

PhD Qualifying Exam, Algorithms: Spring 2000

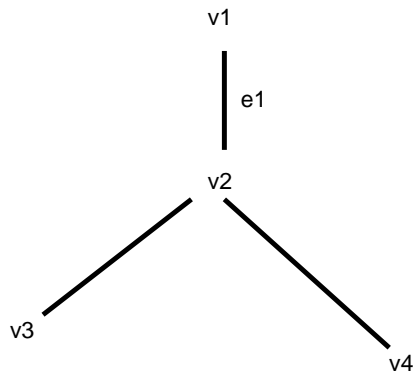
Choose three problems from the first *easy* group, and two from the *hard* group. **You should answer FIVE questions total.**

Easy

1. We have a list of air fares between pairs of cities. We want to find the cheapest way to get from New York to Los Angeles, and this may involve taking a number of flights. What algorithm should we apply, and sketch pseudo code for it (the right algorithm to use should be obvious).
2. What is the complexity (Big-O) of Quicksort if we implement it in assembly language? What if we implement it in Java?
3. Give a definition and draw a graph to illustrate each of the following.
 - $O(f(n))$
 - $\Theta(f(n))$
 - $\Omega(f(n))$
4. Assume we have a linked list, and n elements we wish to insert into this list. We append to the list by traversing it, and attaching new elements to the end. What is the complexity of building the list? Can it be made better?
5. We have 124 teams, in a single elimination tournament. Two teams play each other in each game, and if a team loses a game, they are eliminated. There are no ties in any games. How many games are played to obtain a champion?
6. Professor X develops a new computer which operates in trinary, rather than binary. We have 0, 1, and 2 as digits, rather than just 0 and 1. How many binary bits will we need to represent a 4 digit trinary number?
7. Heaps are used in many algorithms to improve computational complexity. Describe an algorithm which commonly utilizes a heap, give the complexity, and explain how the complexity would increase if another data structure was used (for example, an array which is kept sorted).
8. Hash tables are used in many algorithms to improve computational complexity. Describe an algorithm which commonly utilizes a hash table, give the complexity, and explain how the complexity would increase if the hash table was not used.
9. Give an example of a problem for which dynamic programming is appropriate, and sketch pseudocode for the problem.

Hard

1. Three-coloring a graph involves assigning a unique color to each vertex of a graph, using only Red, Green, and Blue. A classic reduction is to show that if we can solve satisfiability, we can also solve three-coloring. Describe briefly how you would use a satisfiability-solver to perform three-coloring, and give a small example.
2. Suppose you have n unique integers; you could place them into a binary search tree in many different ways, making many different trees. Describe a recursive formulation to calculate the number of possible different trees, and give dynamic programming pseudocode to calculate that number.



Vertex covers:
 $\{v1, v3, v4\}$, or $\{v2\}$.
 The optimal cover is $\{v2\}$.

A possible vertex cover by the algorithm: $\{v1, v2\}$. If the endpoints of edge $e1$ are removed, and all incident edges are also removed, then there are no remaining edges.

Figure 1: A simple graph, and a set of possible vertex covers.

3. Some current algorithms research involves randomization. Explain why we might want to introduce randomization in Quicksort, and discuss briefly other algorithms where we might find randomization.
4. Suppose L_1 and L_2 are two languages, and $L_1 \subseteq L_2$. Suppose L_2 is regular and the complement of L_2 is not empty. Can you deduce that L_1 is regular? Suppose instead that L_2 is recursive but not context-free and the complement of L_2 is not empty. Can you deduce that L_1 is recursive? Prove your assertions.
5. Define a context-sensitive grammar and context-sensitive language. How are context-sensitive languages described using recognizers (Turing machines, automata, etc)? What is the relation between context-sensitive languages and recursive languages?
6. If we have a graph, a vertex cover for the graph is the set of vertices such that at least one endpoint of every edge is a member of the vertex cover (see Figure 1). If we remove all the vertices in the cover from the graph, no two points in the graph will remain connected. Finding the smallest vertex cover is difficult. Suppose we apply the following algorithm.
 - Set VC to empty
 - While there are edges in the graph
 - Select any edge at random
 - Add both endpoints of the edge into VC
 - Remove all edges that connect to either endpoint

What can you say regarding the optimality of the algorithm? Is it optimal? Within some factor of optimal?