D. 2018

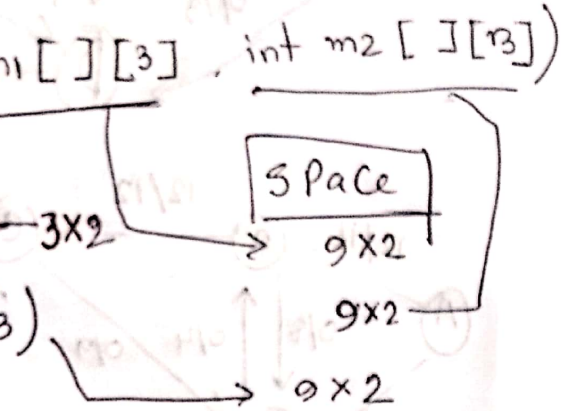
1. (a) 2017 (a)

(b)

```
int **matrix_sum (int m1[][3], int m2[][3])
```

Time

```
1 —— int i, j, **m3 ;
1 —— m3 = malloc (sizeof (int*) * 3)
3 —— for (i=0, i<3, i++)
3 ——  {
          for (j=0, j<3, j++)
1 ——        m3[i][j] = m1[i][j] + m2[i][j]
          }
1 —— return m3 .
```



∴ Time = 3+

∴ Space = (9X2) X 3 + 3X2 unit

= 60 unit.

1.(c) (i) $\log(n!)$

$= \log(1 * 2 * 3 * 4 \ldots n)$

$= \log 1 + \log 2 + \log 3 + \ldots \log n.$

$\log n! \leq n \log n.$

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

(ii) $6 \cdot 2^n + n^2$

$6 \cdot 2^n + n^2 \leq 10 \cdot 2^n$

$\therefore O(2^n).$

(iii) $1000n^2 + 100n - 6$

$1000n^2 + 100n - 6 \leq 3000 \cdot n^2$

$\therefore O(n^2).$

(iv) $\frac{6n^3}{\log n + 1} \leq 6n^3.$

$\therefore O(n^3).$

2. (a) $T(n) = 2T(n/2) + cn$

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

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

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

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

$\vdots$

$= 2^K T\left(\frac{n}{2^k}\right) + K \cdot cn.$

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

$n = 2^K.$

$\therefore K = \log n.$

$\therefore T(n) = nT(1) + \log n \cdot cn.$

$= a.n + c(n \log n)$

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

2.(b)

Brute force	Back tracking
Brute force is straight forwand. It takes all possible values to solve the problem in variable and then checks the constrains	Back tracking checks one value or one possibility and it, it does not satisfies, it immidietly goes back previous stage and assign a new value to variable

2.(c) Constrains means limitation to something. In algorithm constrains gives boundary to a problem.

Constrains are useful in backtracking method to solve the problem efficiently. If the stage does not satisfy the constrain it terminates the stage Thus it reduces the number of stages and finds the optimal soln.

constrain for n queens:

No queen can be placed in same row
" " " " " column
" " " " " diagonal.

```
place (k,i)
{ for j ← 1 to k-1
    do if (x[j] = i) or (Abs(x[j]-i) = (Abs(j-k)))
            return false
  return true;
}

N-Queens (k, N)
for i ← 1 to no
   if (place (k,i))
   { x[k] ← i
   if (k==n)
       write (x[1,2,...n]);
   else
      N-Queens (k+1, n);
   }
```

③ → Ans

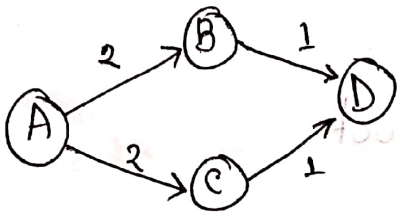
4. (a) 17 3(d) .

4. (b) 7 (a).

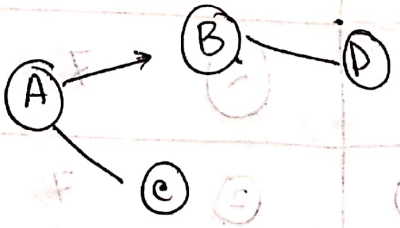
4. (d) MST using prims and Kruskal is not ~~no~~ unique.

Suppose there is two same weigthed edge. ~~Prims~~

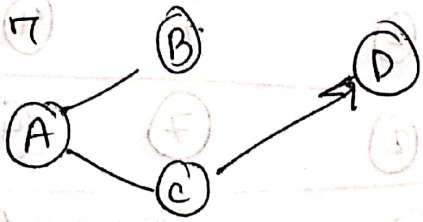
Algorithm can chose any of them.



So the graph may be



or

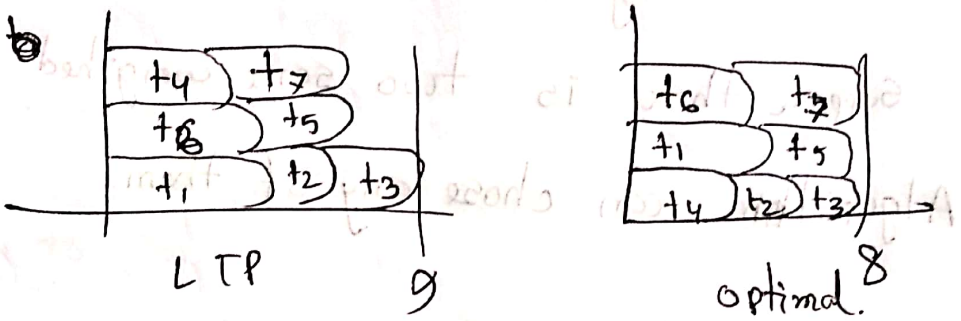


So this are not unique.

5. (a) List $(V+E)$
Matrix (V^2).

5(b): bfs. Because bfs search level wise

5(c):

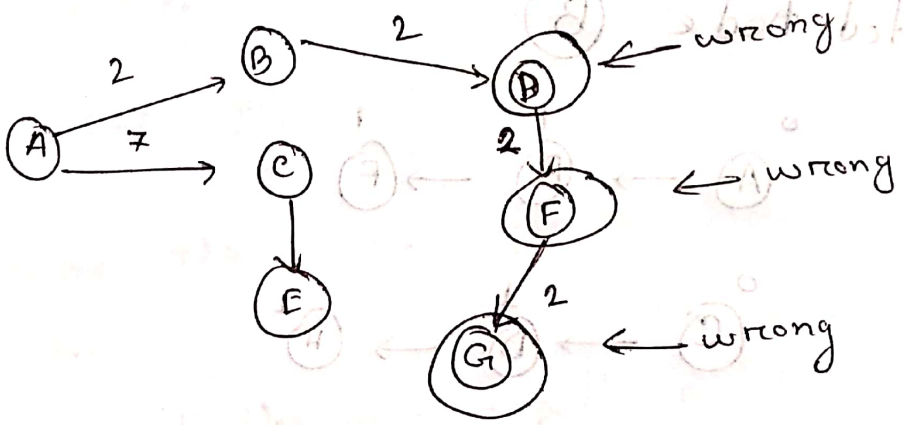


6. (a) (A) is root.

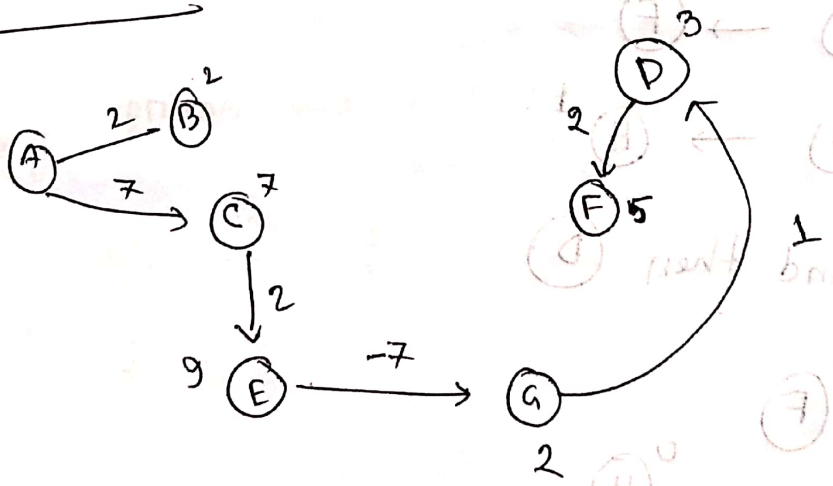
selected	(B)	(C)	(D)	(E)	(F)	(G)
(B)	(2)	7	∞	12	∞	∞
(D)	(2)	7	(4)	12	∞	∞
F	(2)	7	(4)	12	(6)	∞
C	(2)	(7)	(4)	12	(6)	8
G	(2)	(7)	(4)	9	(6)	(8)
E	(2)	(7)	(4)	(9)	(6)	(8)

(II) if we removed E -7 G edge the graph will work correctly

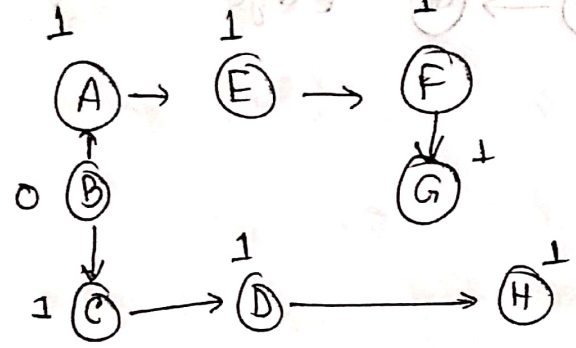
(III) Wrong paths:



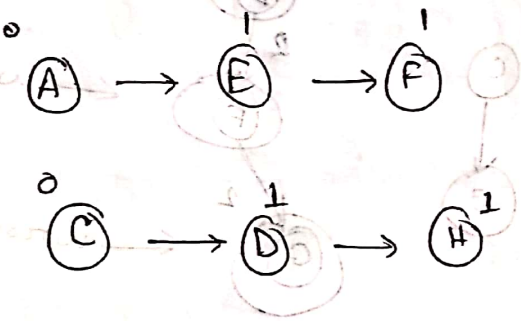
Correct path:



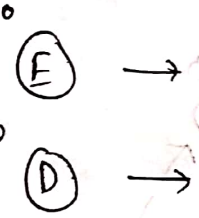
6(b)



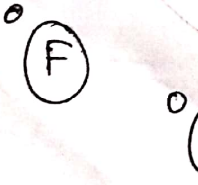
selected node (B)



(A) & (C)



(E) and then (D)



(F) and then (H)

$\therefore$ (B) (A) (C) (E) (D) (F) (H).

2. B C A E D F H

3. ~~B C A B D H~~ B C D H A E F G.

4. B A E F G C D H.

6.(c) Ford fulkarson:

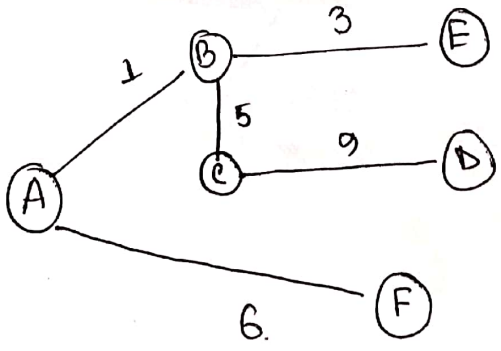
(i) find a path from source to sink using any method.

(ii) Then send the maximum possible flow through that path.

(iii) Again ~~do~~ find ~~the~~ another path and send flow.

(iv) This will continue untill there is no path remaining

7.(a) we apply prims without considering x,



if $x > 9$ it would be ignored

if $x \leq 9$ then it can be added in the graph

7.(b)

A → H → F → A

A → H → F → B → A

G → J → E → G

bipartite graph

8.(b) 2017 8(b)

(a)