ShanghaiTech University

CS101 Algorithms and Data Structures

Fall 2021

Homework 7

Due date: 23:59, October 21, 2021

1. Please write your solutions in English.
2. Submit your solutions to gradescope.com.
3. Set your FULL NAME to your Chinese name and your STUDENT ID correctly in Account Settings.
4. If you want to submit a handwritten version, scan it clearly. Camscanner is recommended.
5. When submitting, match your solutions to the according problem numbers correctly.
6. No late submission will be accepted.
7. Violations to any of the above may result in zero grade.

1: (4×1')Single Choice

The **question1-4** are single choice questions, each question has **only one** correct answer. Select the correct answer.
*Note that you should write you answers of **question1-4** in the box below.*

Question 1	Question 2	Question 3	Question 4
A	C	C	C

Question 1. *A union tree, used in a disjoint set with only union-by-size optimization, has height 6. The number of nodes contained in that tree **can not** be _____.*

(A) 63

(B) 64

(C) 65

(D) 80

Question 2. *Which of the following statements is true?*

(A) If a graph with n vertices has $n-1$ edges, it must be a tree.

(B) A directed graph with n vertices has at least 2n edges to ensure the whole graph is strongly connected.

(C) Both the time complexity of DFS and that of BFS on graph are $\Theta(|V|+|E|)$.

(D) Every edge is visited exactly once in one iteration of DFS on a connected, undirected graph.

Question 3. *Suppose you have an directed graph containing $|V|$ nodes. What is the maximum number of edges you can add to this graph while maintaining it acyclic? (Assume that you are not allowed to add parallel edges.)*

(A) $|V|$

(B) $|V|(|V|+1)/2$

(C) $|V|(|V|-1)/2$

(D) $|V|^2$

Question 4. *Which of the following statements is false?*

(A) In a directed simple graph, the maximal number of edges is $|V|(|V|-1)$.

(B) Undirected graph $G=(V,E)$ is stored in an adjacency matrix A. The degree of V_i is $\sum_{j=1}^{|V|} A[i][j]$.

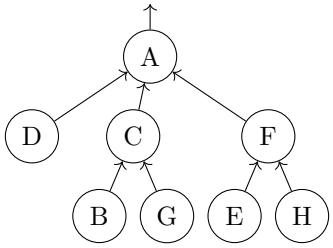
(C) A DFS of a directed graph always produces the same number of tree edges, i.e. independent of the order in which the vertices are considered for DFS.

(D) In BFS, let $d(v)$ be the minimum number of edges between a vertex v and the start vertex. For any two vertices u, v in the fringe, $|d(u)-d(v)|$ is always less than 2.

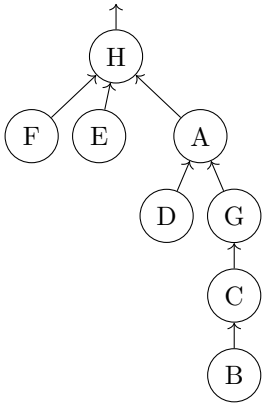
2: (2×2') Disjoint set practice

Given the following set of operations on a disjoint set, show the final disjoint set tree for each of the following optimization strategies.
$set\_union(A, D),\ set\_union(C, B),\ set\_union(F, E),\ set\_union(G, C),$
$set\_union(D, G),\ find(A),\ set\_union(H, E),\ set\_union(E, G),\ find(E)$

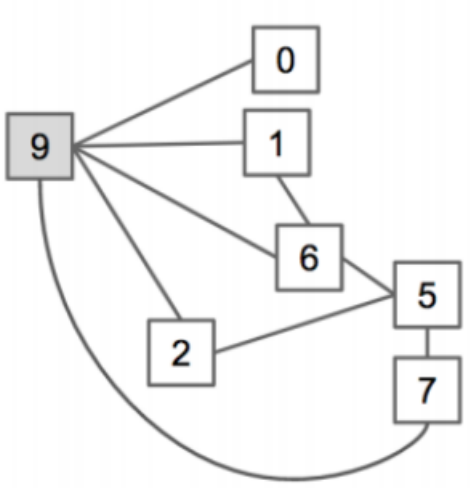
(1)(2') Only with union-by-size optimization. (When two trees have the same height, the set specified first in the union will be the root of the merged set.)



(2)(2') Only with path compression. (The set specified first in the union will always be the root of the merged set.)



3: (3×2') Graph traversal practice



(a) For the graph above, give the depth first search preorder traversal starting from vertex 9, assuming that we break ties by visiting smaller numbers first.
9, 0, 1, 6, 5 ,2 7

(b) For the graph above, give the depth first search postorder traversal starting from vertex 9, assuming that we break ties by visiting smaller numbers first.
0, 2, 7, 5, 6, 1, 9

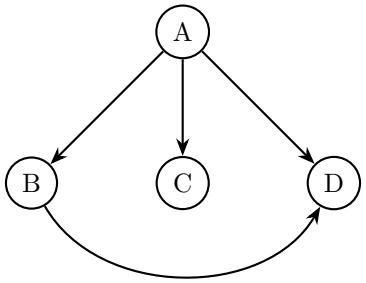
(c) For the graph above, give the breadth first search traversal starting from vertex 9, assuming that we break ties by visiting smaller numbers first.
9, 0, 1, 2, 6, 7, 5

4: (4') DFS went wrong!

The following algorithm which runs DFS on a directed graph, but it contains an fatal error.

```
Create a stack.
Choose the initial vertex and mark it as visited.
Put the initial vertex onto the stack.
while the stack is not empty:
    Pop a vertex V from the top of the stack.
    for each neighbor of V:
        if that neighbor is not marked as visited:
            Mark that neighbor as visited.
            Push that neighbor onto the stack.
```

Please give a graph as an counterexample and briefly explain why this algorithm is wrong.
Note: In this problem, we say a node is "visited" whenever it is marked as visited.

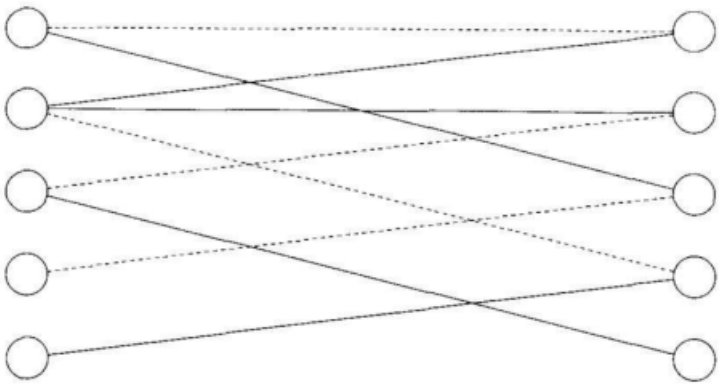


In the above graph, the correct DFS order (preorder) from node A should be A, B, D, C, or A, C, B, D, or A, C, D, B, or A, D, B, C, or A, D, C, B. However, the algorithm above will give A, B, C, D,or A, B, D, C or A, C, B, D, or A, C, D, B, or A, D, B, C, or A, D, C, B. The order A, B, C, D is wrong.This is because D is marked as visited while it actually haven't been visited. The problem of the algorithm is that the vertex should be marked as visited when poped, not pushed. It can be modified as follows.

```
Create a stack.
Choose the initial vertex and mark it as visited.
Put the initial vertex onto the stack.
while the stack is not empty:
        Pop a vertex V from the top of the stack.
        if V is not marked as visited:
                Mark V as visited.
        for each neighbor of V:
                if that neighbor is not marked as visited:
                        Push that neighbor onto the stack.
```

5: (2'+3') Bipartite graph

A bipartite graph, $G = (V, E)$, is a graph such that V can be partitioned into two subsets V_1 and V_2 and no edge has both its vertices in the same subset. See the figure below.



(a) Show that every tree is a bipartite graph.

Firstly, put root into V_1. For each node in the tree, if its parent is in V_1, put the node into V_2, else put the node into V_1. Since each node are connected noly to its parent and children, the previous construction method, each node are in the different set to its parent and children, it doesn't connect to any other node in the same set. Therefore, each tree is a bipartite graph.

(b) The bipartite matching problem is to find the largest subset E' of E such that no vertex is included in more than one edge. A matching of four edges (indicated by dashed edges) as
Suppose now you have the solution of bipartite matching problem, let's call it algorithm A. Show how the bipartite matching problem can be used to solve the following problem: We have a set of workers, a set of tasks, and a list of tasks that each worker is qualified to finish. If no worker is required to finish more than one task, and each worker can only finish his qualified tasks, what is the maximum number of tasks that can be finished?
(You just need to show how to construct this problem as one bipartite matching problem, and do not need to give the detail of algorithm A.)

We see both workers and tasks as vertices in the graph. If a worker is qualified to finish the task according to the list, the worker vertex and the task vertex are connected. Since no connection among workers, nor among tasks, the graph is a bipartite graph $G(V, E)$ with worker vertices been put into V_1 and task vertices been put into V_2. The degree of the worker vertices indicates the number of tasks the worker is going to finish and the degree of the task vertex indicates the how many times it is to be finished. No worker is required to finish more than one task, and no task need to be finished more than once (of course). Therefore, no vertex is included in more than one edge.(*) We want to find the maximum number of tasks that can be finished means we want to find the largest subsets of E meets the previous constraint (*). This is a bipartite matching problem.