COT 5405 Programming Assignment 3

Chapter 4 (Greedy Algorithms), unsolved problem 13

Algorithm Design by Kleinberg and Tardos, 1st Edition

Problem statement

13. A small business—say, a photocopying service with a single large machine—faces the following scheduling problem. Each morning they get a set of jobs from customers. They want to do the jobs on their single machine in an order that keeps their customers happiest. Customer i's job will take t_i time to complete. Given a schedule (i.e., an ordering of the jobs), let C_i denote the finishing time of job i. For example, if job j is the first to be done, we would have $C_j = t_j$; and if job j is done right after job i, we would have $C_j = C_i + t_j$. Each customer i also has a given weight w_i that represents his or her importance to the business. The happiness of customer i is expected to be dependent on the finishing time of i's job. So the company decides that they want to order the jobs to minimize the weighted sum of the completion times, $\sum_{i=1}^{n} w_i C_i$.

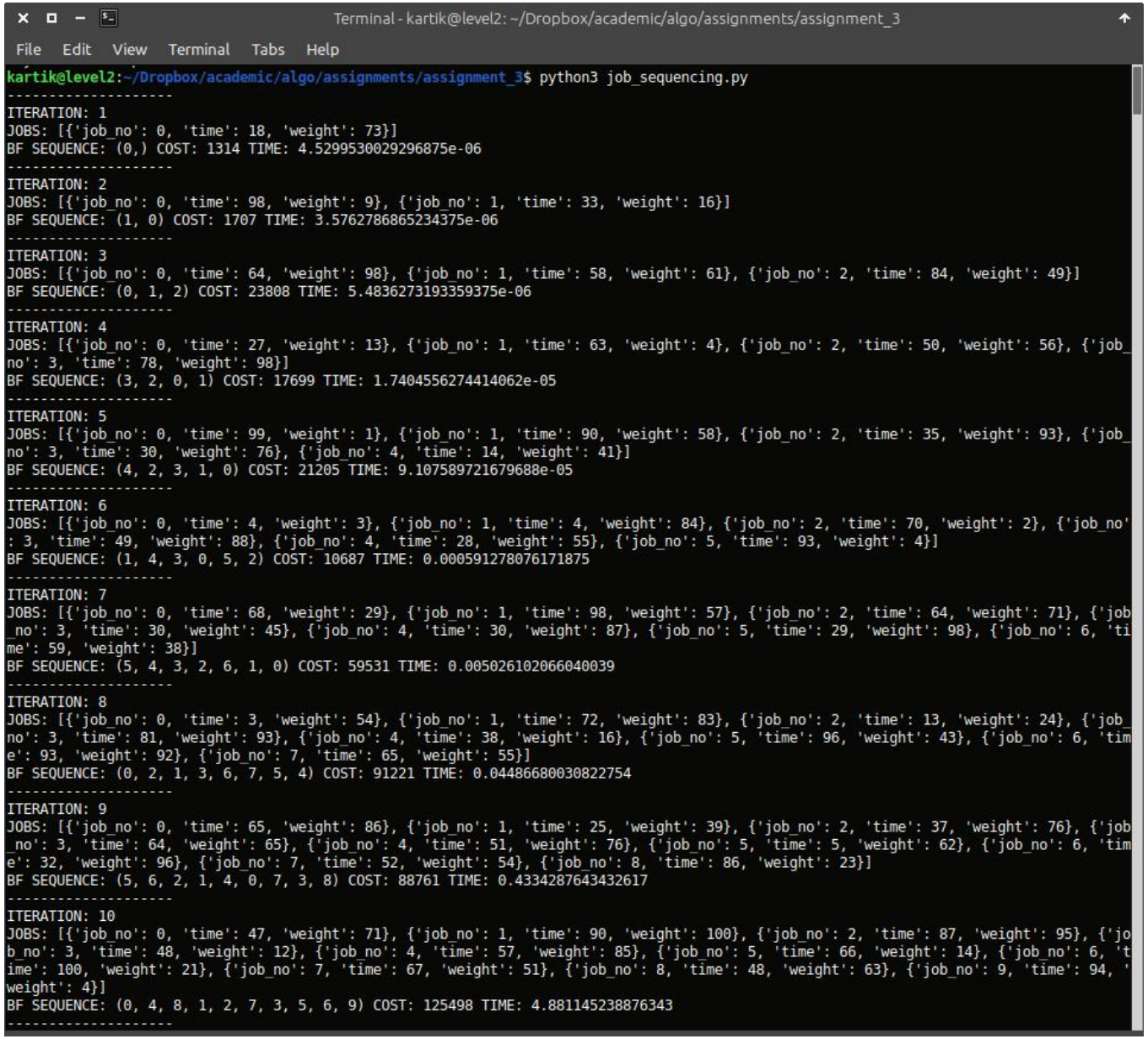
Design an efficient algorithm to solve this problem. That is, you are given a set of n jobs with a processing time t_i and a weight w_i for each job. You want to order the jobs so as to minimize the weighted sum of the completion times, $\sum_{i=1}^{n} w_i C_i$.

Example. Suppose there are two jobs: the first takes time $t_1 = 1$ and has weight $w_1 = 10$, while the second job takes time $t_2 = 3$ and has weight

$w_2 = 2$. Then doing job 1 first would yield a weighted completion time of $10 \cdot 1 + 2 \cdot 4 = 18$, while doing the second job first would yield the larger weighted completion time of $10 \cdot 4 + 2 \cdot 3 = 46$.

Brute Force solution

- Data: 10 test cases, each with k jobs (with random t_k and w_k) for test case k.
- Implementation:
  - Input: k jobs.
  - Generate k! possible sequences.
  - Calculate cost for each sequence.
  - Return sequence with minimum cost.

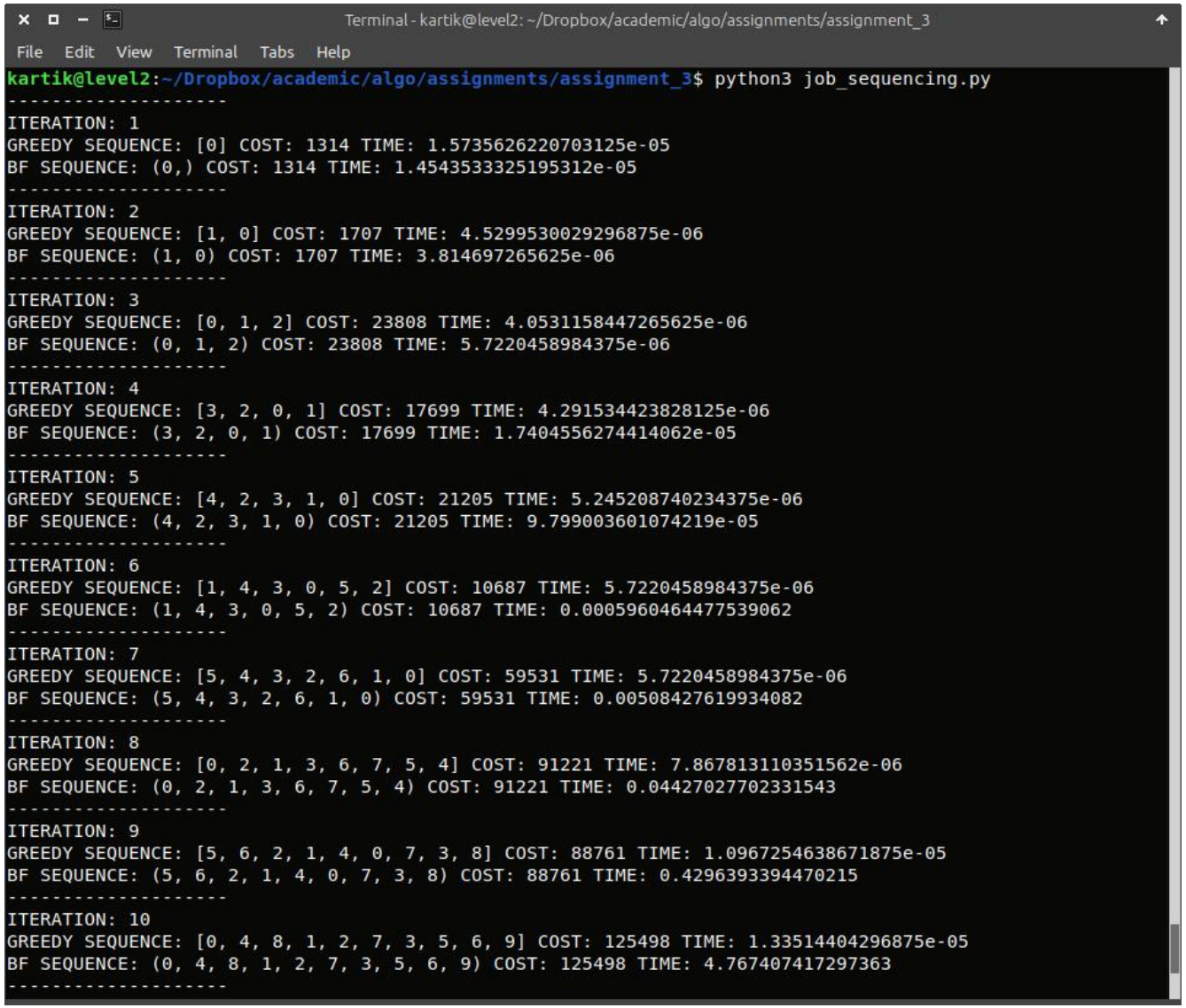


Greedy Solution Pseudocode

- Objective: Minimize $\sum w_i C_i$ where C_i is the finish time of job i.
- Greedy thinking: Short jobs (small t_i values) with high priority (large w_i values) are executed first.
- Pseudo code:
  - For every job in set, store $r_i = t_i / w_i$.
  - Sort the list ***r*** keeping its corresponding job number ***i*** in track.
  - The obtained sequence is the optimal sequence that returns the least cost possible.
- Code: https://github.com/kjain-ucf/misc/algorithms_assignment_3/job_sequencing.py

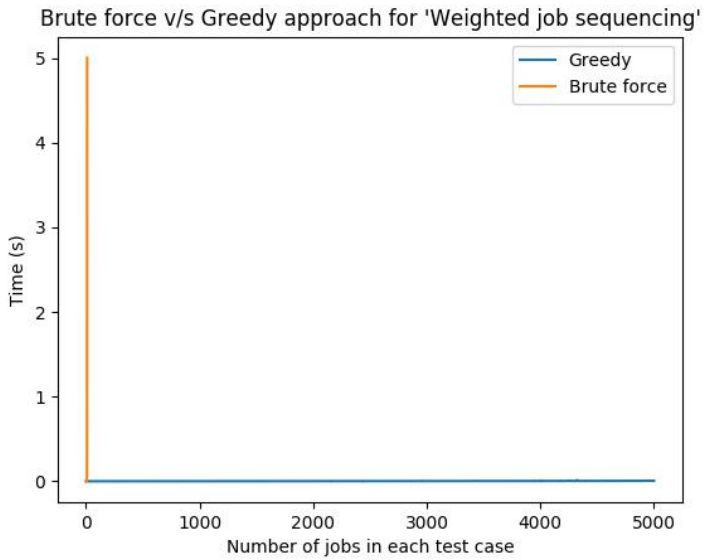
Validation (10 tests)

- Note: Used a seed to generate the same random job sets to replicate results. Comment line number 9 to get different results for every run.
- Note: Same sequences are obtained with both the approaches as expected



Runtime graphs

X linear scale, Y linear scale



X linear scale, Y log scale

X log scale, Y log scale