Candidacy Exam on Algorithms and Data Structures

Department of Computer Science and Engineering
Pennsylvania State University

January 28, 2019

- Books, cellphones, calculators, and other electronic devices (except for watches) are not allowed during the exam.
- Write a solution to each problem on separate sheets of paper.
- This exam contains six problems. Return solutions to all problems for full credit.
- When you describe an algorithm, you should also justify its correctness and your estimate of the running time.

1. **Asymptotic growth** **(25 points)**

The items below describe the functions of a parameter n. For each item, write down a simple asymptotic expression of the form $\Theta\left(2^{an}n^b(\log n)^c\right)$, where a, b, c are constants (possibly 0).

- The running time of the best known algorithm for searching for an item in a sorted array of length n.
- The running time of the best known algorithm for sorting a list of n numbers, where each number is between 1 and n^2.
- The running time of the best known algorithm that takes a sorted array A of n numbers and determines if there exists indices i and j such that $A[i] + A[j] = 0$
- The number of matrix multiplications needed to compute X^{n^2}, where X is a square matrix.
- The expected running time of randomized `QuickSort` (i.e., the pivots are chosen uniformly at random).
- The worst case running time of deteministic `QuickSort` where the pivot is always chosen to be the first element in the array.
- The solution to the recurrence given by $T(1) = 1$, and $T(n) = T(n-1) + n$ for all $n > 1$.
- The solution to the recurrence given by $T(1) = 1$, and $T(n) = 3T(\frac{n}{3}) + n$ for all $n > 1$.
- The solution to the recurrence given by $T(1) = 3$, $T(2) = 6$, and defined by $T(n) = 3T(n-1) - 2T(n-2)$.

2. **Data Structures** **(25 points)**

Consider an array of n numbers $[x_1, x_2, \ldots, x_n]$ that you will be receiving one by one in a single pass (i.e., as a stream of numbers), and not allowed to revisit previous numbers. Assuming you are provided a black-box access to a `Heap` data structure (either min and max heap), design an effective data structure that after receiving the ith number, reports the median of numbers $x_1, x_2, \ldots, x_i$ observed so far. The time complexity of your algorithm should be $O(\log n)$ in worst case. Write down the pseudocode of your algorithm and analyze its time complexity. For simplicity assume there are no duplicates in the stream. [Note: If the input size is odd, the median would be the middle element of sorted data and if the input size is even, the median would be the average of middle two elements in sorted stream.]

3. Divide and conquer (25 points)

You are participating in a hit game called "Beat Your Neighbors!" designed as follows. You are presented with an $m \times n$ grid of boxes, each containing a unique number. The cost of opening a box is \$1. Your goal is to find the box whose number is larger than its neighbors in the grid (above, below, left, and right). If you spend less money than your opponents, you will win the prize.

- Suppose $m = 1$. Design an algorithm that requires opening $O(\log n)$ boxes to find a box with a number that is bigger than any of its neighbors'.
- Suppose the grid is square, i.e., $m = n$. Describe an algorithm that requires opening $O(n)$ boxes to find a box with a number that is larger than any of its than its neighbors.

4. Graph Algorithms (25 points)

You are given a set of cities, along with the pattern of highways between them, in the form of an undirected graph $G = (V, E)$. Each stretch of highway $e \in E$ connects two of the cities, and you know its length in miles, ℓ_e. You want to get from city s to city t. There's one problem: your car can only hold enough gas to cover L miles. There are gas stations in each city, but not between cities. Therefore, you can only take a route if every one of its edges has length $\ell_e \leq L$.

(a) Given the limitation on your car's fuel tank capacity, show how to determine in linear time $O(|V| + |E|)$ whether there is a feasible route from s to t. State your algorithm clearly, prove that it is correct and analyze its running time

(b) You are now planning to buy a new car, and you want to know the minimum fuel tank capacity that is needed to travel from s to t. Give an $O((|V| + |E|) \log |V|)$ algorithm to determine this. State your algorithm clearly, prove that it is correct and analyze its running time

5. **Dynamic Programming** **(25 points)**

You are given a checkerboard which has 4 rows and n columns, and has an integer written in each square. You are also given a set of $2n$ pebbles, and want to place some or all of these on the checkerboard (each pebble can be placed on exactly one square) so as to maximize the sum of the integers that are covered by pebbles. There is one constraint: for a placement of pebbles to be legal, no two of them can be on horizontally or vertically adjacent squares (diagonal adjacency is fine).

(a) Determine the number of legal *patterns* that can occur in any column (in isolation, ignoring the pebbles in adjacent columns) and describe these patterns.

Call two patterns *compatible* if they can be placed on adjacent columns to form a legal placement. Let us consider subproblems consisting of the first k columns $1 \leq k \leq n$. Each subproblem can be assigned a *type*, which is the pattern occurring in the last column.

(b) Using the notions of compatibility and type, give an $O(n)$-time dynamic programming algorithms for computing an optimal placement.

6. **Greedy** **(25 points)**

Alice and Bob are playing a computer game. Each of them has an army with n units. They will line up their respective units in two lines, with each of Alice's units facing exactly one of Bob's and vice versa. Then, each pair of units who face each other will fight and the stronger one will win, while the weaker one will be captured. If two opposing units are equally strong, Alice's unit will lose and will be captured. Alice knows how Bob will arrange its units beforehand, and must decide how to line up hers. Your task is to help Alice maximize the sum of the strengths of her units that are not captured during the battle.

You will be given two arrays of positive integers A and B, each of size n, that specify the strengths of the units of Alice and Bob, respectively. Design an algorithm that computes the maximum total strength of Alice's units that are not captured.

1. **Question: Asymptotic growth** **(25 points)**

2. **Question: Data Structures** **(25 points)**

3. **Question: Divide and conquer** **(25 points)**

(a)

(b)

4. **Question: Graph Algorithms** **(25 points)**

(a)

(b)

5. **Question: Dynamic Programming** **(25 points)**

(a)

(b)

6. **Question: Greedy** **(25 points)**