EE360C: Algorithms
The University of Texas at Austin — Exam #1
Drs. Christine Julien and Andrea Thomaz — February 23, 2017

Name: **EID:**

Exam #1

Instructions. WRITE LEGIBLY.
No calculators, laptops, or other devices are allowed. This exam is **closed book**, but you are allowed to use a **one-page, handwritten** reference sheet. Write your answers on the test pages, in the space provided. If you need scratch paper, use the back of the test pages, but note that we will not look at the backs of pages during our grading process. 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 a question will not be answered during the exam.**

When asked to describe an algorithm, you may describe it in English or in pseudocode. If you choose the latter, make sure the pseudocode is understandable.

If you write information in response to a question, and that information is incorrect, you will not earn full credit. In the same vein, if a question asks for a finite number of things, and you provide "extra" things, we will ignore anything extra, and grade only the first answers. If many you have extra markings on a page, be sure to clearly indicate what is to be considered your answer.
You have **90 minutes** to complete the exam. The maximum possible score is 100.

Name: **EID:**

Problem 1: School District Consulting: Stable High School Admissions [26 points] The city of New York has more than 500 high school programs. Up until about ten years ago, the city annually invited the more than 90,000 students who were completing eighth grade to provide a list of their five preferred high schools, in ranked order[1]. The schools used a variety of different algorithms to admit students. Schools admitted students based on three different approaches: (1) by lottery, i.e., students are randomly selected from among those who requested the school; (2) by locality, i.e., students are admitted based on how close they live to the school; or (3) individually, i.e., based on their educational or extra-curricular achievements, including test scores or proclivity for the arts.

Schools were given the lists of applicants and could decide (based on the above strategies) to which students to offer admission. All schools made decisions independently. Once all first round decisions were made, they were all sent to the students, and each student could opt to take any offered admission (though a student would only be offered admission to a school he or she applied to). After offers had been accepted, schools were allowed to make a second round of offers and then a third. At the end of three rounds, any unassigned students were assigned via an "administrative process".

(a) **[6 points]** Give an example of students with their ranks and schools with their strategies in which a student would remain unmatched at the end of three rounds of offers.

[1] https://www.nytimes.com/2014/12/07/nyregion/how-game-theory-helped-improve-new-york-city-high-school-application-process.html

Name: **EID:**

(b) **[6 points]** Explain, *as briefly as possible*, what is fundamentally wrong with this approach. That is, what about the algorithm design could prevent all students from receiving matches? Hint: recall the proof of correctness for Gale-Shapely, and think about what was true upon termination of the algorithm.

Part II: In 2003, this process was replaced with a new process based loosely on the Gale-Shapely algorithm[2]. For simplicity, let's assume the idea is to use *exactly* the Gale Shapely algorithm.

(a) **[7 points]** Explain, as concisely as possible (full sentences are not required) how to map the *New York City Schools* problem onto Gale-Shapely. Be sure to consider the schools' different strategies.

[2]Abdulkadiroglu, Atila, Parag A. Pathak, and Alvin E. Roth. "The New York City high school match." American Economic Review (2005): 364-367.

Name: **EID:**

(b) **[7 points]** In the original algorithm, parents learned over time, that there were strategies in providing ranking of schools. For instance, for less academically strong students, *not* initially ranking the very-top schools might mean that a student would get into a nearly-top school in the first round. Is it possible to map the problem onto Gale-Shapely in a way that ensures that there is no such "dishonest" strategy on the students' part? If so, state what aspect of your mapping ensures this. If it is not possible, (briefly) explain why.

Name: **EID:**

Problem 2: School District Consulting: Automating Student Ranking [20 points] Returning to the previous scenario, each high school's strategy may be specific to the school. However, also imagine that each middle school wants to protect the academic records of the applicants from that school. This requires delicately handling the schools that use the individualized strategy (strategy (3), above).

Instead of releasing complete student records to the high school, each middle school generates a ranked ordering of its students for *each* high school, based on the high school's supplied metric. Each high school receives M such ranked orderings of eighth graders, where M is the number of middle schools in the city. In such a ranking, each student is represented as a pair $\langle name, score \rangle$, where *score* is the quantitative value that results in executing the high school's metric over the student's record.

The high school needs to submit a *merged* ranking of all students from all of the city's middle schools. That is, if n_i is the number of eighth graders at school i, the length of this merged ranking should be $N = \sum_{i=1}^{M} n_i$.

Give an efficient algorithm for solving this problem. Your algorithm should take as input M ranked lists of arbitrary lengths (each containing $\langle student, score \rangle$ pairs) and generate a single overall ranking of length N. You should give the most efficient algorithm you can think of and state its asymptotic time complexity. You will be graded on both the correctness and efficiency of your algorithm.

Name: **EID:**

Problem 3: Graph Algorithms [16 points]
Give an $O(m * (n + m))$ time algorithm to find all bridge edges in a connected graph. Recall that a bridge of a connected graph is an edge whose removal disconnects the graph. Briefly justify the running time of your algorithm.

Name: **EID:**

EXTRA CREDIT – Up to 5 points

Do not answer this question until you have completed the rest of the exam.

Describe an asymptotically faster algorithm for accomplishing the same goal. State the asymptotic time complexity of your algorithm. You will be graded based on your clarity and the correctness of your algorithm

Name: **EID:**

Problem 4: Graph Algorithms Again [20 points]
Consider the algorithms for Breadth First Search (BFS) and Depth First Search (DFS) (both given via pseudocode below).

```
BFS(s):
  Set Color[s] = gray
  Set Color[v] = white for all other v
  Set the current BFS tree T = 0
  Set the queue Q = {s}
  While Q is not empty
    u = Q.dequeue
    Consider each edge (u,v) incident to u
    If Color[v] = white then
      Set Color[v] = gray
      Q.enqueue(v)
      Add edge (u,v) to T
    Endif
    Color[u] = black
  Endwhile
```

```
DFS(s):
  Initialize S to contain only s
  While S is not empty
    Take a node u from S
    If Explored[u] = false then
      Set Explored[u] = true
      For each edge (u,v) incident to u
        Add v to the stack S
      Endfor
    Endif
  Endwhile
```

Recall that n counts the number of nodes in a graph, while m counts the number of edges. However, now consider b to be the maximum branching factor of the resulting BFS or DFS tree, where a given node's branching factor is defined as its number of children. Consider d to be the maximum depth of the tree, i.e., the longest path from the root to any other node.

(a) **[7 points]** What is the asymptotic **space** complexity of BFS in terms of b and/or d? (Ignore the space used for storing the `Color` array.) That is, at any point in the execution of the algorithm, what is the maximum space that must be allocated to the queue used in BFS? **Briefly** justify your response.

Name: **EID:**

(b) **[3 points]** Give an example graph of $n = 7$ nodes that results in the worst case space usage (over all graphs of 7 nodes) in executing the BFS algorithm above.

(c) **[7 points]** What is the asymptotic **space** complexity of DFS in terms of b and/or d?? That is, at any point in the execution of the algorithm, what is the maximum space that must be allocated to the stack used in DFS? **Briefly** justify your response.

(d) **[3 points]** Give an example graph of $n = 7$ nodes that results in the worst case space usage (over all graphs of 7 nodes) in executing the DFS algorithm above.

Name: **EID:**

Problem 5: Time Complexity [18 points]
Consider the following four versions of the `TEST*` function. Do not try to figure out what they're accomplishing; it's not important. Assume the `check` function executes in constant time.

```
TEST1(N):
  i = N
  foo = false
  While !foo AND i > 0
    foo = check(i)
    i = i - sqrt(N)
  Endwhile
```

```
TEST2(N):
  i = N
  foo = false
  While !foo AND i > 0
    foo = check(i)
    i = i - 1
  Endwhile
```

```
TEST3(N):
  i = N
  foo = false
  While !foo AND i >= 1
    foo = check(i)
    i = i/2
  Endwhile
```

```
TEST4(N):
  i = N
  foo = false
  While !foo AND i > 0
    foo = check(i)
    i = i - 13
  Endwhile
```

(a) **[12 points]** Give a **tight** asymptotic upper bound for the running time of each of the four test functions.

(b) **[6 points]** Assuming the four approaches equally solve some given problem, put them order of desirable running times. Briefly justify your responses.