Algorithms Midterm Exam 1

October 7, 2024

In preparing your solutions to the exam, you are **allowed to use any sources** including textbooks, other professors, previous homeworks and solutions, any sources on the Internet, or (especially!) each other. The only source you may *not* use is me. Of course, I am happy to answer general questions, go over homework problems or examples from class, or answer clarifying questions about exam problems.

You may cite any theorem proved in class or on a problem set without proof. Likewise, you may use any algorithm from class or a problem set as a subroutine, without having to redefine it.

The exam will take place on Monday, October 14, in class. You should bring something to write with, but no other external resources are allowed. I will provide fresh copies of the exam as well as blank paper. If you have accommodations that you wish to make use of, let me know soon so we can make appropriate arrangements.

General advice: trying to simply memorize all the solutions will most likely go poorly. I will not look kindly on solutions which have many of the right ideas but are confused about details, because such solutions tend to reflect a lack of understanding. Ideally, you should deeply understand the ideas and reasoning behind each solution, so you can correctly reconstruct details as necessary.

For credit on the exam, you must score ‘S’ (satisfactory) or better on every problem.

Question 1. Characterize the asymptotic behavior of each of the following in terms of Θ, as a function of n and/or m. Give a short proof/justification for each answer. Full credit will only be given for the best (fastest) possible algorithms.

(a) $1 + 2 + 3 + \cdots + n/2$

(b) $4 + 8 + 16 + 32 + \cdots + 2^n$

(c) The number of handshakes at a party with n people, if everyone shakes hands with everyone else.

(d) Time needed to find the shortest path between two given vertices in an unweighted, undirected graph with n vertices and m edges.

(e) Time needed to find the shortest path between two given vertices in a weighted, directed graph (assuming all weights are nonnegative) with n vertices and m edges.

(f) The number of `print` statements executed by the following Python code:

```
for i in range(n):
  for j in range(i,n):
    print(j*'*')
```

(g) The number of times the `while` loop in the following Python code executes:

```
x = 0
y = 1
s = 0
while y < n:
  s += y
  x,y = y,x+y
```

Question 2. Prove, or disprove with a counterexample: every tree with $n \geq 2$ vertices has at least two leaves.

Question 3. Consider the following algorithm to determine whether an undirected, unweighted graph G has any cycles: pick an arbitrary vertex v and run a breadth-first search (BFS), generating a sequence of layers L_0, L_1, L_2, ... If there is any edge between two vertices in the same layer, then report that G has a cycle; otherwise, report that G has no cycles.

Prove or disprove the correctness of this algorithm.

Question 4. Recall *Dijkstra's algorithm* for finding shortest paths in a directed, weighted graph.

(a) Why doesn't Dijkstra's algorithm work if edges in the graph can have negative weights (even if there are no directed cycles)? Give an example

of a directed graph with no directed cycles where Dijkstra's algorithm fails to find the minimum-weight path between a pair of vertices. Be sure to demonstrate that you understand *why* your graph is a counterexample; that is, show what Dijkstra's algorithm does on your example graph, and explain why the path it finds is not the minimum-weight path.

(b) What happens if we replace the word "smallest" in Dijkstra's algorithm with the word "biggest"—that is, we use a max priority queue so we pull out the vertex u with the *maximum* distance on each iteration, and for each edge (u, v) we update $d[v]$ to be the *maximum* of $d[v]$ and the sum $d[u] + w_{uv}$. Can we use this modified Dijkstra's algorithm to find *longest* paths bewteen nodes in any graph without directed cycles? Prove that this works, or give a counterexample (with explanation) where it doesn't.

Question 5. In class, while describing Kahn's algorithm for computing a topological sort, I mentioned that it could be extended to find directed cycles. **Describe** a $\Theta(V + E)$ algorithm which, given a directed graph $G = (V, E)$ as input, will either find a directed cycle in G, or report that G is acyclic. You should describe your algorithm using appropriate pseudocode, at a level of detail that would enable someone else to turn your algorithm into working code (similar to what we have done in class). **Prove** your algorithm is correct, and **analyze** its asymptotic running time.