CS 161 Midterm – Autumn 2016-17

(Do not turn this page until you are instructed to do so!)

Instructions: Please answer the following questions to the best of your ability. Provide full and rigorous proofs and include all relevant calculations unless the question says otherwise. When writing proofs, please strive for clarity and brevity (in that order). You have **80 minutes** to complete the exam. It is recommended to time yourself according to the problem weights. The exam is closed-book, closed-note, closed-internet, etc. However, you may use one letter-sized sheet (front and back) of notes as reference.

All of the intended answers can be written well within the space provided. Do not use the back side of printed pages for your answers. We have provided two blank pages at the end for your rough work. Good luck!

The following is a statement of the Stanford University Honor Code:

1. *The Honor Code is an undertaking of the students, individually and collectively:*
   (1) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
   (2) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.
2. *The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.*
3. *While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.*

By signing your name below, you acknowledge that you have abided by the Stanford Honor Code while taking this exam.

Signature: ______________________________

Name: ______________________________

SUNetID: ______________________________

Problem	#1	#2	#3	#4	#5	Total
Maximum	10	6	24	28	12	80

Question 1: True or False (10 points)

Indicate whether each statement is true or false. No justification is needed. Each statement is worth two points.

1. If $f(n) = O(g(n))$ and $g(n) = O(n^2)$, then $f(n) = O(n^2)$.

2. If the running time of algorithm A is given by $T_A(n) = T_A(n-2) + 5n$, and algorithm B is given by $T_B(n) = 8T_B(n/3) + n$, then algorithm B is asymptotically faster.

3. If $f(n) = O(g(n))$, then $2^{f(n)} = O(2^{g(n)})$.

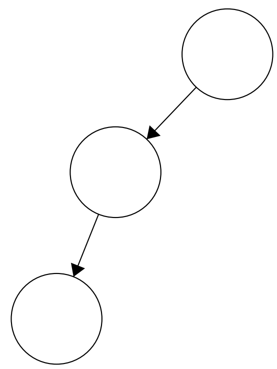
4. No sorting algorithm to sort n numbers in $\{1, \ldots, n\}$ can run in $O(n)$ time.

5. If $f(n) = O(n^2)$ then there is a constant c such that $f(n) \leq cn^2$ for all $n \geq 1$.

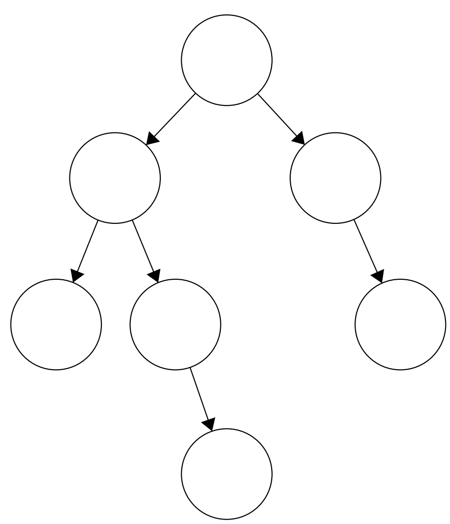
Question 2: Coloring Red-Black Trees (6 points)

For each of the two binary search trees below, is it possible to color it as valid red-black tree? If so, provide a red-black tree coloring. If not, write "impossible". In either case, no justification is needed. If you don't have red and black ink, write "r" to color a node red, or "b" to color it black. Each part is worth 3 points.

(a)



(b)



Question 3: Finding a BLIP in an Array (24 points)

Given an (unsorted) array A of n distinct numbers (i.e. no duplicates), index $i \in \{2, \ldots, n-1\}$ is said to be a BLIP if $A[i]$ is larger than $A[i-1]$ and $A[i+1]$. Index 1 is said to be a BLIP if $A[1] > A[2]$ and index n is said to be a BLIP if $A[n] > A[n-1]$. Our goal is to find an efficient algorithm to find a BLIP in an array of length n.

(a) (4 points) What is a naive solution to find a BLIP? What is the run time of this algorithm?

(b) (18 points) Design an $O(\log n)$ time algorithm for this problem. Outline the solution briefly in pseudocode and prove its correctness. (The next part (c) deals with run time.) *For correctness, it suffices to identify an invariant and explain in a few sentences why the invariant is maintained, as well as why it implies the output of the algorithm is correct.*

(additional space for answer to 3(b))

(c) (2 points) Write down a recurrence relation for the running time of your algorithm. (You need not solve it.)

Question 4: Searching Multiple Locations in an Array (28 points)

The MultiSearch problem is the following:
Input: an unsorted array A of n elements and a subset S of k positions, $S = \{p_1, \ldots, p_k\}$, $p_1 < p_2 < \ldots < p_k$.
Output: for each $i \in \{1, \ldots, k\}$, output the element of A in the p_i^{th} position in sorted order.
For example, if $k = 1$, $S = \{i\}$ corresponds to finding the i^{th} smallest element in the array.
$k = 2$, $S = \{1, n\}$ corresponds to finding the minimum and maximum elements of the array.

(a) (12 points) Design a divide-and-conquer algorithm to solve MultiSearch in $O(n \log k)$ time. Outline the solution briefly in pseudocode. Explain what the algorithm does in a few sentences, but you don't have to prove correctness.

Note: You will analyze the run time in part (b) and prove a lower bound in part (c).

(b) (8 points) Analyze the running time of your algorithm. Note that the running time is a function of two variables: n (the size of A) and k (the size of S). We suggest not trying to solve a recurrence relation directly, but instead reasoning about the recursion tree for your algorithm. What are the sizes of the subproblems at each level of the recursion tree? What is the total work done by the algorithm at each level (outside the recursive calls)? Finally, how many levels does your recursion tree have?

(c) (8 points) Show that any comparison based algorithm for MULTISEARCH needs $\Omega(k \log n)$ comparisons to solve this problem. [Note that the lower bound does not match the upper bound in part (a).]

(*Hint: The output of the algorithm can be viewed as follows: for each $i \in \{1, \ldots, k\}$, output the location in A where the p_i^{th} element in sorted order appears. What is the number of possible outputs that the algorithm can produce?*)

If needed, you can use the following bounds on $n!$: $\sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\left(\frac{1}{12n+1}\right)} \leq n! \leq \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\left(\frac{1}{12n}\right)}$

Question 5: Short Questions (12 points)

(a) Dijkstra's Fails on Negative Edge Weights (6 points)

Recall that when we proved the correctness of Dijkstra's shortest-path algorithm, we assumed that all edge lengths are non-negative. Prove that this assumption is necessary by providing an example where Dijkstra's algorithm fails when there are negative weights. The example should be a *directed* graph which has

- No more than 3 nodes
- Exactly one edge with negative weight
- A clearly marked source

In your example, identify the node for which the returned distance is incorrect, the true shortest path to that node, and the path produced by Dijkstra's algorithm.

(b) Universal Hash Functions (6 points)

Consider the family of hash functions $h_b(x) = 5x + b \pmod{11}$ where $b \in [0, 10]$ and the argument x to the hash function is from the universe $\{0, 1, 2, \ldots, 21\}$. Is this family universal? Explain.

This page left blank for your rough work. **Anything written here will not be graded**.

This page left blank for your rough work. **Anything written here will not be graded**.