CPSC 320 2017W1: Assignment 1

September 9, 2017

Please submit this assignment via GradeScope at `https://gradescope.com`. Be sure to identify everyone in your group if you're making a group submission (which we encourage!).

Submit by the deadline **Friday 22 Sep at 10PM**. For credit, your group must make a **single** submission via one group member's account, marking all other group members in that submission **using GradeScope's interface**. Your group's submission **must**:

- Be on time.
- Consist of a single, clearly legible file uploadable to GradeScope with clearly indicated solutions to the problems. (PDFs produced via LaTeX, Word, Google Docs, or other editing software work well. Scanned documents will likely work well. **High-quality** photographs are OK if we agree they're legible.)
- Include prominent numbering that corresponds to the numbering used in this assignment handout (not the individual quizzes). Put these **in order** starting each problem on a new page, ideally. If not, **very clearly** and prominently indicate which problem is answered where!
- Include at the start of the document the **ugrad.cs.ubc.ca e-mail addresses** of each member of your team. (No names are necessary.)
- Include at the start of the document the statement: "All group members have read and followed the guidelines for academic conduct in CPSC 320. As part of those rules, when collaborating with anyone outside my group, (1) I and my collaborators took no record but names (and GradeScope information) away, and (2) after a suitable break, my group created the assignment I am submitting without help from anyone other than the course staff." (Go read those guidelines!)
- Include at the start of the document your outside-group collaborators' ugrad.cs.ubc.ca e-mail addresses. (Be sure to get those when you collaborate!)

1 Looping Back to Asymptotic Analysis

NOTE: There are 8 blanks plus one **YES** or **NO** question to answer.

Each row of the table below lists a problem posed for an array of n numbers. You are determining **good big-O bounds for the worst-case performance** of efficient algorithms for each problem. In the **left blank**, give a bound for an efficient algorithm if the **input array is not known to be sorted**. In the **right blank**, give a bound for an efficient algorithm if the **input array is known to be already sorted**.

Each bound is one of: $O(1), O(\lg n), O(n), O(n \lg n), O(n^2)$.

Note: throughout this problem, assume basic operations on numbers take constant time.

Problem	big-O bound (unordered)	big-O bound (known to be sorted)
a. Find a given target value		
b. Find the maximum		
c. Find the largest absolute difference between any two values		
d. Find the total of the absolute differences between each pair of values		

Finally, note that for some algorithms over arrays, even if the input is not **known** to be sorted, sorted arrays are a common case worth optimizing for.

A friend suggests that this is the case for finding the maximum of an array of n numbers. Is there a correct and efficient algorithm for this problem that has an asymptotically better runtime in the case where the input happens to be sorted than in the worst case? (Circle the **best** answer.)

Answer: **YES** **NO**

1.1 For the assignment

In addition, for each of the 8 blanks, justify your answer by giving (1) **brief, high-level pseudocode for an efficient algorithm for the problem**, (2) **a brief description of the type of input that generates the algorithm's worst case**, and (3) **a brief justification of why your bound is appropriate for your algorithm/case**.

For the final **YES** or **NO** question, **briefly justify your answer**.

If you need to work with indexes, assume 0-based indexing, i.e., the first element of an array A of length n is $A[0]$ while the last is $A[n-1]$.

2 Structural Engineering

2.1 Choosing Your Structures

For each of the following, choose the data structure that most efficiently supports a solution of: **stack**, **queue**, **pri q** (priority queue implemented as a binary heap), and **dict** (dictionary/map implemented as a hash table). Choose the **best** answer in each case. If there are multiple best answers, just pick one.

1. Given a string s containing only characters from the set `[`, `]`, `(`, `)`, `{`, and `}`, determine if s is balanced. (If A is balanced, then (A), {A}, and [A] are balanced. If A and B are balanced, then AB is balanced. The empty string is balanced.) _____

2. Given a string s, build a data structure that can be used to rapidly find the number of times any character c appears in s. _____

3. Given a map of a cave (represented as a graph), a starting location (a vertex), and a set of exits (vertices), find a path from the start to an exit. _____

4. Given the same cave map inputs as above, find the path containing the least number of passages (edges) from the start to an exit. _____

5. Given the same cave map inputs as above plus a landmark (another vertex), find the path containing the least number of passages (edges) from the start to an exit and passing through the landmark. _____

6. Given an array of integers and value k, find the kth largest element of the array. _____

2.1.1 For the assignment

In addition, for each of the problems above, go back and justify your answer briefly, including briefly describing the algorithm you would implement using your chosen data structure.

2.2 Knowing Your Structures

Each item below describes an operation on a tree data structure. Choose the tightest worst-case running time for a good implementation of the operation among: $O(1), O(\lg n), O(n), O(n \lg n), O(n^2)$. n represents the number of items (keys or key/data pairs) in the structure. Note: BST is binary search tree.

1. Find the minimum key in an AVL tree. _____
2. Insert a key into a BST (not necessarily balanced). _____
3. Find the largest key less than a given key in a balanced BST. _____
4. Determine if a given BST satisfies the AVL balance property. _____
5. Perform a level order traversal of a BST. _____
6. Build a minHeap from a given array of integers. _____
7. Sort the elements of an array in place using the heap sort algorithm. _____

2.2.1 For the assignment

In addition, for each problem above, please justify your bound by giving (1) **brief, high-level pseudocode for an efficient algorithm for the problem**, (2) **a brief description of the type of input that generates the algorithm's worst case**, and (3) **a brief justification of why your bound is appropriate for your algorithm/case**.

Note: If you feel the algorithm you'd use is a widely known, common algorithm (like merge sort), just cite the algorithm in (1), but you'll still need enough detail about how it works for the justification in (3).

3 Marriage Rites

Each of the following problems presents a SMP scenario (for the classic stable marriage problem with n men, n women, and complete preference lists) and a statement about that scenario. For each one, indicate by writing the **bolded** word whether:

1. The statement is **ALWAYS** true, i.e., true in *every* SMP instance matching the scenario.
2. The statement is **SOMETIMES** true, i.e., true in some SMP instance matching the scenario but also false in some such instance.
3. The statement is **NEVER** true, i.e., true in *none* of the SMP instances matching the scenario.

Here are the problems:

1. **Scenario:** Any SMP instance solved using Gale-Shapley with women proposing. **Statement:** A man is unmarried in the solution.

   Circle the **best** answer: **ALWAYS** **SOMETIMES** **NEVER**

2. **Scenario:** Any SMP instance with $n \geq 1$ solved using Gale-Shapley with men proposing. **Statement:** A woman and man marry each other, although each prefers someone else.

   Circle the **best** answer: **ALWAYS** **SOMETIMES** **NEVER**

3. **Scenario:** An SMP instance with $n \geq 2$. **Statement:** A stable solution exists in which everyone gets their second choice partner.

   Circle the **best** answer: **ALWAYS** **SOMETIMES** **NEVER**

4. **Scenario:** Any SMP instance with $n \geq 1$ in which all the men share the same preference list, solved using Gale-Shapley with women proposing. **Statement:** At least one man gets his first choice in the solution.

   Circle the **best** answer: **ALWAYS** **SOMETIMES** **NEVER**

5. **Scenario:** Any SMP instance solved using Gale-Shapley. **Statement:** The loop in Gale-Shapley runs at least n times.

   Circle the **best** answer: **ALWAYS** **SOMETIMES** **NEVER**

3.1 For the assignment

In addition, for each of the problems above, please **briefly** justify your answer.

Justify an **ALWAYS** answer by giving a small instance that fits the scenario for which the statement is true and then briefly sketching the key points in a proof that the statement is true for all instances that fit the scenario.

Justify a **NEVER** answer by giving a small instance that fits the scenario for which the statement is **false** and then briefly sketching the key points in a proof that the statement is **false** for all instances that fit the scenario.

Justify a **SOMETIMES** answer by giving **two** small instances that fit the scenario: one for which the statement is true and one for which the statement is false.

4 Jus de Pokemon a la Mariage

In a new Pokemon Go challenge, trainers (i.e., players) working as a team want to catch Pokemon as rapidly as possible. We define the PGP (Pokemon Go Proximity) problem as: Given n trainers (and their locations) and n Pokemon (and their locations), assign each trainer a Pokemon to catch such that the trainers are nearby their target Pokemon.

1. Complete the following to make a sensible—if **not** necessarily optimal—reduction from PGP to SMP.

   Note: (1) Assume that SMP allows ties in preference lists. (2) A reduction from problem A to problem B (of the type that uses the solver for B exactly once) is a way to transform any instance of A into an instance of B such that the solution to the B instance can be transformed into a solution to the original A instance (including both "directions" of transformation but not including a solver for B).

Reduction: Given an instance of PGP, produce an instance of SMP as follows:

Each of the n men is _____.

Each of the n women is _____.

Construct the men's preference lists using the observation that a man m_i prefers woman w_j to woman w_k exactly when _____.

Generate women's preference lists similarly.

Then, given a solution to SMP, produce a solution to PGP by _____

_____.

2. Sketch a **small, clear instance of PGP** for which this reduction generates an optimal solution.

   For this problem and the next: (1) Optimal means the total distance travelled by trainers to their paired Pokemon is the minimum possible for any pairing. (2) Use **X** for trainers and **O** for Pokemon. (3) Your answer should be close in size to the minimum-size correct answer. Significantly larger instances will receive no credit. (4) There's plenty of room here for correct solutions.

3. Sketch a **small, clear instance of PGP** where this reduction does **not** generate an optimal solution. (Read the notes for the previous problem.)

   Hint: All trainers and Pokemon can sit on a single line. If you feel your reduction is optimal, be aware we're dubious and then sketch a proof of its optimality.

4.1 For the assignment

In addition, briefly sketch a brute force approach to implementing the reduction (not including the solution to the SMP instance, which can just use G-S or some other SMP solver) and give/briefly justify a big-O bound on the worst-case for this implementation.

Also, give the solution to your second instance found by the reduction (briefly explaining how and why this is reached), give the solution to that second instance that is optimal, and show that the optimal solution has lower total distance than the reduction's solution.