Name: Rollno:

ESO207: Data Structures and Algorithms (Quiz 2)

5th November 2023

Total Number of Pages: 5 Time: 50 min Total Points 40

Instructions

1. All questions are compulsory.
2. Answer all the questions in the question paper itself.
3. MCQ type questions can have more than one correct options.
4. The symbols or notations mean as usual unless stated.
5. You may cite and use algorithms and their complexity as done in the class.
6. Cheating or resorting to any unfair means will be severely penalized.
7. Superfluous and irrelevant writing will result in negative marking.
8. Using pens (blue/black ink) and not pencils. Do not use red pens for answering.

Question	Points	Score
1	3	
2	3	
3	4	
4	3	
5	2	
6	3	
7	3	
8	3	
9	3	
10	3	
11	10	
Total:	40	

Helpful hints

1. It is advisable to solve a problem first before writing down the solution.
2. The questions are *not* arranged according to the increasing order of difficulty.

Name: Rollno:

Question 1. (3 points) Which of the following statements is/are correct?

- √ **There is no bipartite graph containing a cycle of length 7.**
- √ **A bipartite graph can be 2-coloured, i.e. every vertex can be assigned one out of 2 colours so that no two adjacent vertices have the same colour.**
- ☐ A bipartite graph can contain a sub-graph which is not bipartite.
- √ **Any graph with maximum degree 1 is bipartite.**

Question 2. (3 points) You are given the adjacency matrix A of an undirected graph $G = (V, E)$.

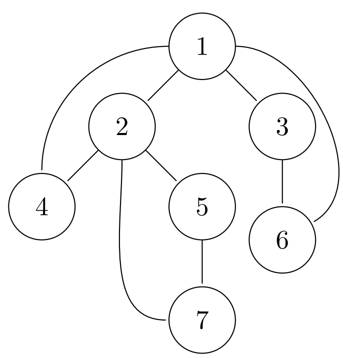
The $(i, j)^{th}$ entry of the exponentiated matrix A^k, $A^k[i, j]$ is the number of **walks** of length k between vertices i and j.

We can check whether there exists a path of length k between vertices i and j in $O(|V| + |E|)$ time.

Question 3. (4 points) Consider the following pseudo-code for performing DFS on a connected, undirected graph. The value of the global variable *time* is 0 initially.

Algorithm 1: DFS(u)

```
visited[u] ← True;
start[u] ← time;
time ← time + 1;
for all v ∈ V such that (u, v) ∈ E do
    if not visited[v] then
        DFS(v);
    end
end
end[u] ← time;
time ← time + 1;
```



Suppose we call **DFS(1)** on the above graph, and assume that all vertices are in numerically increasing order in all adjacency lists. Answer the following:

(a) The $[start[\cdot], end[\cdot]]$ intervals for vertex 1 is **[0,13]**

(b) The $[start[\cdot], end[\cdot]]$ intervals for vertex 2 is **[1,8]**

(c) The $[start[\cdot], end[\cdot]]$ intervals for vertex 6 is **[10, 11]**

(d) What can be said about the intervals $[start[u], end[u]]$ and $[start[v], end[v]]$ if u is an ancestor of v in the DFS tree: **v's interval contained in u's.**

Question 4. (3 points) Let T be a DFS tree obtained by doing DFS in a connected undirected graph G. Which of the following options is/are correct?

☐ Root of T can never be an articulation point in G.

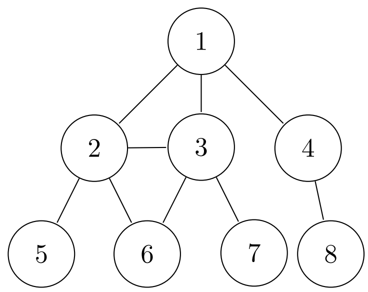
√ **Root of T is an articulation point in G if and only if it has 2 or more children.**

☐ A leaf of T can be an articulation point in G.

☐ If u is an articulation point in G such that x is an ancestor of u in T and y is a descendent of u in T, then all paths from x to y in G must pass through u.

Question 5. (2 points) Let x be a b-approximate median of a set S containing n distinct integers. Then $|S_{>x}|$ is at least $\underline{\quad bn \quad}$ and at most $\underline{\quad (1-b)n \quad}$.

Question 6. (3 points) Which among the following is/are valid BFS traversals of the given graph starting from node 1?



☐ 1, 2, 3, 6, 5, 7, 4, 8

√ **1, 2, 3, 4, 6, 5, 7, 8**

☐ 1, 4, 2, 3, 8, 5, 7, 6

√ **1, 3, 2, 4, 7, 6, 5, 8**

Question 7. (3 points) Take a sequence S having length $n = 2^L$ for an integer $L > 0$. We build a binary tree on intervals of this sequence to perform dynamic range-minima queries, i.e., range minima calculation and element query updations efficiently in $O(\log n)$ time. Refer to the implementation done in class. Let

$$f(n) = \text{Number of nodes compared when } Update(i, x) \text{ is called for any } i, x$$

$$g(n) = \text{Number of nodes compared when } Report\_Min(0, n-1) \text{ is called}$$

Then $f(n) = \underline{\quad 1 + \log_2 n \quad}$ and $g(n) = \underline{\quad 2\log_2 n \quad}$ (write exact values in terms of n).

Question 8. (3 points) Rank the following data structures in non-decreasing order of their worst-case asymptotic time complexity for querying the minimum element in that data structure (ignore any time taken for pre-computing/building the data structure).

(a) Linear array

(b) Min-heap

(c) Max-heap

(d) Balanced BST

Solution: (b) < (d) < (a) = (c)

Question 9. (3 points) Which of the following problems can be solved in $O(|V| + |E|)$ time given the adjacency list representation of an undirected graph?

$\surd$ **Checking graph regularity, i.e. does all vertices have the same degree.**

$\surd$ **Checking if the graph is biconnected.**

$\surd$ **Computing all connected components of the graph.**

$\surd$ **Determining if the graph contains a cycle.**

Question 10. (3 points) Let H be the min-heap formed starting from array $A = [100, 25, 7, 36, 19, 17, 3, 2, 1]$, and calling the linear time operation $Build\_Heap(A)$ on it.

(a) Height of H is ___**4**___ (the number of nodes on the longest path from root to a leaf).

(b) Rightmost element in the bottom-most level of H is ___**36**___.

(c) The answer to the above part after $Extract\_Min(H)$ is called is ___**100**___.

Question 11. **Finding Tree Centroid**: We are given a tree graph $T = (V, E)$ in the adjacency list format. Let $d(u, v)$ be the distance, i.e. the (unique) path length between vertices $u, v \in V$. We define the **centroid** of T as the vertex c which minimizes the sum of distances from itself to all vertices, i.e.

$$c = \arg\min_{u \in V} S(u) = \arg\min_{u \in V} \left\{ \sum_{v \in V} d(u, v) \right\}$$

We first root our tree graph at an arbitrary root r. We then pre-compute the following values for all vertices of the tree T using DFS starting at r:

1. $size[v]$: the size of the sub-tree of v in the rooted tree.
2. $distance[v]$: the distance from the root r to vertex v.

(a) (5 points) Write pseudo-code for the procedure **DFS_precompute**$(u, parent)$ which fills the appropriate *size* and *distance* values for u. Here u is the current vertex being traversed in DFS and *parent* is the previous vertex it was called from, i.e. its parent in the tree. You are allowed to use a *visited* array (but it is not necessary). It should work in $O(\deg u)$ time other than recursive calls, the same as for a standard DFS.

Solution:

Algorithm 2: DFS_precompute$(u, parent)$

```
if parent ≠ NULL then
    distance[u] ← distance[parent] + 1;
end
size[u] ← 1;
for all v such that (u, v) ∈ E do
    if v ≠ parent then
        DFS_precompute(v, u);
        size[u] ← size[u] + size[v];
    end
end
```

(b) (5 points) Write pseudo-code for the procedure **Centroid**(T) which returns the centroid vertex of T in $O(|V|)$ time. You are allowed to use **DFS_precompute** as a subroutine and can define and use other subroutines. (Hint: how to calculate sum of distances from a particular vertex given sum of distances from its parent?)

Solution:

Algorithm 3: Centroid(T)

```
Function DFS(u, parent)
    if parent ≠ NULL then
        S[u] ← S[parent] + |V| − 2 · size[u];
    end
    for all v such that (u, v) ∈ E do
        if v ≠ parent then
            DFS(v, u);
        end
    end
return

Function main()
    DFS_precompute(r, NULL);
    S[r] ← 0;
    for all v ∈ V do
        S[r] ← S[r] + distance[v];
    end
    DFS(r, NULL);
    centroid ← r;
    for all v ∈ V do
        if S[v] < S[centroid] then
            centroid ← v;
        end
    end
return centroid;
```