2023-07-22 - Handout – Greedy Algorithms

Algo's Refresher: Fractional Knapsack problem, Huffman Coding, Prims & Kruskal's algorithm on Minimum Spanning Tree, Dijkstra's Single Source Shortest Path

Q1. Jump Game II

*You are given a **0-indexed** array of integers nums of length n. You are initially positioned at nums[0].*

Each element nums[i] represents the maximum length of a forward jump from index i. In other words, if you are at nums[i], you can jump to any nums[i + j] where:

- *0 <= j <= nums[i] and*
- *i + j < n*

Return the minimum number of jumps to reach nums[n - 1]. The test cases are generated such that you can reach nums[n - 1]..

Example 1: **Input:** nums = [2,3,1,1,4] **Output:** 2

Constraints (It's guaranteed that you can reach nums[n - 1])**:**

- 1 <= nums.length <= 10^4
- 0 <= nums[i] <= 1000

Q2. Minimum Cost to Hire K Workers

There are n workers. You are given two integer arrays quality and wage where quality[i] is the quality of the i^{th} worker and wage[i] is the minimum wage expectation for the i^{th} worker.

We want to hire exactly k workers to form a paid group. To hire a group of k workers, we must pay them according to the following rules:

1. Every worker in the paid group should be paid in the ratio of their quality compared to other workers in the paid group.
2. Every worker in the paid group must be paid at least their minimum wage expectation.

Given the integer k, return *the least amount of money needed to form a paid group satisfying the above conditions*. Answers within 10^{-5} of the actual answer will be accepted.

Example 1:

Input: quality = [10,20,5], wage = [70,50,30], k = 2

Output: 105.00000

Explanation: We pay 70 to 0^{th} worker and 35 to 2^{nd} worker.

Constraints:

- n == quality.length == wage.length
- 1 <= k <= n <= 10^4
- 1 <= quality[i], wage[i] <= 10^4

Q3. Optimize Water Distribution in a Village

There are n houses in a village. We want to supply water for all the houses by building wells and laying pipes.

For each house i, we can either build a well inside it directly with cost wells[i - 1] (note the -1 due to **0-indexing**), or pipe in water from another well to it. The costs to lay pipes between houses are given by the array pipes where each pipes[j] = [$house1_j$, $house2_j$, $cost_j$] represents the cost to connect $house1_j$ and $house2_j$ together using a pipe. Connections are bidirectional, and there could be multiple valid connections between the same two houses with different costs.

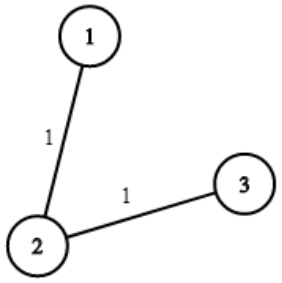
Return *the minimum total cost to supply water to all houses*.

Input: n = 3, wells = [1,2,2], pipes = [[1,2,1],[2,3,1]]

Output: 3

Explanation: The image shows the costs of connecting houses using pipes.

The best strategy is to build a well in the first house with cost 1 and connect the other houses to it with cost 2 so the total cost is 3.



Constraints:

- 2 <= n <= 10^4
- wells.length == n
- 0 <= wells[i] <= 10^5
- 1 <= pipes.length <= 10^4
- pipes[j].length == 3
- 1 <= $house1_j$, $house2_j$ <= n
- 0 <= $cost_j$ <= 10^5
- $house1_j$!= $house2_j$