

There are two required questions. Pick two questions from the “easy” group, two questions from the “medium” group, and one question from the “hard” group. Please put no more than one answer on a page, and indicate which question you are answering.

**REQUIRED 1:** Give formal definitions for O, Omega, Theta, P, and NP.

**REQUIRED 2:** Describe a few algorithms you expect to use as part of your PhD research. What types of problems are you expecting to solve, and at a high level, what algorithmic strategies do you expect to employ?

### **EASY QUESTIONS: PICK ANY TWO**

**EASY 1:** Describe differences between Dijkstra’s algorithm and the Bellman-Ford algorithm. In what cases might one be better than the other?

**EASY 2:** Sketch pseudocode for inserting an element into a heap (it may be either a minimum or maximum heap).

**EASY 3:** What is the Big-O complexity for the following function?

```
int f(int n)
{
    int i, x;

    if (n < 0)
        return 1;

    x = 0;
    for (i = 0; i < n; ++i)
        x = x + i;
    return x + f(n - 2);
}
```

**EASY 4:** Suppose algorithm A is  $O(n \log n)$ , and algorithm B is  $O(n)$ . Is it possible for algorithm A to be faster than B? Under what circumstances?

**MEDIUM QUESTIONS: PICK ANY TWO**

**MEDIUM 1:** Suppose you have  $n$  unique integers, and you wish to put them into a binary search tree. Give a recursive formulation or some efficient pseudocode that can be used to calculate the number of different possible search trees that can be constructed. In the search trees, some values may be stored at internal (non-leaf) nodes. As a specific example, if we have two integers,  $\{1, 2\}$ , there are two different search trees (one with 1 at the root, 2 as the right child, and a second where 2 is at the root, and 1 is the left child).

**MEDIUM 2:** Describe the disjoint set data structure that might be used in Kruskal's algorithm. Sketch an small example to show how set merge operates, and how one would determine if two elements are in the same set. Also describe "path compression."

**MEDIUM 3:** Why might one use randomization in a quicksort implementation? Under what conditions would quicksort perform poorly, and how would the randomization help to reduce this problem.

**MEDIUM 4:** The Max-Flow Min-Cut theorem gives three conditions for a maximum flow network. If any of the three conditions is true, all must be true. State the conditions, and give a brief description on why they hold.

**HARD QUESTIONS: PICK ANY ONE**

**HARD 1:** Provide more details on the differences between P and NP. Your description should give specific examples of problems in each set, and also address what it means for a problem to be NP-Complete. Further, please briefly describe a polynomial time reduction, and give a simple example.

**HARD 2:** The traveling salesman problem is well known, and there is a trivial  $O(n!)$  algorithm for it. Describe a better approach, where the computational complexity is  $O(2^n n^2)$ .

**HARD 3:** Suppose we have  $n$  people, and some of these people dislike each other. We need to gather them at a round dinner table, and cannot place two people who dislike each other in adjacent seats. Describe an algorithm that will find a seating plan, or will state that no such plan is possible. Prove that this problem is NP complete.