# Algorithms Lab 7

The in-lab problems are to be solved during the lab time. Work with your team. You do not need to turn in your answers. Ask me for feedback.

The homework problem set is due in one week. Work with your team, but write your solutions individually. List the people with whom you discussed the problems clearly on the first page.

**Grading:** As we have started to see more "interesting" problems besides searching and sorting, a discussion of what is expected of a good answer is in order. Use these guildeline for both assignments and exams[1]

The bottom line is that a solution is evaluated forproblem), but also for:

- correctness (does your algorithm solve the problem)
- analysis (specify its space and time complexity requirements)
- style (should look professional, neat, that means not messy, imagine this is your first assignment at Google and you explain it to your supervisor; ;eave space for us to write feedback)
- clarity (shoudl be easy to understand)
- conciseness (avoid unnecessary explanations)

| | |
|---|---|
| 10 points | The solution is clear and correct, clear, neat and concise. |
| 8-9 points | The solution contains a few mistakes, but they are mostly arithmetic or of little significance to the overall argument. |
| 6-7 points | The solution hits on the main points, but has at least one logical gap. |
| 4-5 points | The solution contains several logical mistakes, but parts of it are salvageable. |
| 2-3 points | The solution is just plain wrong. |
| <2 points | No attempt is made at solving the problem. |

[1]The guildelines have been adapted from Williams, Swarthmore, Carleton, MIT

## In lab

The weighted interval scheduling problem (handout from class).

## Homework

1. You've decided to become a ski bum, and hooked up with Sugarloaf Ski Resort. They'll let you ski for free all winter, in exchange for helping their ski rental shop with an algorithm to assign skis to skiers.

   Ideally, each skier should obtain a pair of skis whose heigth matches his or her own height exactly. Unfortunately, this is generally not possible. We define the disparity between a skier and his/her skis as the absolute value of the difference between the height of the skier and the height of the skis. The objective is to find an assignment of skis to skiers that minimizes the sum of the disparities.

   (a) First, let's assume that there are $n$ skiers and $n$ skis. Consider the following algorithm: consider all possible assignments of skis to skiers; for each one, compute the sum of the disparities; select the one that minimizes the total disparity. How much time would this algorithm take on a 1GHz computer, if there were 50 skiers and 50 skis?

   (b) One day, while waiting for the lift, you make an interesting discovery: if we have a short person and a tall person, it would never be better to give to the shorter person a taller pair of skis than were given to the taller person. Prove that this is always true.

   (c) Describe a greedy algorithm to assign the skis to skiers, and argue why this algorithm is correct. What is the time complexity of your algorithm?

   (d) Your next task is to design an efficient dynamic programming algorithm for the more general case where there are $m$ skiers and $n$ pairs of skis ($n \geq m$).

   Here is some notation that may help you. Sort the skiers and skis by increasing height. Let $h_i$ denote the height of the $i$th skier in sorted order, and $s_j$ denote the height of the $j$th pair of skis in sorted order. Let OPT$(i, j)$ be the optimal cost (disparity) for matching the first $i$ skiers with skis from the set $\{1, 2, ..., j\}$. The solution we seek is simply OPT(m, n). Define OPT$(i, j)$ recursively.

   (f) Briefly describe how your algorithm can be modified to allow you to find an actual optimal assignment (rather than just the cost) of skis to skiers. How does this affect the running time of your algorithm?

   Note: This problem was adapted from Harvey Mudd College.

2. You have been hired to design algorithms for optimizing system performance. Your input is an array $J[1..n]$ where $J[i]$ or $J_i$ represents the running time of job $i$; jobs do not have specific start and end times, but they can be started at any time (this is a different scenario than in interval scheduling). The running times are integers.

Generally speaking, your task is to find an optimal load balance of these tasks over two processors.

Design an algorithm for determining whether there is a subset $S$ in $J$ such that the running time of the elements in $S$ sum up precisely to the same amount as the sum of the elements not in $S$; more formally, $\sum_{J_i \in S} J_i = \sum_{J_i \in J-S} J_i$. The algorithm should run in time $O(n \cdot N)$, where $N$ is the sum of the running times of the $n$ jobs.

EXTRA CEDIT Same setup as above, but now you want to find a partition of $J$ into two sets, such that jobs in one set will be executed on one processor and tasks in the other set will be executed on the other processor, in parallel. The partition has to be such that it minimizes the total alapsed time to complete all the tasks. Describe an algorithm to come up with such a partition.