The first page of your homework submission must be a cover sheet answering the following questions. Do not leave it until the last minute; it's fine to fill out the cover sheet before you have completely finished the assignment. Assignments submitted without a cover sheet, or with a cover sheet obviously dashed off without much thought at the last minute, will not be graded.

- How many hours would you estimate that you spent on this assignment?

- Explain (in one or two sentences) one thing you learned through doing this assignment.

- What is one thing you think you need to review or study more? What do you plan to do about it?

Question 1 (K&T 6.4, 10 points)**.** You are running a consulting business, with clients on both east and west coasts. Each month, you can run your business from an office in either New York or San Fransisco. In month i, you will incur an *operating cost* of N_i if you run the business out of New York; you will incur an operating cost of S_i if you run the business out of San Fransisco (the costs can change each month depending on the distribution of client demands).

However, if you run the business out of one city in month i, and then out of the other city in month $i + 1$, then you incur a fixed *moving cost* of M to switch cities.

Given a sequence of n months, a *plan* is a sequence of n locations—each one equal to either NY or SF—such that the ith location indicates the city in which you will be based in the ith month. The *cost* of a plan is the sum of the operating costs for each of the n months, plus a moving cost of M for each time you switch cities. The plan can begin in either city.

Given a value for the moving cost M, and sequences of operating costs $N_1, \ldots, N_n$ and $S_1, \ldots, S_n$, the problem is to find a plan with minimum total cost.

(a) Show that the following greedy algorithm does not correctly solve this problem, by giving an instance on which it does not return the correct answer.

Algorithm 1: GreedyPlan

```
for i ← 1 to n do
    if N_i < S_i then
        Output "NY in month i"
    else
        Output "SF in month i"
```

In your example, say what the correct answer is and also what the above algorithm finds.

(b) Give an efficient algorithm that takes values for n, M, and sequences of operating costs $N_1, \ldots, N_n$ and $S_1, \ldots, S_n$, and returns the *cost* of an optimal plan.

(c) Explain how to extend your algorithm to also find the optimal plan itself, not just its cost.

Question 2 (5 points)**.** You are building a toy train track out of a sequence of straight pieces laid end-to-end. Some pieces are one unit long, and some are three units long. How many different ways are there to build a track that is five hundred units long?

For example, there are 9 different ways to build a track that is 7 units long, as illustrated below.

Figure 1: The nine ways to build a length-7 train track

2

2

Question 3 (K&T 6.3, 10 points)**.** Let $G = (V, E)$ be a directed, unweighted graph with nodes $v_1, \dots, v_n$. We say that G is an *ordered graph* if it has the following properties:

(i) Each edge goes from a node with a lower index to a node with a higher index. That is, every directed edge has the form (v_i, v_j) with $i < j$.

(ii) Every node other than v_n has at least one outgoing edge.

Given an ordered graph G, we want to find the *longest* path from v_1 to v_n.

(a) Consider the following greedy algorithm.

Algorithm 2: GreedyLongestPath

```
w ← v_1
L ← 0
while w ≠ v_n do
    Choose the outgoing edge (w, v_j) with the smallest j
    w ← v_j
    Increment L
return L
```

Show that this algorithm does *not* correctly solve the problem, by giving an example of an ordered graph for which it does not return the correct answer. Be sure to explain what the correct answer is and what incorrect answer is returned by the algorithm.

(b) Give an efficient algorithm that takes an ordered graph G and returns the length of the longest path from v_1 to v_n. Justify its correctness and analyze its time complexity.

3(b)

(c) Explain how to modify your algorithm so that it can also be used to recover the longest path itself, rather than only its length.