CS430 Summer 2024

Final

(Ethics: Any behavior on any homework or exam that could be considered copying or cheating will result in an immediate zero on the assignment or exam for all parties involved and will be reported to academichonesty@iit.edu. See the IIT Code of Academic Honesty, https://web.iit.edu/student-affairs/handbook/fine-print/code-academic-honesty)

1. **Greedy Algorithm (10 + 10 pts)**

In Lecture 13, we discussed three greedy algorithms for the Knapsack Problem:

1) Iteratively choosing the item with the largest value.
2) Iteratively choosing the item with the smallest size.
3) Iteratively choosing the item with the largest unit value (the ratio between value and size).

We discussed in class that none of these approaches are guaranteed to give an optimal solution to the Knapsack Problem; but we didn't discuss how "wrong" the solutions given by these algorithms can be. Actually, these three algorithms can all give arbitrarily "wrong" solutions.

For example, let's consider the following Knapsack problem instance: we have a knapsack of size 2 and we have three items:

Item index i	1	2	3
Size s_i	1	1	2
Value v_i	1	1	$2B$

Here, B is a very large number (it can be arbitrarily large). It is easy to see, the optimal solution should be choosing item 3, and the total value in the knapsack will be $2B$. However, if we use the greedy algorithm 2) (in which we iteratively choose the item with the smallest size), we will end up choosing items 1 and 2 with total value in the knapsack $= 2$. We can see that, the ratio between the optimal solution and the greedy solution is:

$$\frac{2B}{2} = B,$$

which can arbitrarily large.

Here is our task: for each of the greedy algorithms 1) and 3), construct a Knapsack problem instance, so that the ratio between the optimal solution to that instance and the greedy solution can be arbitrarily large. In other words, for each of the greedy algorithms 1) and 3), do what we did above for the greedy algorithm 2).

2. **Minimum Spanning Trees (20 pts)**

Let $G = (V, E)$ be a weighted connected undirected simple graph with $|V| \geq 3$. Prove or disprove the following conjecture:

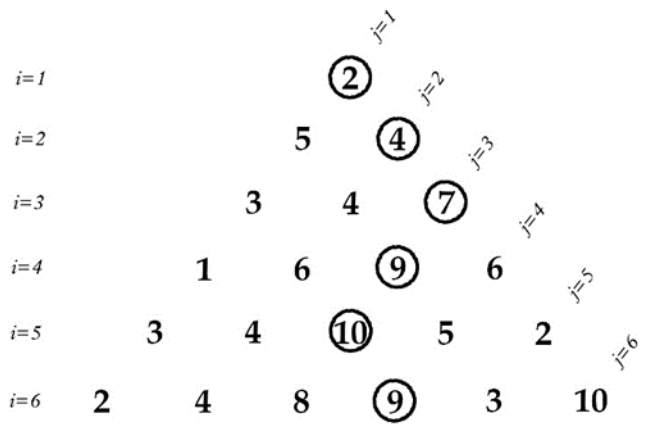
> If G contains exactly two edges e_1 and e_2 with the same minimum weight among all edges of G, then every minimum spanning tree of G contains both e_1 and e_2.

3. **Shortest Paths (20 pts)**

Jim adopted a dog recently. To walk the dog, he wants to find a short simple cycle that starts and ends at his house and consists of unduplicated streets. He generalizes his problem to a graph problem: given an undirected graph $G = (V, E)$ with positive edge weights: $w: E \to \mathbb{R}^+$ and a vertex $s \in V$, how to find a non-empty simple cycle (which means a cycle that contains at least one edge and no repeating vertices) containing s with the minimum weight? Describe an algorithm that solves this graph problem in $O(|V||E| + |V|^2 \lg |V|)$ time.

4. **Dynamic Programming (20 pts)**

We sometimes call a triangle-shaped $n \times n$ array a pyramid. Here is an example of a pyramid of size 6×6. We want to find a path from the peak ($i = 1, j = 1$) of the pyramid to the base ($i = n$, or $i = 6$ in the example) through a sequence of adjacent numbers, one per each level, such that the sum of numbers on the path is maximized (as shown by the circled numbers).



This problem has an optimal substructure. We define $b(i, j) =$ "the sum of numbers on the optimal path from spot (i, j) to the base of a pyramid". Present a recursive definition of $b(i, j)$ that can help us to solve the above problem using dynamic programming. Please be careful of all the base cases.

5. **Graph Search (20 pts)**

We have the following conjecture about depth-first search:

> In a directed graph, if there is a path from vertex u to v, and the discover time of u is earlier than the discover time of v in some depth-first search, then u is an ancestor of v in the depth-first forest produced.

Name each vertex, give direction on each edge in the following graph, then show a depth-first search procedure on the graph that is a counterexample to the above conjecture.