CS 161 Winter 2021 Section 8

March 5, 2021

Warm-up: Greedy or Not?

Sometimes it can be tricky to tell when a greedy algorithm applies. For each problem, say whether or not the greedy solution would work for the problem. If it wouldn't work, give a counter example.

1. You have unlimited objects of different sizes, and you want to completely fill a box with as few objects as possible. (Greedy: Keep putting the largest object possible in for the space you have left)
2. You have unlimited objects, all of which are size 3^k for some integer k, and you want to completely fill a box with as few objects as possible. (Greedy: same approach as the previous problem)
3. You have lines that can fit a fixed number of characters. You want to print out a sentence using as few lines as possible. It doesn't matter if words are split between lines (Greedy: Fit as many characters as you can on a given line)
4. You want to get from hotel 1 to hotel n, and you can travel at most k distance between hotels before collapsing from exhaustion. Find the minimum cost of hotels. (Greedy: Go as far as you can before stopping at a hotel)

Cutting Ropes

Suppose we are given n ropes of different lengths, and we want to tie these ropes into a single rope. The cost to connect two ropes is equal to sum of their lengths. We want to connect all the ropes with the minimum cost.

For example, suppose we have 4 ropes of lengths 7, 3, 5, and 1. One (not optimal!) solution would be to combine the 7 and 3 rope for a rope of size 10, then combine this new size 10 rope with the size 5 rope for a rope of size 15, then combine the rope of size 15 with the rope of size 1 for a final rope of size 16. The total cost would be 10 + 15 + 16 = 41. (Note: the optimal cost for this problem is 29. How might you combine the ropes for that cost?)

Find a greedy algorithm for the minimum cost and prove the correctness of your algorithm.

Dice Probabilities

We wish to find the probability that rolling k 6-sided fair dice will result in a sum S. Devise an algorithm to find this probability.

Egg Dropping

We have e eggs and f floors. We want to find the minimum number of drops needed in the worst case to find the highest floor in which an egg will not break after dropping. Assume that all of the eggs are the same strength; if one egg breaks after dropping from a floor, all eggs will break after dropping from that floor.

Design an algorithm that returns the minimum number of drops needed to accomplish this task in the worst case, without actually dropping any eggs.