Design & Analysis Of Algorithm

BASIL ALI KHAN
20K-0477

Question # 01

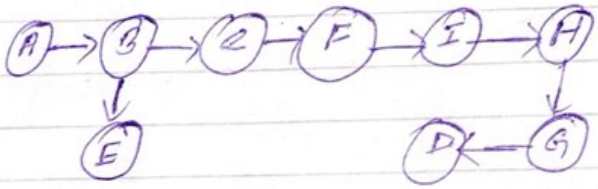
(a)

①

visited order : D G H I F C

②

visited order : D G H I F C E B A



(b)

Back edges :
DH, HF, E, A.

Forward edges :
EH

Cross edges :
AD, HI.

Question #02:

(a)

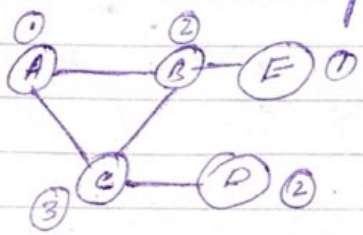
```
bool color (int G[][4], int color[], int position, int x) {
    if (color[position] != -1 && color[position] != x) {
        return false;
    }
    color[position] = x;
    bool flag = true
    for (i=0 ; i<4 ; i++) {
        if (G[position][i] == true) {
            if (color[i] == -1) {
                val = color(G, color, i, 1-x);
            }
        if (color[i] != -1 && color[i] != 1-x) {
            return false;
        }
        if (!val) {
            return false;
        }
```

```
bool isBipartite (int G[][4]) {
    int color[4];
    for(i=0; i<4; i++) {
        color[i] = -1;
    }
    int pos = 0;
    return color(G, color, pos, 1);
}
```

```
int main() {
    int G[][4] = { {1, 0, 0, 1},
                   {0, 1, 1, 0},
                   {1, 0, 0, 0},
                   {0, 0, 0, 1} };
    if (isBipartite(G)) {
        printf("Bipartite Graph.");
    }
    else {
        printf("Not Bipartite Graph");
    }
}
```

(b)

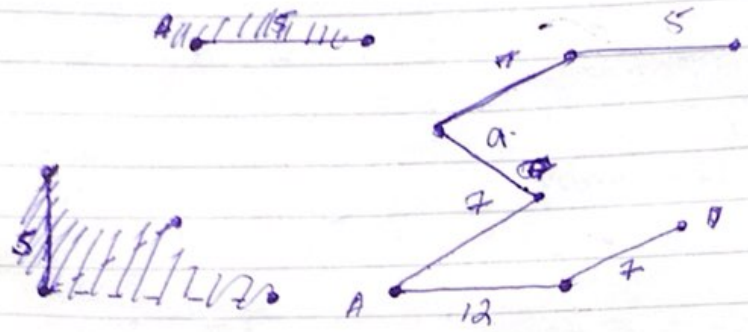
We will need atmost n colors wit one odd cycle containing n vertices in that cycle



n = 3 => Vertices in odd cycle.
Colors = 3 ⇒ Num of colors required atmost

Question # 03 :

(a)



$$\text{Cost} = 12+9+7+7+7+5$$
$$= 47$$

(b)

If weights are different from each other then we get unique max spanning tree. both will give same total cost but if weights are not different we get different spanning trees but same total.

Question #04 :

(1)

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 2 & \infty & \infty & 4 \\ \infty & 0 & 3 & \infty & \infty \\ \infty & \infty & 0 & 5 & 1 \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 7 & 0 \end{bmatrix} \end{array}$$

(2)

$$\begin{bmatrix} 0 & 2 & \infty & \infty & 4 \\ \infty & 0 & 3 & \infty & \infty \\ \infty & 10 & \infty & 0 & 12 \\ \infty & \infty & \infty & 7 & 0 \end{bmatrix}$$

(3)

$$\begin{bmatrix} 0 & 2 & 5 & \infty & 4 \\ \infty & 0 & 3 & \infty & \infty \\ \infty & \infty & 0 & 5 & 1 \\ 8 & 10 & 13 & 0 & 12 \\ \infty & \infty & \infty & 7 & 0 \end{bmatrix}$$

(4)

$$\begin{bmatrix} 0 & 2 & 5 & 10 & 4 \\ \infty & 0 & 3 & 8 & 4 \\ \infty & \infty & 0 & 5 & 1 \\ 8 & 10 & 13 & 0 & 12 \\ \infty & \infty & \infty & 7 & 0 \end{bmatrix}$$

(5)

$$\begin{bmatrix} 0 & 2 & 5 & 10 & 4 \\ 16 & 0 & 3 & 8 & 4 \\ 13 & 15 & 0 & 5 & 1 \\ 8 & 10 & 13 & 0 & 12 \\ 15 & 17 & 20 & 7 & 0 \end{bmatrix}$$

(6)

	0	1	2	3	4
0	0	2	5	10	4
1	16	0	3	8	4
2	13	15	0	5	1
3	8	10	13	0	12
4	15	17	20	7	0

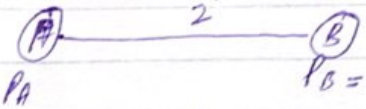
Time complexity :
$O(V^3)$

Space complexity :
$O(V^2)$.

Question # 05

PART 1:

With all pairs Thonson algorithm shows that in graph with negative edges length, all pairs shortest path like can effectively be reduced to just N indications of Dijkstra Algorithm and I indicates of Bellman Ford and same running time complexities. O(mnlogn).



$2 + (-3) = -1$ reweights using vertex weight $[P_A]$ add same amount to every s-t paths

PART 2:

Directed graph $G = (V, E)$

so form G' by adding a new vertex 's' and a new edge (s, v) with length 0 for each $v \in G$.

Run Bellman Ford on G' with vertex s

if B-F detects negative edge cycle in G' (lies in G), halt and report this result.

For each vertex $v \in G$ define P_A = length as short $s \to v$ path in G'.

For each edge $e = (u, v) \in G$, define $c'e = c e + P_A - P_A$

For each vertex u of G run Dijkstra Algorithm in G with edge length with source vertex u, for shortest path distance $d'(u, v)$ for each $v \in G$

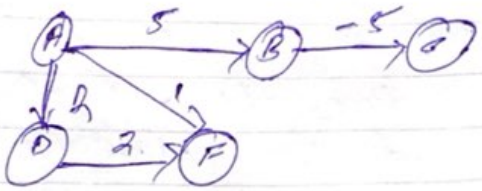
For each pair $u, v \in G$, return the shortest path distance $d(u, v) = d'(u, v) - P_A + P_B$

Running time complexities:

$O(mn \log n)$

Question # 06

(a)



(b)

Not Valid

As adding constant to edge weight increases length of a path with more edges than increases length of a path with only few edges.

Example:

Path A: A —(-2)— B —(4)— C ⇒ Cost = 2

Path B: A —(-4)— B —(2)— C —(3)— D ⇒ Cost = -1

Path B smaller cost

After adding constant.

Path A: A —(0)— B —(8)— C Cost = 8

Path B: A —(0)— B —(5)— C —(7)— D Cost = 13

Now path B greater

So Not Valid.

(c) acyclic

Work only in a directed graph but if there is cycle whose edge weights turn to a negative value. Dijikstra fails but if no negative cycle exists each time the algorithm visit a vertex it will do so when the vertex further and alter any vertex closer.

Question # 7

BFS :
adjacency list : $O(V+E)$
adjacency matrix : $O(V^2)$

DFS :
adjacency list : $O(V+E)$
adjacency matrix : $O(V^2)$

Kruskals :
adjacency list : $O(eV \log V)$
adjacency matrix : $O(E \log E)$

Prims
adjacency list : $O(e \log V)$
adjacency matrix : $O(V^2)$