Name(last, first): ______________________________

ID (rightmost 4 digits): _ _ _ _

U C L A **Computer Science Department**

CS 180 Algorithms & Complexity

Final Exam **Total Time: 3 hours** **December 10, 2019**

***** Write all algorithms in bullet form (as done in the past) *****

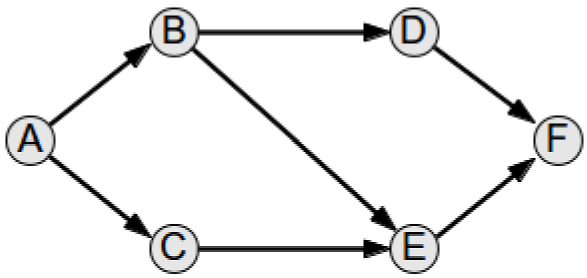
You need to prove EVERY answer that you provide.

There are a total of 8 pages including this page.

1. (20 points: each part has 10 points)

a. Consider a S-T network N. **Prove** that if **f** is a maxflow in the network N then there is a cut C with its capacity equal to **f**.

b. Consider the following network with source **A** and sink **F**. If the Ford-Fulkerson max flow algorithm initially finds the path **A,B,E,F** in the network below and sends **2 units** of flow on it, **show the residual network** (also known as the augmented network) and all subsequent steps of Ford-Fulkerson algorithm on this network (all capacities are equal to **2**).



2. (15 points) a. Given a n x n matrix where all numbers are distinct, design an **efficient algorithm** that finds the maximum length path (starting from any cell) such that all cells along the path are in increasing order with a difference of 1.

b. Analyze the time complexity of your algorithm

We can move in 4 directions from a given cell (i, j), i.e., we can move to (i+1, j) or (i, j+1) or (i-1, j) or (i, j-1) with the condition that the adjacent cells have a difference of 1.

```
Input:  mat[][] = {{1, 2, 9}
                   {5, 3, 8}
                   {4, 6, 7}}
Output: 4
The longest path is 6-7-8-9.
```

3. (20 points: Each part has 10 points)

a. Consider **d** sorted array of integers each containing n1, n2, .. nd numbers. The numbers ni's can be very different. The total number of all elements is **n** (sum of all ni's). Design an **O(n log d)** algorithm that merges all arrays into one sorted list. You may wish to use a data structure that we have discussed in class.

b. Prove a lower bound on sorting **n** numbers in the decision tree model (using comparison exchange).

4. (15 points)

Consider an array a1,...,an of n integers, that is hidden from us. We have access to this array through an procedure knapsack(.,.). For a set S ⊆ {1,...,n} and an integer k, knapsack(S,k) will output “yes” if there is a subset T ⊆ S such that the numbers indexed in T add up to k, and it will output “no” otherwise.

Design an algorithm that calls knapsack only O(n) times and outputs a set S ⊆{1,...,n} such that the numbers indexed in S add up to k, if such a set exists. You can use ONLY the knapsack function (e.g., you cannot sort the numbers or do any other operations on them).

For example, suppose a1 = 2, a2 = 4, a3 = 3, a4 = 1, and k = 7. Then, knapsack({1,2,3,4},7) returns “yes” and knapsack({1,3,4},7) returns “no”. In this case your algorithm can output either of the sets {1,2,4} or {2,3}. Note that for example {1,2,4} are indices of the numbers, that is, a1, a2, and a4.

5. (15points)
A **Hamiltonian cycle** in a graph with **n** vertices is a cycle of length n, i.e., it is a cycle that visits all vertices exactly once and returns back to the starting point. A **Hamiltonian path** in a graph with n vertices is a path of length n−1, i.e., it is a path that visits all vertices of the graph exactly once.
Hamil-cycle problem is defined as follows: Given a graph G = (V,E), does it have a Hamiltonian cycle? Hamil-path problem is defined as follows: Given a graph G = (V,E), does it have a Hamiltonian path?
Prove that Hamil-path is polynomial-time transformable to Hamil-cyle.
That is Hamil-path ≤P Hamil-cycle.

6. (15 points)

You are given a tree T where every node **i** has weight **wi** ≥ 0.

a. Design a polynomial time algorithm to find the weight of the largest weight independent set in T: among all independent sets one with maximum sum of the weights (an **independent set** is a subset of vertices where there are no edges between any of them).

For example, suppose in the following picture w1 = 3, w2 = 1, w3 = 4, w4 = 3, w5 = 6. The maximum independent set has nodes 3,4,5 with weight 4 + 3 + 6 = 13.

```
1
| \
2  3
| \
4  5
(1 is connected to 2 and 3.  And 2 is connected to 4 and 5)
```

b. Analyze the time complexity of your algorithm.