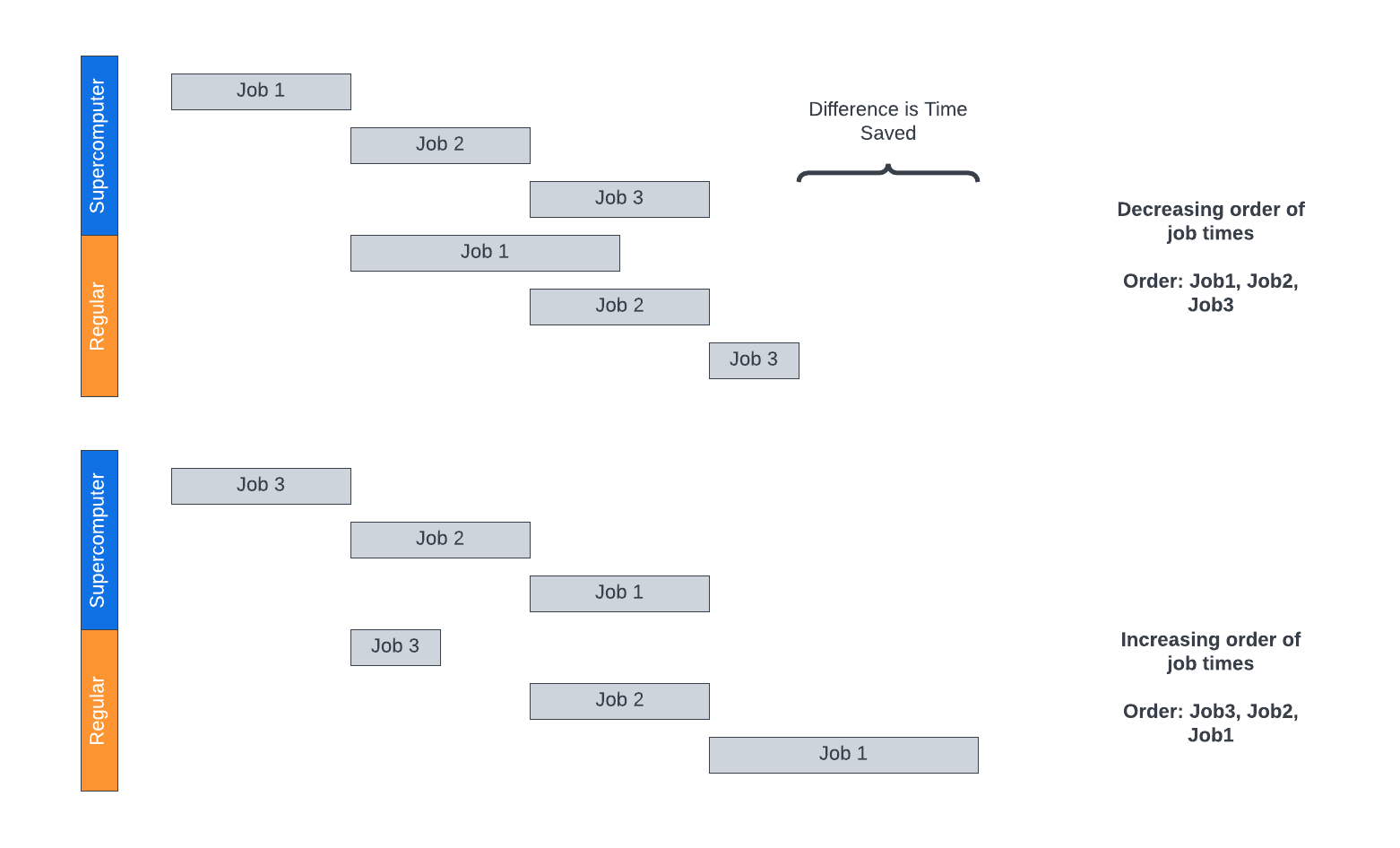
**Problem 1 Super or Normal (10 points)**

* You have one supercomputer and *n* normal computers on which you need to run *n* jobs.
* Each job*i* first spends *si* time on the supercomputer and then *ni*time on the normal computer.
* A job can only start running on any of the normal computers after it has finished on the supercomputer. However, as soon as any job finishes on the supercomputer, it can immediately start on one of the free normal computers. The goal is to finish running all the jobs as soon as possible.
* Note that since there is only one supercomputer, you’ll always have to wait  time so that the jobs finish running on the supercomputer. However, you can optimize when you run the jobs on the normal computers to try to finish running all the jobs as soon as possible.
* Show the following schedule is optimal: execute jobs after sorting them in decreasing order by *ni*. Note: Include any assumptions on this problem to clarify the justification. But do not allow your assumption(s) to contradict what is in the problem statement.

Answer:

Our objective here is to show a schedule is optimal when we execute jobs after sorting them in decreasing order by ni in which ni represents the time on the normal computer, which essentially marks the time in which the job has its earliest completion.



Since all jobs must spend si time on the supercomputer first, we are only able to optimize the algorithm when it comes to the ni time spent on the normal computers. The first step here is to sort the jobs in order of decreasing ni time, allowing the jobs with the longest durations to go first, and the ones with the shortest durations to go last. The idea here is that more jobs run in parallel allows us to complete more jobs at the same time as demonstrated in the figure above. The greedy nature of this algorithm is to use the jobs time as our metric and select the longest jobs first.

We assume that the jobs once run in the supercomputer, move immediately to the normal computers without any downtime or additional delays.

<https://www.chegg.com/homework-help/questions-and-answers/one-supercomputer-n-normal-computers-need-run-n-jobs-job-first-spends-time-supercomputer-t-q103327491>

<http://mypathtothe4.blogspot.com/2013/03/greedy-algorithm-example.html>

<http://pages.cpsc.ucalgary.ca/~verwaal/courses/413old/ProblemSets/set4/PS4_solution.pdf>

References:

[1] https://northeastern.instructure.com/courses/117409/pages/module-6-6-dot-2-interval-scheduling-2?module\_item\_id=7833143

[2] https://en.wikipedia.org/wiki/Interval\_scheduling#:~:text=Interval%20scheduling%20is%20a%20class,%2C%20

scheduled%20on%20some%20resource).

[3] Introduction to Algorithms, Cormen, Third Edition. (CLRS)

**Problem 2 Skiing Agency (10 pts)**

* A ski rental agency has n pairs of skis, where the height of the ith pair of skis is si.
* There are n skiers who wish to rent skis, where the height of the ith skier is hi.
* Ideally, each skier should obtain a pair of skis whose height matches her/his own height as closely as possible.
* We would like to assign skis to skiers so that the sum of the absolute differences of the heights of each skier and her/his skis is minimized. Design a greedy algorithm for the problem.
* **Prove** the correctness of your algorithm. Note: Include any assumptions on this problem to clarify your proof. But do not allow your assumption(s) to contradict what is in the problem statement.
* (Hint: Start with two skis and two skiers. How would you match them? Continue to three skis and three skiers, and identify a strategy.)

Proof
using
method of contradiction
n = 3
All
E.gi-
y no. of skis
S[n] = {2,5, u} .height of skis
h (n] = {9, 14,5} height of

<https://www.chegg.com/homework-help/questions-and-answers/ski-rental-agency-n-pairs-skis-height-ith-pair-skis-si--n-skiers-wish-rent-skis-height-th--q93300945>

<https://www.chegg.com/homework-help/questions-and-answers/written-explanation-included-along-pseudocode-algorithm-q60114752>

**Problem 3 Negative Edges (10 pts)**

Consider a directed graph in which the only negative edges are those that leave *s*; all other edges are positive. Can Dijkstra’s algorithm, started at *s,* fail on such a graph? Prove your answer.

<http://web.stanford.edu/class/archive/cs/cs161/cs161.1182/Lectures/Lecture11/CS161Lecture11.pdf>

<https://www.chegg.com/homework-help/questions-and-answers/consider-directed-graph-negative-edges-leave-s-edges-positive-dijkstra-s-algorithm-started-q366092?trackid=fd848693faec&strackid=b3aa47178a51>