Sample Solution for Problem Set 9

Data Structures and Algorithms, Fall 2021

December 13, 2021

Contents

1 Problem 1

(a)

vertex	d	p
q	0	null
r	0	null
s	1	q
t	0	null
u	1	r
v	2	s
w	1	q
x	1	t
y	1	r
z	2	x

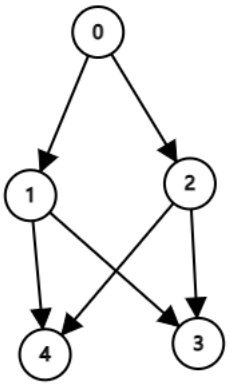
(b)

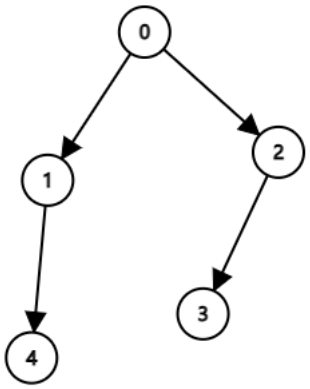
vertex	d	f
q	1	8
r	9	14
s	2	7
t	15	20
u	10	13
v	3	6
w	4	5
x	16	19
y	11	12
z	17	18

- tree edge: (q,s), (s,v), (v,w), (r,u), (u,y), (t,x), (x,z)
- back edge: (w,s), (z,x)
- forward edge: (q,w), (r,y)
- cross edge: (y,q), (t,y)

2 Problem 2

(a)





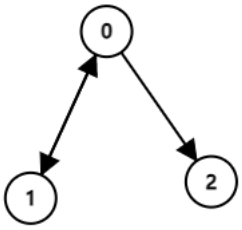
(b)

In the following chart, TBFC stands for tree edge, back edge, forward edge and cross edge respectively.

(i,j)	White	Gray	Black
White	TBFC	BC	C
Gray	TF	TFB	TFC
Black	-	B	TBFC

(c)

In the following counterexample, the values in these nodes indicate the visiting order.



3 Problem 3

```
Graph G;
int cc;

DFS(u):
    belong[u] = cc;
    for each edge (u, v) in G:
        if belong[v] == -1 :
            DFS(v);

main():
    cc = 0;
    for each v in G :
        belong[v] = -1
    for each v in G :
        if belong[v] == -1 :
            ++ cc;
            DFS(v);
```

4 Problem 4

Without loss of generality, we may assume that graph $G = (V, E)$ is connected.

(a)(b)

It can be seen that graph contains odd cycle cannot be bipartite. We will now prove that graph without odd cycle is bipartite.

Fix an arbitrary node $u \in V$. Let $L = \{v \in V | dist(u, v) \text{ is odd.}\}$, and $R = V \setminus L$. It can be proved that $(v, w) \notin E$ for any $v \in L$ and $w \in R$. The proof is standard and we will omit here. Please refer to this thread (you may click here) for more information.

(c)

As mentioned before, we may divide fix an arbitrary node $u \in V$, and set $L = \{v \in V | dist(u, v) \text{ is odd.}\}$. It only remains to check if exactly one of two endpoints in each edge falls into L. Therefore, we may determine whether graph $G = (V, E)$ is bipartite in $O(|V| + |E|)$ using BFS.

5 Problem 5

Algorithm To answer the minimum number of moves required to solve a given number maze, or correctly reports that the maze has no solution, we can use BFS. The pseudocode is givin below.

Here, $G(i, j)$ represents the positive number on square (i, j), $dep(i, j)$ represents the minimum number of moves required to move to that square, which initiates to be -1. And Q is a FIFO queue.

Algorithm 1 Maze

```
function MAZE(G, n, dep, Q)
    for i ← 1, n do
        for j ← 1, n do
            dep(i, j) ← −1
    dep(1, 1) ← 0
    Q.push((1, 1))
    while not Q.empty() do
        (u, v) ← Q.pop()
        for (x, y) in {(u + G(u, v), v), (u − G(u, v), v), (u, v + G(u, v)), (u, v − G(u, v))} do
            if x ≥ 1 and x ≤ n and y ≥ 1 and y ≤ n and dep(x, y) = −1 then
                dep(x, y) ← dep(u, v) + 1
                Q.push((x, y))
    return dep(n, n)
```

Analysis The time complexity is $O(|V| + |E|) = O(n^2)$.

6 Problem 6

6.1 Description

- Create a new graph $G' = (V', E')$.
  - For each vertex $u \in V$, create 3 vertices u_R, u_W, u_B
  - For each edge $(u, v) \in E$ with color c, create a new edge:
    if $c = W$, (u_R, v_W)
    if $c = B$, (u_W, v_B)
    if $c = R$, (u_B, v_R)
- A vertex u can be reached from v in G iff at least one of u_R, u_W, u_B can be reached from u_B in G'. Run BFS or DFS in G' from u_B and check the condition for each vertex is OK.
- We just create a new graph and run DFS/BFS on it, so the time complexity is $O(|V'| + |E'|)$, where $|V'| = 3|V|, |E'| = |E|$. Obviously, it is $O(|V| + |E|)$
- Pseudocode for this algorithm is trivial. We don't provide it here. See CLRS for detailed implementation.

6.2 Remark

Run BFS/DFS directly on G is wrong. For example, Let $V = \{1, 2, 3, 4\}, E = \{(1, 2, R), (2, 3, W), (1, 3, R), (3, 4, B)\}$. Can your algorithm compute the answer correctly for any execution?