CS 580 Algorithm Design, Analysis, And Implementation
Fall 2021 Vassilis Zikas Midterm

1. (50 points total, 5 + 25 + 20 points) You are given a directed graph $G : (V, E)$. Each node v has a weight w_v associated with it. Consider the following parts:

   (a) Given a start node $s \in V$ design an algorithm that outputs the maximum weight of any node reachable [1] by s. Give time complexity and provide a brief proof of correctness.

   (b) Now suppose that G is a DAG. Design an algorithm that, **for all** $|V|$ **nodes** $v \in V$, outputs the maximum weight of any node reachable by v. Give time complexity and provide a brief proof of correctness.

   (c) Now suppose that G is no longer a DAG i.e. it may contain directed cycles. Design an algorithm that, for all nodes $v \in V$, outputs the maximum weight of any node reachable by v. Give time complexity and provide a brief proof of correctness. You can use an algorithm for part (b) as a black box here.

   Answer:

   (a) BFS/DFS. Time: $O(n+m)$.

   (b) Topological Sort followed by DP with the subproblem $OPT(v)$ = maximum weight of any node reachable by v. Time complexity $O(n+m)$.

   (c) Find SCC in G. Reduce each SCC to one node with weight equal to maximum weight of any node in the SCC. Apply algorithm for the previous part. Time: $O(n+m)$.

2. (50 points total, 20 + 10 + 20 points)

   (a) You are given a set of teachers T and a set of courses C. Each teacher $t_i \in T$ has a set of courses $c_i \subseteq C$ that they are willing to teach. Each course can be assigned at most one teacher and each teacher can teach at most three courses in the semester. Design and analyze an algorithm to find teacher-course assignments that maximizes the number of courses with teachers assigned. Provide a brief proof of correctness. Assume there are n teachers and m courses.

   (b) Suppose that you are given a directed graph $G : (V, E)$. Let $u, v, w, t \in V$ be four distinct vertices in G. Design an algorithm that gives three edge-disjoint paths from t to each one of u, v, and w i.e. one from t to u, another from t to v, and the third from t to w. If three such paths do not exist, output None. Provide a brief proof of correctness. Recall that a path does not repeat vertices.

   (c) Suppose that you are given a directed graph $G : (V, E)$. Let $u, v, w, t \in V$ be four distinct vertices in G. Design an algorithm that gives three **vertex-disjoint** paths from t to each one of u, v, and w i.e. t must be the only common vertex between any two paths. If three such paths do not exist, output None. Provide a brief proof of correctness.

   Answer:

   (a) Reduce to max flow. Connect source node to each teacher node with edge capacity 3. Teachers to course nodes that they are willing to teach with capacity 1 edges. And course nodes to target node with capacity 1 edges.

[1] We say a node t is reachable by s, if there exists a s-t path in G.

(b) Connect u, v, w to a new node t'. Find max-flow between t and t'.

(c) Same as previous part and then add vertex disjoint constraints i.e. split each node v into two nodes v_{in}, v_{out} with an edge of capacity 1 between them. Edge (u, v) changes to (u_{out}, v_{in}). Run max flow.

3. (50 points total, 25 + 5 + 20 pints)

(a) You have found the MST T of a huge graph where the weight of a an edge (v_1, v_2) was entered as 10. This weight is later updated. How would you update your MST in each of the following cases:

i. (v_1, v_2) was in T, and the updated weight is 5.

ii. (v_1, v_2) was in T, and the updated weight is 20.

iii. (v_1, v_2) was not in T, and the updated weight is 5.

iv. (v_1, v_2) was not in T, and the updated weight is 20.

For each part analyze the running time of your proposed solution. and provide a brief proof of correctness. Here we are looking for solutions that update T instead of building a new spanning tree of the updated graph.

(b) Find the maximum and minimum element of an array of integers using divide and conquer in $O(n \log n)$ time. Provide a brief proof of correctness.

(c) Consider the following divide and conquer algorithm that returns the maximum and minimum element of an array A. Complete the blanks in the algorithm. Write a recurrence relation for the runtime of the algorithm and solve it to find the runtime.

Algorithm 1: FindMaxMin(A)

```
if |A| = 1 then
    // Base Case
    return _____, _____
else
    Split array A in half to obtain A_l and A_r;

    // Obtain outputs from both halves
    max_l, min_l ← FindMaxMin(A_l)
    _____, _____ ← FindMaxMin(A_r)

    // Reconstruct extra information for next level of recursion
    max_A ← _____
    min_A ← _____

    return max_A, min_A
end
```

Answer:

(a) i. No update necessary.

ii. Use cut property. Takes linear time.

iii. Use cycle property. Takes linear time.

iv. No update necessary.

(b) Merge sort and then return first and last element.

(c) The blanks are: $A[0], A[0], \max_r, \min_r, \max(\max_l, \max_r), \min(\min_l, \min_r)$. Recurrence relation is $T(n) = 2T(\frac{n}{2}) + O(1)$. Therefore, $T(n) = O(n)$.