CS515: Analysis of Algorithms, Fall 2019 – Midterm

November 5, 2019

- Solve **at most 4** problems of the following 5 problems.
- *I don't know policy:* you may write "I don't know" *and nothing else* to answer a question and receive 25 percent of the total points for that problem whereas a completely wrong answer will receive zero.
- You can use algorithms in class or in the assignments as black boxes.

problem 1. Fibonacci sequence is defined recursively. The first two numbers of the sequence are ones. Then, each number in the sequence is the sum of its previous two numbers. From the definition we obtain the following recursive algorithm for computing the nth number of the sequence.

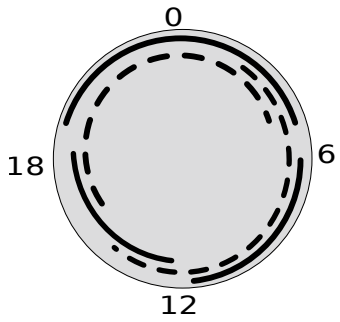
```
FIBONACCI(n)
  if n < 3 then
    return 1
  else
    return FIBONACCI(n − 1) + FIBONACCI(n − 2)
  end if
```

(a) Write a recursive relation for the running time of FIBONACCI?

(b) Show that the running time FIBONACCI is at least $2^{n/2}$. (comment: you can assume that the running time of FIBONACCI(k) is at least the running time of FIBONACCI($k-1$), for all k)

(c) Use memoization or iterative dynamic programming to make FIBONACCI faster. What is the running time of your new algorithm?

Problem 2. We have n jobs that should be repeatedly done every day. Each job has a start time and finish time; a job can start before midnight and finish after midnight. We have only one computer for the jobs, and we want to choose as many jobs as possible to perform everyday. Note, that if we choose a job we should do it everyday. Design an algorithm to choose as many jobs as possible. The following figure shows an example of 5 jobs. The three shown by solid lines can be done by one computer.



Problem 3. Alice has two sorted arrays $A[1, \ldots, n]$, $B[1, \ldots, n+1]$. She knows that A is composed of distinct positive numbers, and B is derived from inserting a zero into A. She would like to know the index of this zero. Unfortunately, she does not have enough time to search the arrays, so she is looking for a faster algorithm. She wonders if you can design and analyze a fast algorithm for her to find the index of the zero in B. (Also, she has told me not to give many points for algorithms with running time much larger than $O(\log n)$).

She has provided the following example to ensure that the problem statement is clear.

$$A : 1, 3, 4, 6, 7, 8, 9, 20$$
$$B : 1, 3, 0, 4, 6, 7, 8, 9, 20.$$

Your algorithm should return 3 in this case, which is the index of the zero in B.

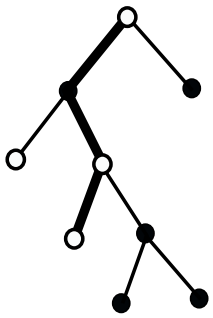
Problem 4. Given a positive integer n, find the minimum number of perfect squares that sum to n. For example if $n = 7$ then it can be written as $4 + 1 + 1 + 1$, which is a sum of four perfect squares, or if $n = 13$ then it can be written as $9 + 4$, which is a sum of two perfect squares. Here is a sequence of recommended steps for this problem:

(a) Come up with a recursion, and argue that it is correct.

(b) Use memoization or dynamic programming to obtain a fast algorithm.

(c) Analyze the running time of your algorithm.

Problem 5. Given a tree with black and white vertices find a path from the root to a leaf with the maximum number of white vertices. The example below shows such a path with three white vertices.



Design an algorithm to find such a path for a given binary tree. Argue that your algorithm is correct and analyze its running time. (Hint: Use recursion!)