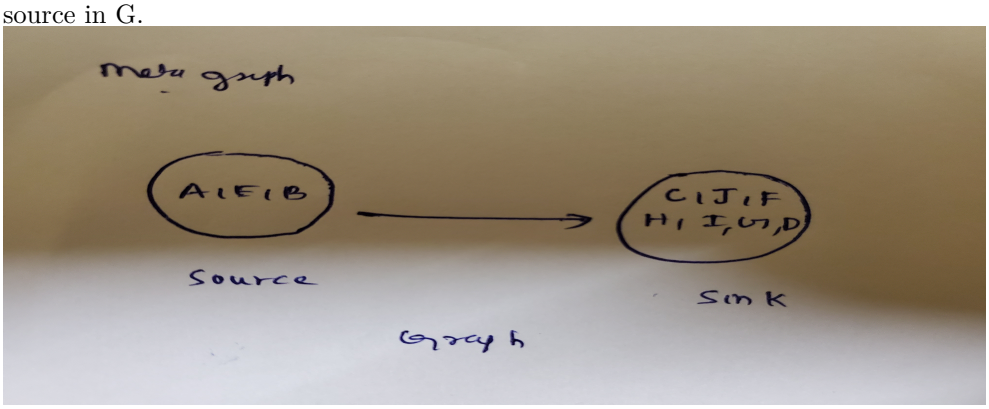
Solution.

- The nth root of unity is ω. $1,\omega^2,\omega^3..\omega^(n-1)$ are nth roots of unity. some of all roots is zero $1 + \ omega + \omega^2 + \omega^3 ... \omega^(n-1) = 0$
- The product of all roots are $1*\omega*\omega^2*\omega^3..\omega^(n-1) = (-1)^(n-1)$
- so if n is odd then the product is positive one i.e. 1
- if n is even then the product is negative i.e -1

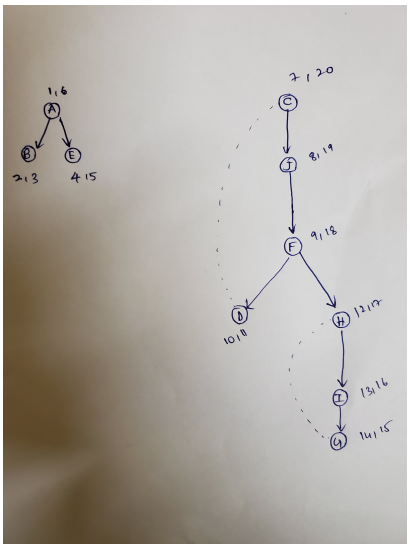
Problem 6

Solution.

- The order of strongly connected components are $[C, J, F, H, I, G, D]$,$[A, E, B]$
- $[C, J, F, H, I, G, D]$ is the source in GR so it is sink in G and $[A, E, B]$ is sink in GR so source in G.



- Minimum of edges are needed to the graph to make it strongly connected.



Problem 7

Solution.

a) Lets consider a simple undirected graph $B - A - C - F$ here total number of edges are 3 and total vertices are 4.
degree of B = 1
degree of A = 2
degree of C = 2
degree of F = 1
sum of degrees$(A + B + C + F) letsD = 1 + 2 + 2 + 1 = 6$ so here D is equals to twice of total number of edges i.e. $2 * E = 2 * 3 = 6$

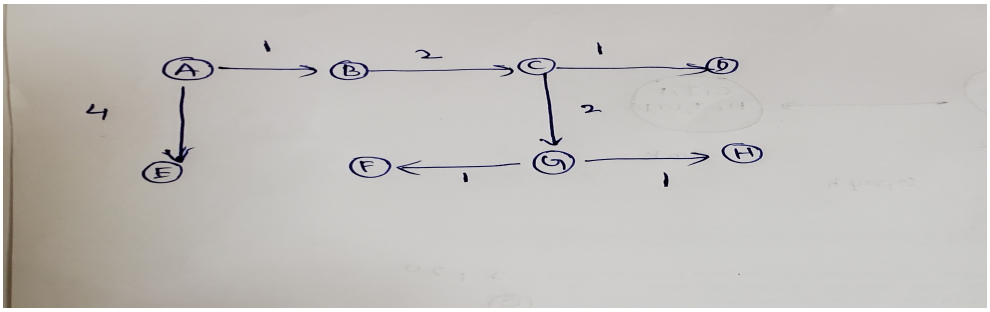
b) The total sum of degrees in an undirected graph must be an even number, if there were an odd number of vertices whose degree were odd, then the total degrees for the graph would be odd which violates the equality stated above.

Problem 8

Solution.

a) Table

Vertices	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11
A	0	0	0	0	0	0	0	0	0	0	0
B	1	1	1	1	1	1	1	1	1	1	1
C	∞	∞	∞	3	3	3	3	3	3	3	3
D	∞	∞	∞	∞	∞	∞	4	4	4	4	4
E	∞	4	4	4	4	4	4	4	4	4	4
F	∞	∞	8	8	8	7	7	7	7	6	6
G	∞	∞	∞	∞	7	7	7	5	5	5	5
H	∞	∞	∞	∞	∞	∞	∞	∞	8	8	8



Problem 9

Solution.

a) The shortestpath between u to v via vo is estimated by shortest path from u to vo and vo to v. The shortest distance from all vertices to vo is same as the shortest distance from vo to all vertices.

b) First apply Dijikstras algorithm on Given Graph with all vertices to vo i.e. Dijikstras(Graph,u)-here we are finding all shortest paths from u to all vertices and store all paths in SP1 .

c) Now by selecting a perticular vo and by reversing the graph we need to find the shortest path by using Dijikstras(Gr,vo) and store all the paths SP2.

d) now for all u,v in Graph:
Create a Matrix with SP1 and SP2 for all u,v

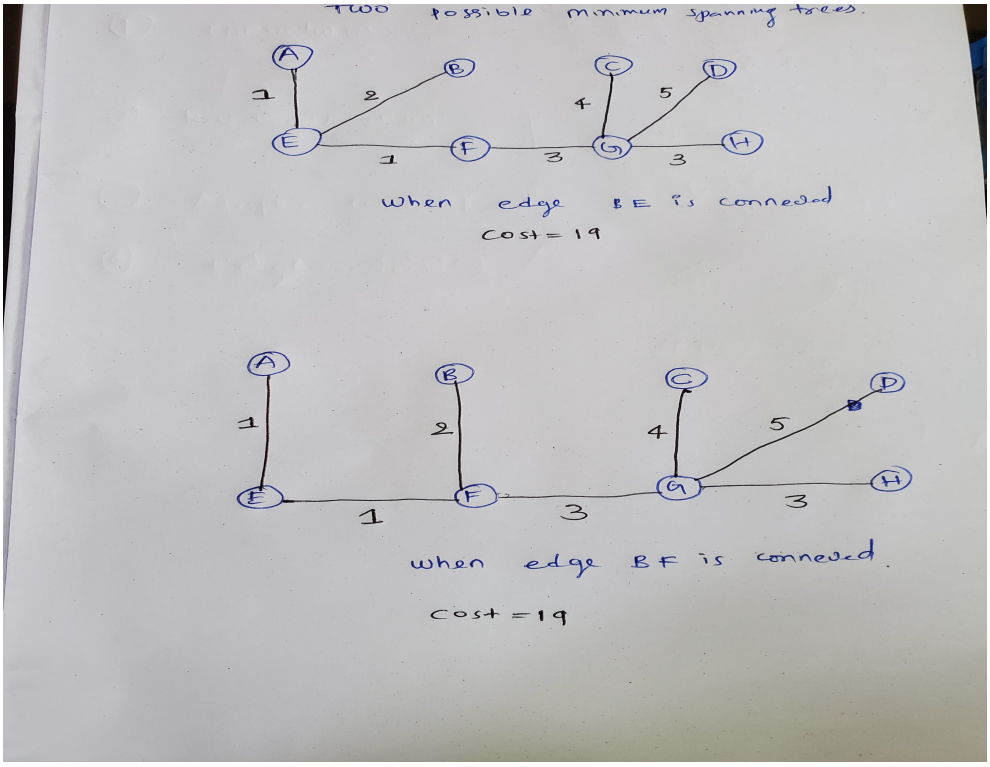
e) return Matrix

Problem 10

Solution.

a) MST cost => 1+1+2+3+3+4+5 = 19

b)



c) AE,EF,BE,FG,GH,CG,GD is the order of the edges added.

Problem 11

Solution.

a) Table of Costs

Vertices	I1	I2	I3	I4	I5	I6	I7	I8
A	0	0	0	0	0	0	0	0
B	∞	1	1	1	1	1	1	1
C	∞	∞	2	2	2	2	2	2
D	∞	8	6	1	1	1	1	1
E	∞	4	4	4	4	4	4	4
F	∞	8	6	6	1	1	1	1
G	∞	∞	6	2	2	2	2	2
H	∞	∞	∞	∞	1	1	1	1

Problem 12

Solution. Lets assume that the graph has 2 MST - MST1 and MST2. Let E be the set of edges present in MST2 but not in MST1.

Consider MST1. If this is a minimum spanning tree, adding an edge to it should create a cycle. Consider adding an edge 'e' from E. Add 'e' to MST1. This would create a cycle. Hence, this new tree (say T) is just 1 edge away from being a MST. To make a MINIMUM spanning tree out of it, you have to remove the most expensive edge in the cycle. Because all the edges have different weights, the most expensive edge will be only one of its kind. If 'e' is the most expensive edge, then, you don't get multiple MST. If 'e' is not the most expensive edge, then MST1 was not a MINIMUM spanning tree.

Problem 13

Solution. The total penalty is the sum of $optimum[j]$ and $(200 - (a_j - a_j)^2)$ i.e cost of the single day trip from j to i.

Optimum[0]=0
for in range= $1, 2, \ldots, N$ **do**
    $optimum[i]$=min{$optimum[j]$+$(200 - (a_j - a_j)^2)$ for j in =0,1,2,3 ....i-}
end for

Problem 14

Solution.

for *iteration* = $1, 2, \ldots$ **do**
    E[i,0]=i
end for
for *actor* = $1, 2, \ldots, N$ **do**
    E[0,j]=j
end for
for *iinrange* = $1, 2, \ldots$ **do**
    for *iinrange* = $1, 2, \ldots$ **do**
        $E[i,j]$=$max\{E[i-1,j-1] + \delta[x(i), y(j)], E[i-1,j] + \delta[\_, y(j)], E[i,j-1] + \delta[x(i), \_]\}$
    end for
end for