DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Indian Institute of Technology Jodhpur

Advanced Data Structures and Algorithms (CSL 7560)

Data Structures and Algorithmic Techniques (CSL 7561) Instructor: Pallavi Jain

Friday 8th September, 2023

Time: 1 hour | *Minor 1* | Maximum Marks: 15

Instructions: In True/False, wrong answer is worth -0.5.

1. A red-black tree must have at least one red node. **(True or False)** [1]

2. Suppose we want to maintain a sequence S of n numbers to support, besides the usual dictionary operations insert, search, and delete, $find(k, S)$, which finds the k-th smallest element in the sequence.
   (a) How would you augment a balanced binary search tree to support this operation in $O(\log n)$ time? [2]
   (b) Explain, how insert and delete operations can still be maintained in $O(\log n)$ time? [2]

3. How do you determine whether a graph is connected? What is the complexity of your algorithm? [2]

4. In a Red-Black tree, can a red node have exactly one black child? Justify your answer. [2]

5. Let $A[1, \ldots, n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called a *bad pair* of A.
   (a) List the five bad pairs of the array $\langle 2, 3, 8, 6, 1 \rangle$. [1]
   (b) What array with elements from the set $\{1, 2, \ldots, n\}$ has the most bad pairs? How many does it have? [2]
   (c) Design an algorithm to count the number of bad pairs in an array of size n in time $O(n \log n)$? [Hint: you can try to use order-statistic tree.] [3]

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Indian Institute of Technology Jodhpur

Advanced Data Structures and Algorithms (CSL 7560)

Data Structures and Algorithmic Techniques (CSL 7561) Instructor: Pallavi Jain

Tuesday 17th October, 2023

Time: 1 hour | *Minor 2* | Maximum Marks: 15

1. For the following problems, mention whether they are solvable in polynomial time or NP-complete. No proofs are necessary. For problems where there is an additional parameter k, it is an integer, and it is part of the input, and could be as large as n. **The wrong answer is worth -0.5.**
   (a) Given an undirected graph on n vertices and two designated vertices s and t, is there a path of length *at most* k between s and t? [1]
   (b) Given an undirected graph on n vertices and two designated vertices s and t, is there a path of length *at least* k between s and t? [1]
   (c) Given a graph G, a set $S \subseteq V(G)$ is called a *vertex cover* of G if for every edge $uv \in E(G)$, either u or v is in S. Given an undirected graph on n vertices, does it have a vertex cover on 15 vertices? [1]
   (d) Given a graph G, a set $S \subseteq V(G)$ is called a *clique* if G has an edge between every pair of vertices in S. Given a bipartite graph, does it have a clique on k vertices? [1]
   (e) Given a graph G, partition the vertex set into sets X and Y such that X and Y are independent sets in G. [1]
   (f) Find the smallest independent set in a graph. [1]

2. Show ONE of the following problems is NP-complete.
   (a) Given an undirected graph and an integer k, does it have at most k vertices that cover all cycles in the graph? I.e. the removal of the k vertices makes the graph acyclic. [4]
   (b) Given a set S of positive integers, is there a way to partition S into two subsets S_1 and S_2 that have the same sum? [4]

   You can assume that the following two problems are NP-Complete.

   - VERTEX COVER: Given a graph G and an integer k, find a vertex cover of G of size at most k.
   - SUBSET SUM: Given a set S of positive integers and a target integer T, is there a subset of S whose sum is T?

3. Consider the flow network D given in Figure 1.
   (a) Draw the residual network R of D [1]
   (b) Show an augmenting path in the residual network R. [1]
   (c) Show an augmented flow for D. [1]

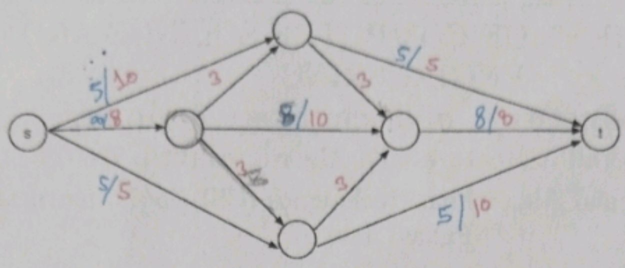

Figure 1: The red colored number shows capacity and blue colored numbers show flow sent on each edge.

OR

Let G be a graph of maximum degree two that has weights on the vertices. Design a polynomial time algorithm to find a maximum weight independent set of G. [3]

4. Consider the following modification to the generic Ford-Fulkerson augmenting path algorithm. Instead of maintaining a residual graph, just reduce the capacity of edges along the augmenting path. In particular, whenever we saturate an edge, just remove it from the graph. Does this algorithm compute a maximum flow? Justify your answer. The pseudocode is in Algorithm 1. [2]

```
for every edge e ∈ E(G), f(e) = 0
while there is a path from s to t do
    let P be an arbitrary path from s to t ;
    let F be minimum capacity of any edge in P ;
    for every edge e in P do
        f(e) = f(e) + F ;
        if c(e) = F then
            remover e from G
        else
            c(e) = c(e) − F
        end if
    end for
end while
```

Algorithm 1: Algorithm for Max Flow