SI 335 Spring 2013: Problem Set 3

Your name:

Due: Wednesday, April 3

Instructions: Review the course honor policy: informal *discussions* are permitted, but no *written collaboration* of any kind..

This cover sheet must be the front page of what you hand in. Fill out the left column in the table to the right before you hand it in.

Use separate paper for your solutions and make sure they are submitted in order — please!.

Comments or suggestions about this problem set:

Comments or suggestions about the course so far:

I had discussions with:

Citations (be specific about websites):

Grading rubric:

- **5**: Solution is completely correct, concisely presented, and neatly written.
- **4**: The solution is mostly correct, but one or two minor details were missed, or the presentation could be more concise.
- **3**: The main idea is correct, but there are some significant mistakes. The presentation is somewhat sloppy or confused.
- **2**: A complete effort was made, but the result is mostly incorrect. There may be some basic misunderstandings of the topic or the problem.
- **1**: The beginning of an attempt was made, but the work is clearly incomplete.
- **0**: Not submitted.

Problem	Self-assessment	Final assessment
1		
2		
3		

1 Shoe Matching

Scenario:

Imagine the following scenario. You and a group of friends all decide to jump in the lake for a swim. Naturally, you all leave your shoes up on the beach. And for some reason, you all have exactly the same looking shoes.

While you are in the water, a practical joker removes all the labels from the shoes and mixes them all around. So when you and your friends get out of the water, you are met with a big pile of shoes and you have no idea whose is whose. All you can do is start trying them on until everyone finds their own pair.

Knowing that you are an algorithms expert, your friends turn to you for the quickest way to sort all this out.

Specifics:

You *must* solve the problem under the following assumptions. No "side-channel" solutions like saying that you all find the practical joker, take his money, and buy yourselves new sneakers.

1) There are n friends and n pairs of shoes.
2) You can immediately tell the difference between a left shoe and a right shoe.
3) There is *absolutely no way* to tell the shoes for the same foot apart except by trying them on.
4) An individual can try on one shoe at a time.
5) After trying on a shoe, the individual knows whether it was too small, too big, or just right.
6) For each of the n friends, there is exactly one shoe in the pile (no more, no less) that fits each of their feet perfectly.
7) No two shoes are exactly the same, and no two feet are exactly the same.

Tasks

a) Devise an algorithm to match everyone with their shoes. Describe your algorithm in words, or using pseudocode, or both. It is *your job* to describe your algorithm as simply and as clearly as possible. Remember that the only thing you can do is have an individual try on a shoe, and after trying it on they know if is their shoe (just right), or if it is too small, or if it is too big.

b) Analyze your algorithm to determine the number of shoes that must be tried on, in total, in the worst case. Come up with a big-Theta bound for this worst-case cost.

Hints and Comments

- Think about the sorting algorithms we have studied to get you started.
- It is much more important to have an algorithm that is correct, clearly explained, and correctly analyzed, than to have an algorithm that is blazingly fast.
- HOWEVER, more points will be awarded to faster algorithms. You should try to make the big-Theta cost of your algorithm as small as possible.
- If you want to be sure you have the fastest possible algorithm, try proving that it is *asymptotically optimal*!

2 Road Trip Planning

Scenario:

You are planning a road trip that will follow a certain route and stop in a bunch of towns along the way. You have determined the exact sequence of towns (each of which has a place to sleep up for the night), and the distances between each consecutive pair of towns.

Given a certain number of days alloted for your road trip, you want to break up the legs of your journey so that they are as balanced as possible. Essentially, you'd like to drive as little as possible each day, while still finishing your trip in the required number of days. Mathematically, you are going to find which way of splitting up the days will minimize the sum of the squares of the distances travelled each day.

For example, if your journey has 4 legs of 2, 5, 3, and 3 miles (in that order), and you have only 2 days for the trip, you have the following options:

```
Option A: 0 miles day 1, (2+5+3+3) miles day 2
  Sum of squares: 0^2 + 13^2 = 169

Option B: 2 miles day 1, (5+3+3) miles day 2
  Sum of squares: 2^2 + 11^2 = 125

Option C: (2+5) miles day 1, (3+3) miles day 2
  Sum of squares: 7^2 + 6^2 = 85

Option D: (2+5+3) miles day 1, 3 miles day 2
  Sum of squares: 10^2 + 3^2 = 109

Option E: (2+5+3+3) miles day 1, 0 miles day 2
  Sum of squares: 13^2 + 0^2 = 169
```

So in this example, the best choice would be to take the first two legs of the journey on day 1, and the last two legs on day 2.

Specifics

Your road trip itinerary has n legs, each represented by a numeric distance in miles. You have d days to complete the road trip.

So the input to your algorithm is a list of numbers of length n: $[x_1, x_2, \ldots, x_n]$

and the output is a d stopping points (each is an integer from 0 up to n): $[s_1, s_2, \ldots, s_d]$

such that the sum of squares

$$\left(x_1 + \cdots + x_{s_1}\right)^2 \quad + \quad \left(x_{s_1+1} + \cdots + s_{s_2}\right)^2 \quad + \quad \cdots \quad + \quad \left(x_{s_{d-1}+1} + \cdots + x_n\right)^2$$

is as small as possible.

Tasks

a) Devise an algorithm to determine the optimal road trip. Describe your algorithm in words, or using pseudocode, or both. It is *your job* to describe your algorithm as simply and as clearly as possible.

b) Analyze your algorithm and determine a big-Theta bound for the worst-case cost in terms of n and d.

Hints and Comments

- The most important thing is that your algorithm is CORRECT and clearly explained. Correctness in this case means that your algorithm *always* finds the optimal solution. This means that, if you think of a way of speeding up the algorithm, but you're not sure if it will still give the absolutely best solution every time, *don't do it*!
- Of course, more points will be awarded to faster algorithms. You should try to make your algorithm as fast as possible in every case, particularly the worst case.
- Planning this trip is sort of like putting parentheses around the distances, grouping together each day. Try to solve this problem in a similar way to what we did for the matrix chain multiplication problem in class.
- Lao-tzu said, "The journey of a thousand miles begins with a single step". Similarly, the road trip of n legs begins with a single day. You might want to structure your algorithm recursively by (at the top level) figuring out the optimal number of legs to take on day 1.
- I am not sure what the asymptotically optimal algorithm for this problem is. However, I assure you that it can be done in polynomial-time (*not* exponential) in terms of n and d. Do your best!

3 Selection (up to 5% BONUS)

Consider the following variant on the selection problem: Given a list A of n elements, and an integer k, find the kth largest DISTINCT element in A, counting from 0. This is like the original selection problem, but ignoring duplicated elements.

Show that, at least in the comparison model, there can't possibly be a $O(n)$-time algorithm for this modified selection problem that ignores duplicate elements.