Question Solve 2017

1. (a)

Algorithm: An algorithm is a finite set of instruction that is followed to accomplish a particular task.

Basic criteria:

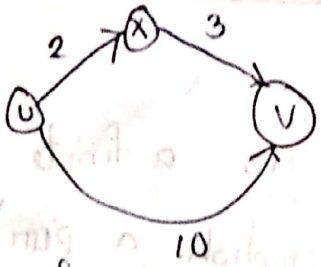
1. Input: Zero or more quantities are externally supplied
2. Output: At least one quantity must be produced.
3. Definiteness: Each instruction is clear and ~~unab~~ unambigious.
4. Finiteness: For all test cases algorithm must terminate within finite number of steps.
5. Effectiveness: Every instruction must be basic so that it is carried out in principle.

1. (b) The properies of DP:

1. Overlapping sub problem: একটা solution বার বার লাগবে। রাখা (divide and conquer এর মত)
তাই ওইটা স্টোর করে

2. Optimal sub structure: A given problem has optimal substructure property if the optimal solution of the problem

can be obtained from its subproblem.



Here distance of u to v goes via X. So it is an example of optimal subproblem where we go from u to X and X to v to get our optimal result.

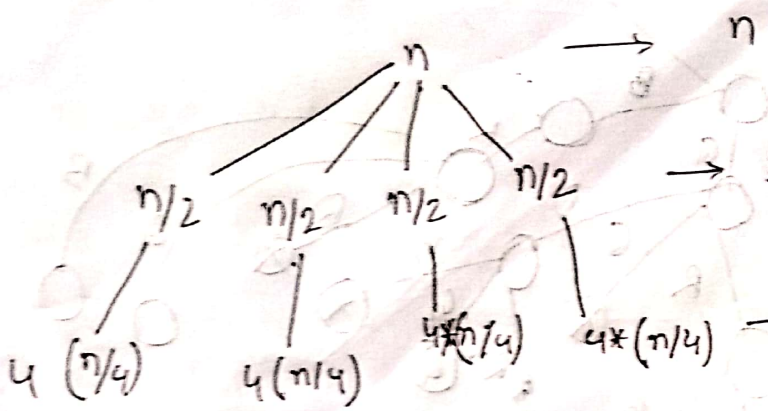
Step of DP:

1. Characterize the structure of optimal sol^n
2. Recursively define the value of an optimal sol^n.
3. Compute the value of an optimal sol^n.
4. Construct optimal sol^n. from the computed info.

1. (c) $T(n) = 4T(n/2) + cn$

```
T(n) ← void test (int n)
{
    if (n > 1)
    {
cn ←|   for (i=0 ; i<n; i++)
    |   {
    |   }
    }
}
```

1(c) $T(n) = 4T(\frac{n}{2}) + cn$



$\rightarrow n$

$\rightarrow 4.(n/2) = 2n$

$\rightarrow 4^2.(n/4) = 4n$

$\rightarrow 4^3(n/8) = 8n.$

$\cdot\ 4^{h-1}(n/2^{h-1}) = 2^{h-1}n.$

$4^h . T(1)$

we know, for

$\frac{n}{2^h} = 1.$

$\therefore n = 2^h.$

$\therefore h = \log n.$

$T(n) = 4^h T(1) + (n + 2n + 4n + \dots 2^{h-1}n)$

$= 4^{\log n} T(1) + n(1 + 2 + 4 + \dots 2^{h-1}).$

now

$4^{\log n} = x.$

$2^{2\log n} = x.$

$\log x = 2\log n$

$\text{or } \log x = \log n^2$

$\therefore x = n^2$

$$\therefore T(n) = n^2 T(1) + n \cdot \frac{1.(2^n - 1)}{2-1}$$

$$= n^2 + n \cdot \frac{(n-1)}{1}$$

$$= n^2 + n^2 - n$$

$$= 2n^2 - n.$$

$$\therefore O(n^2).$$

now with subsitution,

$$T(n) = 4T(n/2) + cn$$

$$= 4\left(4T(n/4) + \frac{cn}{2}\right) + cn$$

$$= 4^2 T(n/4) + 3cn.$$

$$= 4^2 \left\{4T\left(\frac{n}{8}\right) + \frac{cn}{8}\right\} + 3cn.$$

$$= 4^3 T\left(\frac{n}{8}\right) + 7cn.$$

$$\vdots$$

$$= 4^K T\left(\frac{n}{2^K}\right) + (2^K - 1)cn$$

$$\frac{n}{2^K} = 1.$$

$$\therefore \log n = K.$$

$$= 4^{\log n} \cdot T(1) + (2^{\log n} - 1)cn.$$

$$= n^2 + (n-1)cn.$$

$$= n^2 + cn^2 - n$$

$$\therefore O(n^2). \quad \text{[proved]}$$

2. (a)

(i) $1000n^2 + 16n + 2^n < 3.2^n$.

$\therefore O(2^n)$

(ii) $50n + n\log(n^2) + 1000\log(n)$

$= 50n + 2n\log(n) + 1000\log(n)$. $< 10000\, n \log n$.

$\therefore O(n\log(n))$.

(iii) $\log(n) + 10000$

$\log(n) + 100000 < 10003\log(n)$

$\therefore O(\log(n))$

(iv) $2^{20} + 3^7$

This is constant time.

$\therefore O(1)$.

2. (b) Pseudo code:

1. Color the first vertex with first color

2. for next vertex :
   (i) consider the currently picked vertex
   (ii) color it with lowest numbered color that has not used to any adjacent node of it
   (iii) If all colors are used assign a new color

2.(c) Performance analysis is theoritically calculating the time complexity and space complexity of an algorithm. This is ~~any~~ analyzed by the function

Performance measurement is measuring the excute time ~~for~~ of a program for some definite inputs

2. (d)

Brunch and Bound ~~Back tracking~~	Back tracking ~~Brunch and Bound~~
It is used to find the optimal combination of solution	1. Used to find all possible solution.
Finds the solution by dividing the problem in least new restricted subproblem.	Finds the solution by dividing the problem with all possible subproblem and solving them

3. (a) Greedy:

sack capasity = 10.

index	1	2	3
total weigh :	10	3	5
profit :	40	20	30
profit per weight :	4	6.66	6.

So taking object in the sack:

Index	Weight	profit
2	3	20
3	5	30
1	2	8
	10	58

But in 0/1 knapsack problem we can not break an object. So this approach is not correct.

Now Dynamic Programming to ~~sto~~ solve the problem:

P_i	W_i		0	1	2	3	4	5	6	7	8	9	10
	-	0	0	0	0	0	0	0	0	0	0	0	0
20	3	1	0	0	0	20	20	20	20	20	20	20	20
30	5	2	0	0	0	20	20	30	30	30	50	50	50
40	10	3	0	0	0	20	20	30	30	30	50	50	50

So Ans is 50 and object is (2,3)

With Branch and Bound:

with set method:

$S^0 = \{(0,0)\}$

$\{(10, 40)\}$ ← 1st object added

$S^1 = \{(0,0), (10,40)\}$

$\{(3, 20), (13, 60)\}$ ← 2nd object added (out of bound)

$S^2 = \{(0,0) (10, 40) (3,20)\}$

$\{(5,30), (15, 70), (8, 60)\}$ ← 3rd object (out of bound)

$S^3 = \{(0,0), (10,40) (3,20) (5,30) (8,60)\}$

weight can't be in decreasing order.

∴ (8,60) is in S^3

So, 3rd object is must.

remaining weight and profit (8-5, 60-30)

(3, 20)

(3,20) is in S^2

So, 2nd object is must

∴ object (2,3) is added.

with branch and bound:

Least cost tree

①	②	③
10	3	5
40	20	30

$m = 10$

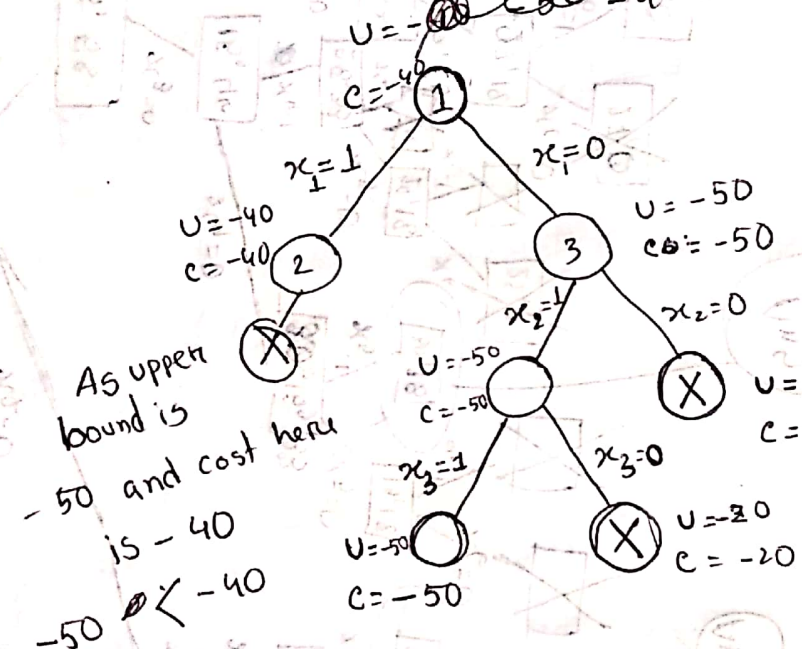
Starting upper bound $U = -40$

$c = -40$

-40

Upper $= -40, -50$

Rough
cost = 50
U = -50



As upper bound is -50 and cost here is -40

$-50 < -40$

$-50 < -30$

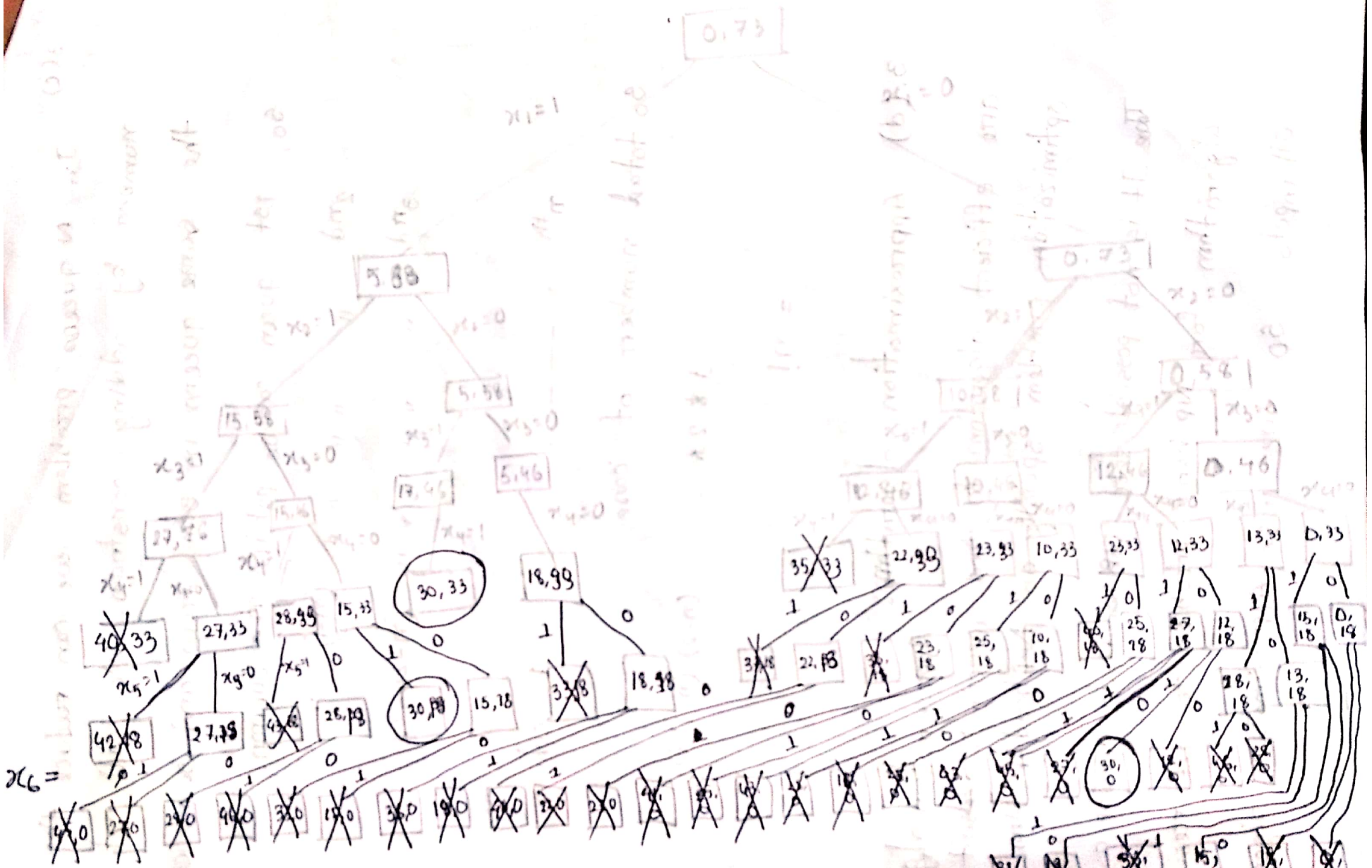
∴ Ans: { 0 1 1 }

Only object {2, 3} is included.

3.(b) Sum of subset time complexity $= 2^n$

$n = 30$

$\{5, 10, 12, 13, 15, 18\}$ Total = 73



So Ans: {1011001} {0010011}
{1100101}

3(c) In N queens problem we can reduce the case number by adding constrain. We will not put the ~~same~~ queens in same row and same column.

So, 1st queen can be put in n column.

2nd " " " " " (n-1) "

3rd " " " " " (n-2) "

⋮

nth ——————— 1 column

So totall number of case,

$1 * 2 * \ldots\ldots (n-2)(n-1)(n)$

$= n!$

3.(d) Approximation algorithm:

Approximation algorithm are efficient algorithm to find approximate result of optimization problem specially on N-phard problem.

~~There~~ It is not possible ~~for~~ ~~NP~~ hard to develop a generalize algorithm for NP hard problem to get result for all inputs. So we use Approximation algorithm

to get a result which is near to the solution.
For these kinds of problem we use approximation algorithm.

4. (a) Divide and conquer feature:

It divides the big problem into some small problem then try to solve those small problem. If those problem is also too big then those are also so cuts into small problems. Then Recursively this process will continue until the program can solve the problem. Then it will combine the subsolution and make the real solution.

Time complexity is calculated with

$$T(n) = a\,T(n/b) + f(n)$$

Here $f(n)$ is time for other parts of algorithm/function

a is number of recursive calls.

b is size of subproblems.

4.(b) Greedy Choice property:

We can make a decision based on the previous result and then solve the sub-problems arises later. Greedy algo. does not depend on future subproblem or future choices.

Greedy choice Property: A optimal solution can be reached by chosing optimal choice at each steps.

Characteristics of greedy optimization:

1. Greedy choice property
2. Optimal substructure: If an optimal solution to the total problem contains the optimal solution to the sub problems.

So greedy will chose an optimal solution in each stage and thus the problem would be solved.

int ppw vector <int> profit, weight ← weight and profit array/vector

vector ppw a[n]

for (i=0; i< n; i++)

{ a[i].index = i;

a[i].profit = profit[i] / weight[i];

}

~~while (sack != full)~~ m = sack capasity

{

~~if while(~~

sort (a, a+n, cmp)

while (~~ea taken object~~ total weigth ≤ m)

{ i=0;

totalweight = ~~a~~ weight[a[i].index]

total profit = profit [a[i].profit]

i++.

}.

5. (a) pre condition: Pre conditions are such condition which needed to be fulfilled to use an algorithm.

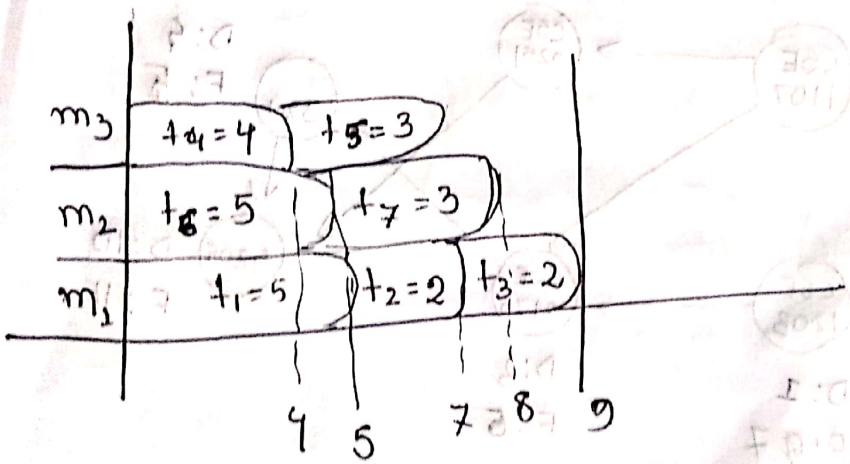
pre condition of Bellman ford :

1. The should not be any negitive weighted cycle in the graph.

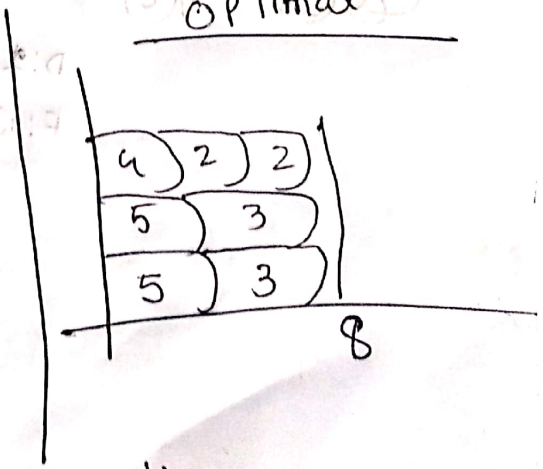
Belmen ford is a single source shortest path algorithm If there is a negitive weighted cycle it would not be possible to find shortest path.

5. (b) So we sort the task with their processing time:

$$(5, 5, 4, 3, 3, 2, 2) = (t_1, t_6, t_4, t_5, t_7, t_2, t_3).$$



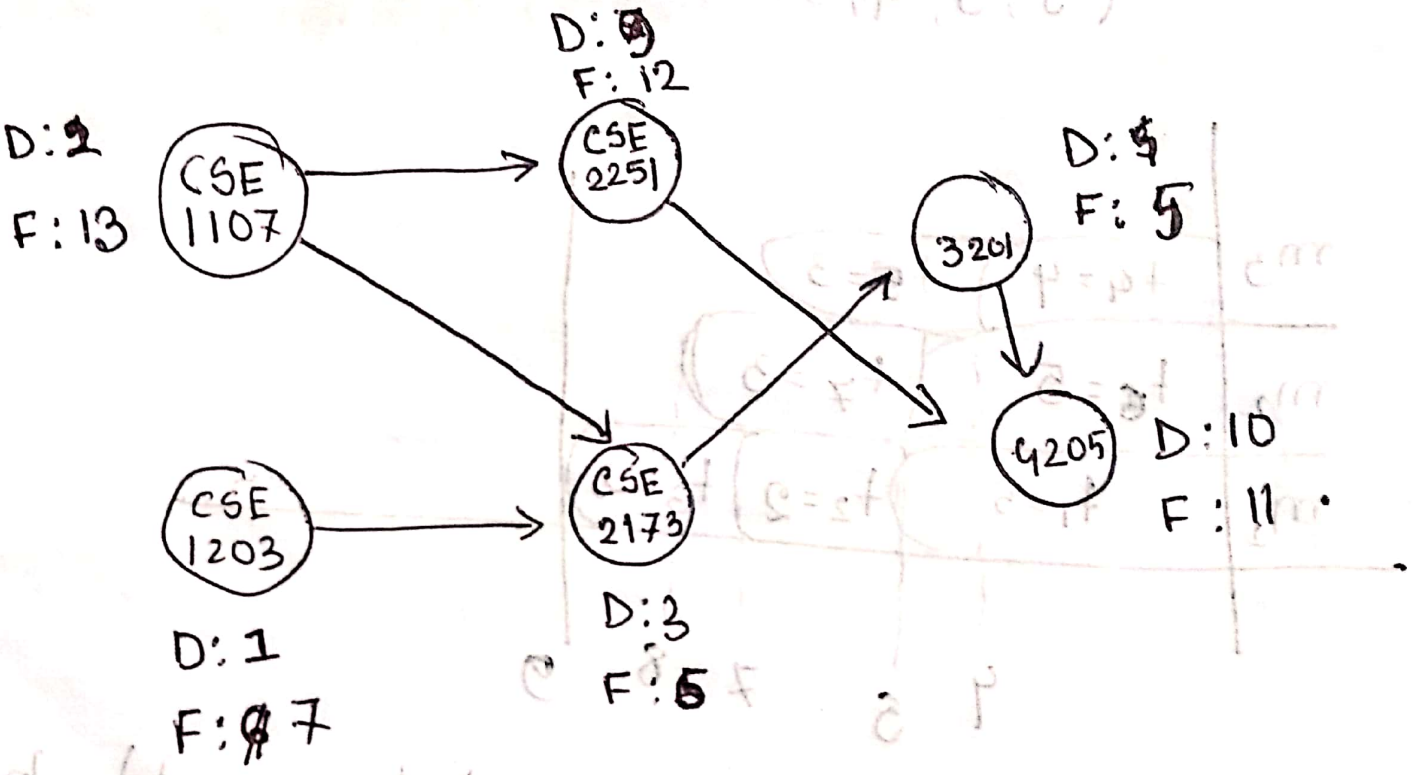
optimal



optimized schedule would be the same.

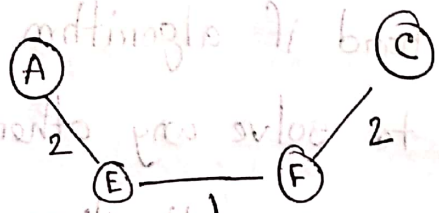
(B) It would form a cycle.

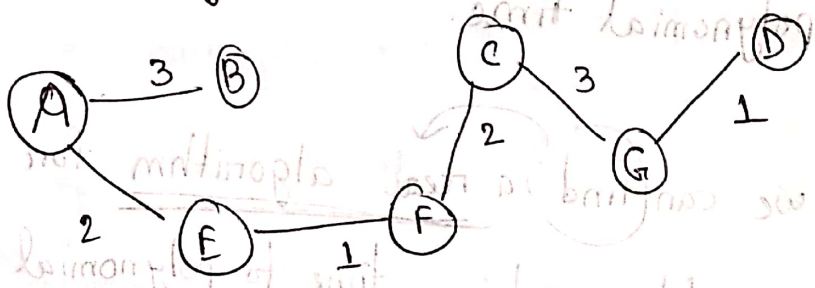
4(c)



6. (a) Spanning Tree:

Spanning tree is a representation of graph where all the n nodes are connected with edges and there are no cycles.





(i) Distance between $A \rightarrow D = 9$

(ii) Total length = 12

6. (b)(c) Least cost search is such a searching technique where we select lowest weighted ~~egd~~ edge to go on next ~~desi~~ stage

7. (b) NP means Nondeterministic polynomial. And P means Deterministic polynomial. when we find a real algorithm that can work on polynomial time that would be 'P' algorithm or 'P' problem. If we still did not find the ~~algorithm~~ but formula but we are assuming and working on the formula that is NP.

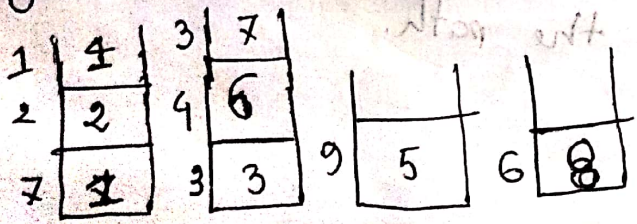
So all the P problem was NP before it was invented

So we can say the P is a subset of NP

7. (c) ① first fit:

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

B = 10



If we start selecting the node that has fewest ~~nub~~ number of arc, LC ~~true~~ method would convert into bfs.

~~And if~~.

7. (a) A problem is NP hard if algorithm for solving this problem can be used to solve any other NP problem and the algorithm will reduce the time for solving into a polynomial time.

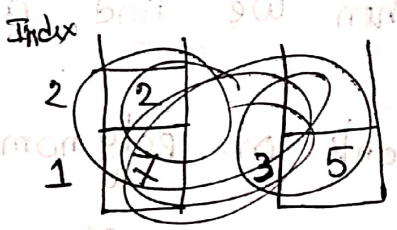
If we can find a non deterministic algorithm ~~reat~~ algorithm for ~~reducting~~ reducing the problem solving time to polynomial time, this is known as NP complete problem.

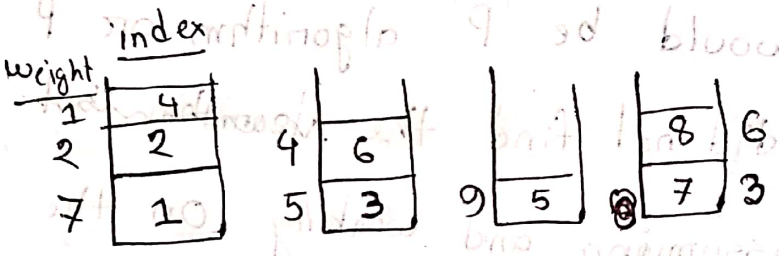
NP hard Example: Halting Problem, Vanlex covelt

NP complete: Determine whether a Boolean formula is stisfiable or not.

Best fit:

index	1	2	3	4	5	6	7	8
weight	7	2	5	1	9	4	3	6



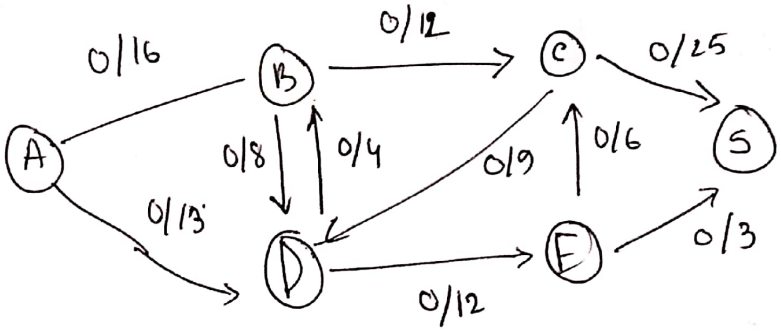


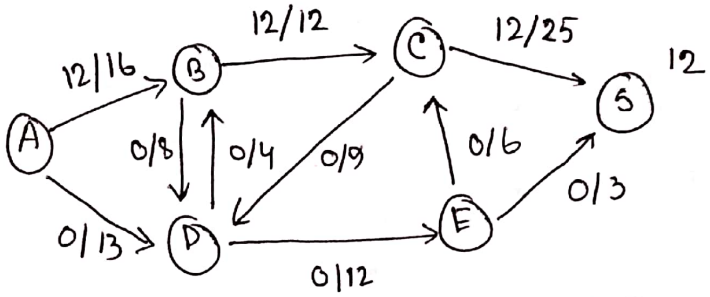
8. (a) If we define starting node as source and ending node as sink, the maximum flow that can be sent through the vertex ~~is~~ from source to sink is at called max flow.

(b) Edmond ~~Gof~~ Karp use BFS to find a path from source to sink

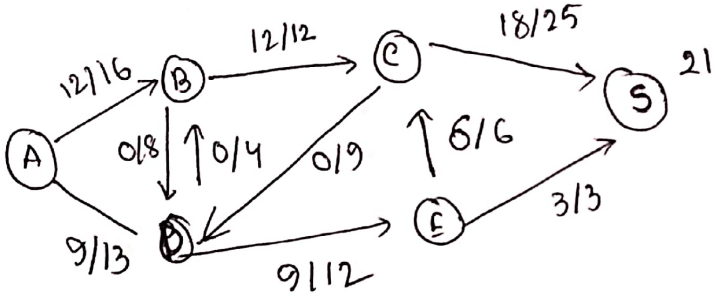
where in Ford-Fulkerson there is no definite algorithm to find the path.

8.(b)





A → B → C → S



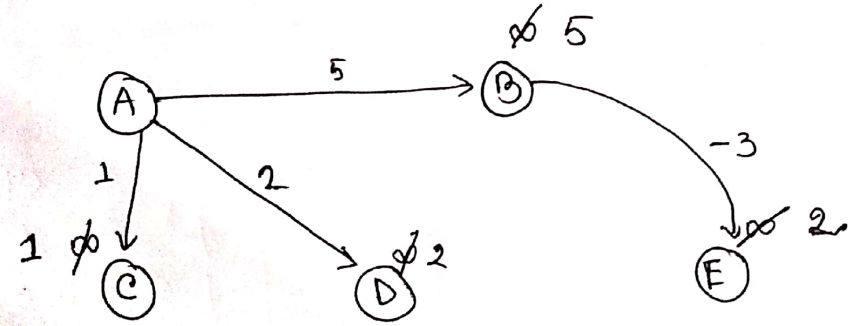
A → D → E → S

A → D → E → C → S

So max flow = 21

min cut = 21

8(c).



So this is the final graph.