Programming Assignment #1

Quiz, 3 questions

3/3 points (100%)

Congratulations! You passed!

Next Item

1 / 1 point

1.

In this programming problem and the next you'll code up the greedy algorithms from lecture for minimizing the weighted sum of completion times..

Download the text file below.

jobs.txt

This file describes a set of jobs with positive and integral weights and lengths. It has the format

[number_of_jobs]

[job_1_weight] [job_1_length]

[job_2_weight] [job_2_length]

...

For example, the third line of the file is "74 59", indicating that the second job has weight 74 and length 59.

You should NOT assume that edge weights or lengths are distinct.

Your task in this problem is to run the greedy algorithm that schedules jobs in decreasing order of the difference (weight - length). Recall from lecture that this algorithm is not always optimal. IMPORTANT: if two jobs have equal difference (weight - length), you should schedule the job with higher weight first. Beware: if you break ties in a different way, you are likely to get the wrong answer. You should report the sum of weighted completion times of the resulting schedule --- a positive integer --- in the box below.

ADVICE: If you get the wrong answer, try out some small test cases to debug your algorithm (and post your test cases to the discussion forum).

---

1 / 1 point

2.

For this problem, use the same data set as in the previous problem.

3/3 points (100%)

Your task now is to run the greedy algorithm that schedules jobs (optimally) in decreasing order of the ratio (weight/length). In this algorithm, it does not matter how you break ties. You should report the sum of weighted completion times of the resulting schedule --- a positive integer --- in the box below.

---

1 / 1 point

3.
In this programming problem you'll code up Prim's minimum spanning tree algorithm.

Download the text file below.

edges.txt

This file describes an undirected graph with integer edge costs. It has the format

[number_of_nodes] [number_of_edges]

[one_node_of_edge_1] [other_node_of_edge_1] [edge_1_cost]

[one_node_of_edge_2] [other_node_of_edge_2] [edge_2_cost]

...

For example, the third line of the file is "2 3 -8874", indicating that there is an edge connecting vertex #2 and vertex #3 that has cost -8874.

You should NOT assume that edge costs are positive, nor should you assume that they are distinct.

Your task is to run Prim's minimum spanning tree algorithm on this graph. You should report the overall cost of a minimum spanning tree --- an integer, which may or may not be negative --- in the box below.

IMPLEMENTATION NOTES: This graph is small enough that the straightforward O(mn) time implementation of Prim's algorithm should work fine. OPTIONAL: For those of you seeking an additional challenge, try implementing a heap-based version. The simpler approach, which should already give you a healthy speed-up, is to maintain relevant edges in a heap (with keys = edge costs). The superior approach stores the unprocessed vertices in the heap, as described in lecture. Note this requires a heap that supports deletions, and you'll probably need to maintain some kind of mapping between vertices and their positions in the heap.
