EE360C: Algorithms
The University of Texas at Austin — Exam #1
Dr. Pedro Santacruz & Dr. Evdokia Nikolova — October 6, 2014

Name: **EID:**

Exam #1

Instructions. No calculators, laptops, or other devices are allowed. This exam is **closed book**, but you are allowed to use a **one-page** cheat sheet. You must submit your cheat sheet with the exam. Write your answers on the test pages. If you need scratch paper, use the back of the test pages, but indicate where your answers are. Write down your process for solving questions and intermediate answers that **may** earn you partial credit.

If you are unsure of the meaning of a specific test question, write down your assumptions and proceed to answer the question on that basis. **Questions about the meaning of an exam question will not be answered during the test.**

You have **75 minutes** to complete the exam. The maximum possible score is 100.

Some useful information:

Logarithms and Factorial:

$$\log(n!) = \Theta(n \log n)$$

Arithmetic Series:

$$\sum_{k=1}^{n} k = \frac{1}{2}n(n+1)$$

Sum of Squares:

$$\sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

Sum of Cubes:

$$\sum_{k=0}^{n} k^3 = \frac{n^2(n+1)^2}{4}$$

Geometric Series:

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1}$$

Infinite Geometric Series:

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$$

Problem 1: Graphs and Proofs [10 points]
Consider a mobile network of n devices. As the devices move around, they define a graph at any point in time as follows: there is a node representing each device, and there is an edge between device i and device j if the physical locations of i and j are no more than 500 meters apart. In designing a mobile network, we would like it to be the case that the network is *connected*, i.e., that all nodes are always *reachable* from all other nodes. Consider the following property: at all times, each device i is within 500 meters of at least $n/2$ of the other devices. Does this property by itself guarantee that the network is connected? Either prove that this is true or give a counter example.

Solution

Consider a graph G with the given properties. Suppose by way of contradiction that G is not connected. Let S be the nodes in its smallest connected component. Since there are at least two connected components (because the graph is not connected), we have $|S| \leq n/2$. Consider any node $u \in S$. Its neighbors must all live in S (given the definition of connected component), so its degree can be at most $|S| - 1 \leq n/2 - 1 < n/2$. This contradicts the statement that every node has degree at least $n/2$.

Problem 2: Asymptotic Notation [15 points]
Prove or disprove each of the following. You may use either the definitions of the asymptotic notations or the limit method. Assume all functions below are positive everywhere.

(a) $f(n) = O(g(n))$ implies $g(n) = O(f(n))$

Solution

False. Counter example: $f(n) = n$, $g(n) = n^2$.

(b) $f(n) + g(n) = \Omega(\min(f(n), g(n)))$.

Solution

True. Use the limit theorem. $\lim_{n\to\infty} \frac{f(n)+g(n)}{\min(f(n),g(n))} = \infty$ or c

(c) $f(n) = O((f(n))^2)$

Solution

False. Counter example: $f(n) = 1/n$

Problem 3: Stable Marriage [30 points]
Gale and Shapley published their paper on the stable marriage problem in 1962; but a version of their algorithm had already been in use for ten years by the National Resident Matching Program, for the problem of assigning medical residents to hospitals.

Basically, the situation was the following. There were m hospitals, each with a certain number of available positions for hiring residents. There were n medical students graduating in a given year, each interested in joining one of the hospitals. Each hospital had a ranking of the students in order of preference, and each student had a ranking of the hospitals in order of preference. We will assume that there were more students graduating than there were slots available in the m hospitals.

The interest, naturally, was in finding a way of assigning each student to at most one hospital, in such a way that all available positions in all hospitals were filled. (Since we are assuming a surplus of students, there would be some students who do not get assigned to any hospital.)

We say that an assignment of students to hospitals is *stable* if neither of the following situations arises:

- First type of instability: There are students s and s', and a hospital h so that:
  - s is assigned to h,
  - s' is assigned to no hospital, and
  - h prefers s' to s.
- Second type of instability: There are students s and s', and hospitals h and h' so that:
  - s is assigned to h,
  - s' is assigned to h',
  - h prefers s' to s, and
  - s' prefers h to h'.

So we basically have the stable marriage problem except that (i) hospitals generally want more than one resident, and (ii) there is a surplus of medical students.

(a) (15 points) Give an efficient algorithm to find a stable assignment.

(b) (15 points) Show that there is always a stable assignment of students to hospitals.

Please use the next page to write your answer

Problem 3: Stable Marriage

(a) (15 points) Give an efficient algorithm to find a stable assignment.

Solution

The algorithm is very similar to the basic Gale-Shapley algorithm from the text. At any point in time, a student is either "committed" to a hospital or "free." A hospital either has available positions, or it is "full." The algorithm is as follows:

```
while some hospital $h_i$ has available positions do
  $h_i$ offers a position to the next student $s_j$ on its preference list
  if $s_j$ is free then
    $s_j$ accepts the offer
  else
    ($s_j$ is already committed to a hospital $h_k$)
    if $s_j$ prefers $h_k$ to $h_i$ then
      $s_j$ remains committed to $h_k$
    else
      $s_j$ becomes committed to $h_i$
      the number of available positions at $h_k$ increases by one
      the number of available positions at $h_i$ decreases by one
    end if
  end if
end while
```

The algorithm terminates in $O(mn)$ steps because each hospital offers a position to a student at most once, and in each iteration some hospital offers a position to some student.

Suppose there are $p_i > 0$ positions available at hospital h_i. The algorithm terminates with an assignment in which all available positions are filled, because any hospital that did not fill all its positions must have offered one to every student; but then, all these students would be committed to some hospital, which contradicts our assumption that $\sum_{i=1}^{m} p_i < n$.

(b) (15 points) Show that there is always a stable assignment of students to hospitals.

Solution

- For the first kind of instability, suppose there are students s and s', and a hospital h as above. If h prefers s' to s, then h would have offered a position to s' before it offered one to s; from then on, s' would have a position at some hospital, and hence would not be free at the end – a contradiction.
- For the second kind of instability, suppose that (h_i, s_j) is a pair that causes instability. Then h_i must have offered a position to s_j, for otherwise it has p_i residents all of whom it prefers to s_j. Moreover, s_j must have rejected h_i in favor of some h_k which he/she preferred; and s_j must therefore be committed to some h_l (possibly from h_k) which he/she also prefers to h_i.

Problem 4: Heaps [25 points]
A d-**ary heap** is like a binary heap, but non-leaf nodes have d children instead of 2 children.

(a) (5 points) How would you represent a d-ary heap using an array?

Solution
The root resides in $A[1]$, its d children reside in $A[2]$ through $A[d+1]$, their children reside in order in $A[d+2]$ through $A[d^2+d+1]$, and so on.

(b) (5 points) What is the height of a d-ary heap of n elements in terms of n and d?

Solution
Since each node has d children, the height of a d-ary heap with n nodes is $\Theta(\log_d n)$.

(c) (10 points) Give an efficient implementation of an operation that extracts the minimum from a d-ary min-heap. (That is, after extracting the minimum element, you must return a valid min-heap.)

Solution
It's the same as for a binary heap. Take the minimum element out of the root. Then take the last element and put it as the root and call heapify down. You have to adjust heapify to find the min of d children instead of the min of 2 children, but other than that it's the same.

(d) (5 points) What is the running time of your new algorithm? Justify your answer.

Solution
Heapify down may have to float the old value all the way down the tree, but at every level, we have to do d comparisons (instead of the old constant 2). So the running time is $O(d \log_d n)$.

Problem 5: Algorithm Design [20 points]
Design a $\Theta(n \log n)$ time algorithm that, given a set S of n integers and another integer z can determine if S contains two numbers x and y that sum to z. Prove the correctness of your algorithm and analyze its worst case running time. (Namely, prove that the running time is what you claim it is.)

Solution

Sort S. This takes $\Theta(n \log n)$ time. Create a second set S' that contains an element $(z - x_i)$ for each element $x_i \in S$. This takes $\Theta(n)$ time. For each element $y_i \in S'$, search for y_i in S. This can be done with a binary search that takes $\Theta(\log n)$ time for each of the n elements, for a total time of $\Theta(n \log n)$ time. This algorithm is correct. It treats every element of S as a candidate x (the second step) and searches for $z - x$ in S. If $z - x$ (i.e., y) exists, the algorithm will find it. Since it tries this for anything that could be x, the algorithm is guaranteed to find a suitable x and y if they exist.

Scratch Page