Isai Lopez Rodas
1605542 ilopezro@ucsc.edu

CMPS 102 — Spring 2020 – Homework 3

Five problems, due Friday May 1 (on canvas) and Monday May 11 (11:59 PM) on Gradescope.
Due class size, it is likely that only a subset of the problems will be graded.
Ver 0405.2

Before you begin the assignment, please read the following carefully.

- **Read the Homework Guidelines**.
- Every part of each question begins on a new page. Do not change this.
- This does not mean that you should write a full page for every question. Your answers should be short and precise. Lengthy and wordy answers will lose points.
- Do not change the format of this document. Simply type your answers as directed.
- You are **not** allowed to work in teams.

I have read and agree to the collaboration policy. – Isai Lopez Rodas, ilopezro@ucsc.edu
Collaborators: None

---

1. (1 pt) Take the pre-quiz on Canvas by 11 PM on Friday May 1st. This will be short quiz designed to test if you have read and understood the other problems. The quiz should be available by Thursday afternoon.

2. (6 pts): Chapter 4, Problem 2. Two true or false statements, with justification (3 pts each).

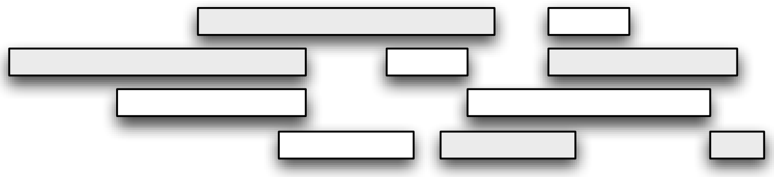
   Solution. Problem 2.

   - This would be **true** because Kruskal's Algorithm would sort them in the same exact way, but the subsets of the edges in the MST would be the same because this algorithm only takes a look at the order of the costs rather than the actual given values.
   - This would be **false** because if we were given edges in a given graph with edges (s, v), (v, t), and (s, t) with values 2, 5, and 6 then we know that (s, t) is the shortest path because it has value 6; however, if we were to square the values of each edge we would have the following values: (s, v), (v, t), and (s, t) with values 4, 25, and 36. Now, the shortest path would be (s, v) and (v, t) with a value of 29.

I have read and agree to the collaboration policy. – Isai Lopez Rodas, ilopezro@ucsc.edu
Collaborators: None

---

3. (3 + 5 + 5 + 2 = 15 pts). The Menlo Park Surgical Hospital admitted a patient, Mr. Banks, who was in a car accident and is still in critical condition and needs continuous monitoring over the next 48 hours (i.e. all real-valued times between 0 and 48). At any given time only one nurse needs to be on call for the patient though. For this we have an availability interval for each of the n nurses, which is a time from which they become available, a_i, to the time they must leave for other commitments, b_i The a_i and b_i are real valued, and you may assume that for each i, $0 \le a_i < b_i \le 48$. You need to devise an algorithm that determines a set of nurses to use to cover the next 48 hours of Mr. Banks' stay while having the minimum number of nurses disrupt their normal routine to be on call for him (or report that some time cannot be covered) A nurse leaving at the same time as another arrives is acceptable. Following is a picture of what the intervals for the nurses might look like. The darker bars correspond to a set of 5 of the 10 nurses who can cover the entire duration. (Notice, though, that 4 nurses would have sufficed.)



Your efficient **greedy** algorithm should take as input a list of pairs of times (a_i, b_i) for $i = 1$ to n and the end time T.

Continued on next page.

(a) Consider the greedy algorithm that selects nurses by repeatedly choosing the nurse who will be there for the longest time among the periods not covered by previously selected nurses. For example,if times 2 through 8 are covered by already selected nurses a another nurses interval is (5,10), then adding that nurse would only provide additional coverage form 8 to 10, or 2 units of time. Give an example showing that this algorithm does **not** always find the smallest set of nurses.

Solution.

This algorithm is not the most efficient because it does not choose the smallest set of available nurses. If we have three nurses with the following schedules:

- Nurse 1: 0-24
- Nurse 2: 10-45
- Nurse 3: 24-48

The algorithm will choose nurse 2 first since it has the longest available schedule. Nurse 2 is able to cover 35 hours while nurse 1 and 2 can only cover 24 hours. This is a problem because you need someone to cover the first 10 hours and the last 3 hours. Therefore the algorithm will **HAVE** to schedule nurses 1 and 2. Therefore the algorithm has chosen all three nurses when there is a more optimal solution, which is to choose nurses 1 and 3 to cover the shifts. Therefore, this algorithm is not correct.

(b) Present an algorithm that always outputs a smallest subset of nurses that can cover the entire 48 hours or report that no such subset exists.

Solution.

i. For all a_i, sort in ascending order.

   A. If there are two nurses who have the same a_i, sort b_i in decending order.

ii. After this sorting is done, we not have a list of nurses and their schedules where nurse 1 is (ideally) supposed to start off at the 0th hour.

iii. If $a_0 > 0$, there is no solution.

iv. Else if $a_0 = 0$, then the first scheduled nurse is (a_0, b_0)

v. Starting at $i = 1$, look for a a_i that fits the following inequality: $a_i \leq b_0$. Once you find a nurse that fits the following, you are guaranteed to have a nurse$_1$ either start before or when nurse$_0$ ends. We will keep a counter for the index x where b_x is the biggest.

   A. If you've got a situation when $a_1 > b_0$, you have no solution because that means that no nurse is able to start when nurse$_0$'s shift ends.

   B. After an iteration finds such an $a_i > b_1$, we set a variable $y = i$.

vi. The next nurse that can cover the next shift then becomes: (a_y, b_y)

vii. Next loop would start at $i = y$. Line $v, v.A$, and $v.B$ would be repeated as neccesary.

viii. Iteration of this loop would end when b_i = 48.

(c) Prove that your algorithm is correct (i.e. alway finds a smallest possible subset of the nurses the cover the entire stay).

Solution.

We will begin with assuming we have a solution, s_1, produced by the algorithm above.

Base Case: Another solution exists besides the one provided by s_1
Therefore we know that $a_0 = 0$ is true because there must exist a nurse that covers that first hour. We also know that the $b_1 \geq b_i$. Therefor if we replace (a_i, b_i) with (a_1, b_1), we still have a valid solution, thus the greedy choice applies in this case.

Inductive Step: Assume there is another different solution that starts with (a_i, b_i). Therefore, we want to prove that:

$$a_i = a_0, b_i = b_0, a_{i+1} = a_y, b_{i+1} = b_y, \cdots, a_{i+j} = a_{y+j}, b_{i+j} = b_{y+j}$$

where

$$b_{y+j} > b_{y+j+1}$$

To satisfy the fact that there must be a valid nurse that can start at the scheduled end time for the previous nurse. The greedy algorithm will find the a value for b_{y+j} while acknowledging inequality: $a_{i+j} \leq b_{i+y+1}$. Therefore we can replace (a_i, b_i) with (a_{i+j}, b_{i+j}). Which procudes a valid solution according to the greedy choice application. Therefore, the algorithm is correct.

(d) State its running time with a brief one-to-three sentence justification.

Solution. Because we are sorting the nurses at the beginning of the algorithm, the sorting takes $\mathcal{O}(n \log n)$ time. The comparisons in the middle are only made once per element, which then produces an $\mathcal{O}(n)$ runningtime for the comparisons. Therefore, the total runningtime of the algorithm is $\mathcal{O}(n \log n)$.

I have read and agree to the collaboration policy. – Isai Lopez Rodas, ilopezro@ucsc.edu
Collaborators: None

---

4. (15 pts) A first grade teacher has a set of n books, say $\{1, 2, \ldots, n\}$, stored on a bookshelf in the classroom. Each book k has a popularity represented by the probability $p(k)$ of being wanted by a student (since these are probabilities they are non-negative and sum to 1). Unfortunately, the students are not sophisticated enough to do binary search, and so do a sequential search for the book from left to right. When s_1 is the first book on shelf, s_2 the second, and so on, and some book s_i is wanted, it will time i for the student to locate it. When all the books are on the shelf, the average search time is $\sum_{i=1}^{n} i \cdot p(s_i)$. The problem is to find (and prove correct) a greedy algorithm that that takes the $p(k)$ values and determines an order $s_1, s_2, \ldots, s_n$ for storing the books on the shelf that minimizes this average search time.

   (Hint: Use the structure of the problem or an exchange argument to prove your greedy algorithm's solution is optimal.)

   Continued on next page.

Algorithm.

Proof of correctness.

Big-O Analysis.

I have read and agree to the collaboration policy. – Isai Lopez, ilopezro@ucsc.edu
Collaborators: None

5. (15 pts) Is it a break-in?

The security staff for a high-tech company suspects that a notorious hacker is breaking into their network and stealing their trade secrets. The traffic on the network over the last couple of days is represented by a sequence of events $E = (e_1, e_2, \ldots, e_n)$ where each e_i is an integer event ID. These IDs can come from all sorts of applications, and the same event number can be seen multiple times. The notorious hacker has a predictable way to break in: he causes a particular sequence of $k < n$ events $H = (h_1, h_2, \ldots, h_k)$ to happen on the network to help him gain access. These events have to occur in that particular order, and some event numbers may be be repeated in H. Your task is to design and prove correct a greedy algorithm for determining if H is a subsequence of E and when the hacker could have broken in. Recall that H is a subsequence of E if H is embedded in E, i.e. some number (possibly 0) of the events in E can be deleted so that your are left with H. More formally, the k length sequence H is a subsequence of E if there is a sequence of k indices $i_1 < i_2 < \cdots < i_k$ such that each $h_j = e_{i_j}$.

For example, neither (2, 4, 5) nor (4, 2, 2, 5) are subsequences of (1, 9, 4, 1, 4, 2, 6, 5, 7), but (9,5) and (4, 4, 2, 5) are.

Design a greedy algorithm running in $O(n)$ time that determines if H is a subsequence of E and returns the first (lowest) ℓ such that H is a substring of $E_\ell = e_1, e_2, \ldots, e_\ell$ (i.e. E_ℓ is the shortest prefix of E that contains H as a substring). Prove that your algorithm is correct and briefly justify its running time.

To prove correctness, it may be helpful for your algorithm to compute the sequence of indices where the events in E match those in H.

Continued on next page.

Algorithm.

```
procedure ISBREAKIN($E, H$)
    $lenE \leftarrow len(E)$
    $lenH \leftarrow len(H)$
    $counter \leftarrow 0$
    for (i = 0); i<lenE and counter $<$ lenH; i++
    if(E[i] = H[counter])
    counter++;
    if (counter = i )
    return i
    return -1
end procedure
```

Proof of correctness.

When we have a sequence of evenets E_1 that is also in H the algorithm selects the first event in E_1 that matches with E_1. It will then proceed to that with subevent $e_2 \cdots e_i$. After, our algorithm will return the index in which E is a subsequence of H. Therefore, our algorithm's greedy choice is the optimal solution.

Big-O Analysis. In the worst case scenario, our algorithm will have to go through all events in E, which means that it has to run a total time of $\mathcal{O}(n)$.