CSCI-GA.3520-001: Honors Analysis of Algorithms

Final Exam, Dec 19, 2019, 10:00-2:00pm

- This is a four hour exam. There are six questions, worth 10 points each. Answer all questions and all their subparts.
- This is a closed book exam. No books, notes, reference material, either hard-copy, soft-copy or online, is allowed.
- In the past years, to pass this exam, one needed to answer 3 or 4 problems well, instead of answering all the problems poorly.
- Please print your name and SID on the front of the envelope only (not on the exam booklets). Please answer each question in a separate booklet, and number each booklet according to the question.
- Read the questions carefully. Keep your answers legible, and brief but precise. Assume standard results and algorithms (i.e., those taught in class or referred to in the homeworks).
- You must prove correctness of your algorithm and prove its time bound unless stated otherwise. The algorithm can be written in plain English (preferred) or as a pseudo-code.

Best of luck!

Problem 1 (A Heap Challenge)

You are given a *min heap* containing n data items, along with a data item x and a positive integer k.

Your task is to design an algorithm that runs in time $O(k)$ and answers the following question: are there are least k items in the heap that are less than x?

Of course, you could go through the entire heap and just count the number of items that are less than x, but this would take time proportional to n. The challenge is to design an algorithm whose running time is $O(k)$ by somehow using the heap property.

Note: You can assume that all data items are positive integers and are distinct. In a min heap, the item stored at a vertex is less than the items stored at its children. The heap is given by presenting a pointer to the root. Each vertex has pointers to its children and its parent. The number of children of a given vertex can be arbitrary.

Problem 2 (Consecutive Sub-array of Maximum Sum)

You are given an array $A[1], \ldots, A[n]$ of integer values. The values can be positive, zero, or negative. For $1 \leq i \leq j \leq n$, a consecutive sub-array consists of the entries $A[i], A[i+1], \ldots, A[j-1], A[j]$. A consecutive sub-array could also be empty, denoted as $\emptyset$.

1. How many consecutive sub-arrays are there of an array of size n? Your answer needs to be correct up to a constant factor.

2. Give an $O(n)$ time algorithm to find a consecutive sub-array whose sum of values is the maximum among all consecutive sub-arrays. This sum would be zero for the empty array. You can assume that each integer operation, e.g. comparison and addition, takes constant time.

Note: Partial credit for less efficient algorithm, e.g. $O(n \log n)$ time algorithm.

Problem 3 (Hyper-graph Independent Set)

Let $H(V, E)$ be a 3-uniform hyper-graph, i.e. V is a set of vertices, E is a set of hyper-edges, and each hyper-edge $e \in E$ is a 3-element subset of V. An independent set in a hyper-graph is a set of vertices S such that there is no edge $e \in E$ such that $e \subseteq S$. Define the language HYPERGRAPH INDEPENDENT SET as follows.

$$\textsf{HYPERGRAPH INDEPENDENT SET} = \{\langle H(V,E), k\rangle \mid H(V,E) \text{ is a 3-uniform hyper-graph and has an independent set of size } k\}.$$

Show that HYPERGRAPH INDEPENDENT SET is NP-complete.

Problem 4 (Solving Inequalities)

You are given a system of m inequalities involving n variables, $x_1, \ldots, x_n$. Each inequality is of the form

$$x_j \geq x_i + c_{ij}$$

for some pair of indices $i, j \in \{1, \ldots, n\}$ and some constant c_{ij}, which is non-negative number (i.e., $c_{ij} \geq 0$). The system is given as a weighted, directed graph G. The graph G has vertices, numbered $1, \ldots, n$, and m edges. Each inequality as above is represented in G as an edge from vertex i to vertex j with weight c_{ij}. Moreover, G is represented in adjacency list format.

1. First assume that G is acyclic. Your task is to design an algorithm that takes G as input, and computes an assignment to the variables that satisfies the system of inequalities represented by G. More precisely, such a *satisfying assignment* is a list of non-negative numbers $(a_1, \ldots, a_n)$ that satisfy $a_j \geq a_i + c_{ij}$ for each inequality $x_j \geq x_i + c_{ij}$ in the system of inequalities. For full credit, your algorithm should run in time $O(m + n)$.

2. Now consider the previous problem, but G is a general directed graph (not necessarily acyclic). Design an algorithm that determines if the system of inequalities has a satisfying assignment, and if so, outputs a satisfying assignment. Your algorithm should run in time $O(m + n)$.

Hint: Think carefully about the precise conditions under which the system of inequalities has a satisfying assignment. You may use any standard algorithm as a subroutine. You may also use an algorithm that solves the first part of the problem as a subroutine to solve the second part.

Problem 5 (Balls and Bins)

There are n balls $b_1, \ldots, b_n$ and n bins $B_1, \ldots, B_n$. Each ball b_i is placed into a uniformly random bin, independently for $1 \leq i \leq n$. Let k be a positive integer. Fix some $1 \leq j \leq n$ and let $\mathcal{E}_j$ be the event that the bin B_j has at least k balls. For a subset $T \subseteq \{b_1, \ldots, b_n\}$, $|T| = k$, let $\mathcal{E}_{j,T}$ be the event that all the balls in subset T are placed in the bin B_j.

1. What is $\Pr[\mathcal{E}_{j,T}]$?

2. Express the event $\mathcal{E}_j$ in terms of events $\mathcal{E}_{j,T}$.

3. What is an upper bound on $\Pr[\mathcal{E}_j]$?

4. What is the least value of k (call it k^*), as a function of n, such that

$$\Pr[\mathcal{E}_j] \leq \frac{0.01}{n}.$$

   Give justification. Your answer needs to be correct up to a constant factor (partial credit otherwise).

5. Show that with probability 0.99, *all* bins have at most k^* balls.

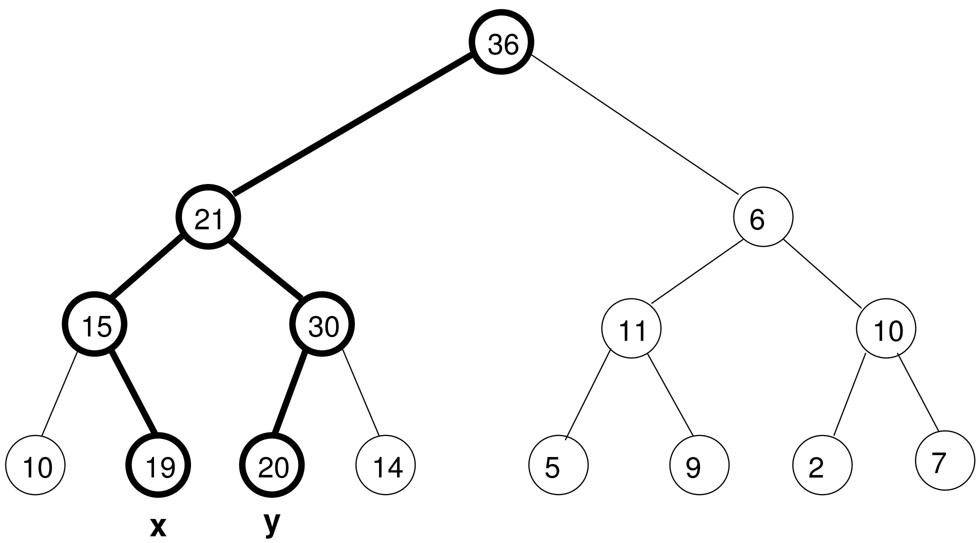
Note: You can assume that $2^{\frac{1}{2}k \log k} \leq k! \leq 2^{k \log k}$.

Problem 6

Suppose you are given a complete binary tree of height h with $n = 2^h$ leaves, where each vertex x (including each leaf) of this tree has an associated value $v(x)$ (an arbitrary real number). If x is a leaf, we denote by $A(x)$ the set of ancestors of x (including x as one of its own ancestors). That is, $A(x)$ consists of x, its parent, grandparent, etc, up to the root of the tree. Similarly, if x and y are distinct leaves we denote by $A(x, y)$ the ancestors of either x or y. That is,

$$A(x, y) = A(x) \cup A(y).$$

Define the function $f(x, y)$ to be the sum of the values of the vertices in $A(x, y)$ (see an example below). Give an algorithm that efficiently finds two leaves x_0 and y_0 such that $f(x_0, y_0)$ is as large as possible. What is the running time of your algorithm? For full credit, your algorithm should run in time $O(n)$.



A(x,y) shown in bold

f(x,y) = 19+15+21+36+20+30 = 141