Due 16th June 2022 at 4pm Sydney time

In this assignment we review some basic algorithms and data structures, and we apply the divide-and-conquer paradigm. There are *four problems* each worth 20 marks, for a total of 80 marks.

Your solutions must be typed, machine readable PDF files. *All submissions will be checked for plagiarism!*

For each question requiring you to design an algorithm, you *must* justify the correctness of your algorithm. If a time bound is specified in the question, you also *must* argue that your algorithm meets this time bound.

Partial credit will be awarded for progress towards a solution.

The clarifications may be updated further in the Assignment 1 FAQ thread on the Ed forum: https://edstem.org/au/courses/8646/discussion/904022.

General clarifications:

- **How should I format/explain my submissions for algorithm design problems?**

  Explain the steps of your algorithm, and the logic behind these steps (when not obvious) in plain English. You may choose to include the justification of correctness as you describe the algorithm, or instead as a standalone section. You must also justify the claimed time complexity of your algorithm.

  You should aim for a similar standard to the solutions of the tutorial exercises. A rough guideline is "present your algorithms in such a way that an intern could implement them in a programming language of their own choice without requiring further clarifications".

- **Can my submission include diagrams?**

  Yes, but only as a supplement to an otherwise complete explanation. You cannot give an adequate justification of correctness by simply drawing an example.

- **Can my submission include code or pseudocode?**

  Pseudocode can be included but only as a supplement to an otherwise complete explanation. Code in any programming language will be ignored. Marks will not be deducted for including these, but we will only be marking the parts of your submission that are written in plain English (including mathematical notation).

- **What data structures and algorithms can I use?**

  In addition to the new material in this course, you can use any data structure or algorithm from Lecture 2 (i.e. those seen in prerequisite courses). If you are using them in their generic form, there is no need to reprove any of their properties, and you can simply quote the operations including their time complexity. If you need to modify the data structure or algorithm, you must provide details of the modification, and any impact it might have on the time complexity.

  Note that you should refer to these data structures and algorithms in abstract terms (e.g. hash table) as opposed to their implementation in any particular programming language (e.g. HashMap).

- **If the question has asked for $O(\ldots)$, can I submit an asymptotically faster solution?**

  Yes, $O(\ldots)$ is technically just an upper bound. Beware of attempting this though - seldom do we see it done correctly.

- **If the question has asked for $\Theta(\ldots)$, can I submit an asymptotically faster solution?**

  No.

- **Do the requested time complexities mean best, average or worst case?**

  By default, we mean worst case. We will always explicitly specify if we want the average case (i.e. expected time).

- **Is this on the right track? Can you give any additional hints? What have I missed or done wrong?**

  Sorry, we can't comment on these types of questions from individual students. Hints will be provided halfway between the release date and the due date of each assignment.

Question 1

Your friend attempted to send you an array of n bits, starting with a 0, and alternating between 0s and 1s. However, due to the network being unreliable, one of the bits was not sent. You received the array A of length $n-1$, and you would like to determine which bit was not sent.

1.1 [2 marks] Suppose $n = 8$, and you received the array $A = [0, 1, 0, 1, 1, 0, 1]$. Identify the bit that was not sent by its 1-based index in the array your friend tried to send you.

1.2 [5 marks] Suppose $n = 20$, and you received an array in which $A[10] = 0$. Which of the twenty bits could have been omitted from the original array? Provide reasoning to justify your answer.

1.3 [13 marks] Design an algorithm which runs in $O(\log n)$ time and finds the index of the missing bit.

Repeatedly apply the idea of 1.2 to narrow the search interval until the exact index is found.

Clarifications:

- **What is 1-based indexing?**

  Throughout this course, arrays begin from index 1, not 0. Therefore, an array A with n elements (denoted $A[1..n]$ for short) is indexed in full as $A[1], \ldots, A[n]$.

- **Should I explain the base cases?**

  Yes, it requires only one extra sentence.

Question 2

You are given an array A of n integers. You are required to find indices i, j, k (not necessarily distinct) such that $A[i] + A[j] = A[k]$, or return that no such indices exist.

Design algorithms which solve this problem and run in:

2.1 [**6 marks**] worst case $\Theta(n^2 \log n)$ time.

Try to test each pair of indices i, j in $\Theta(\log n)$ time.

2.2 [**6 marks**] *expected* $\Theta(n^2)$ time.

We have only made reference to the expected (i.e. average case) time complexity for two algorithms and the operations of one data structure.

2.3 [**8 marks**] worst case $\Theta(n^2)$ time.

Clarifications:

- **Do I need to find all triples (i, j, k) satisfying the equation?**

  No. The task is to find one such triple, or determine that none exist.

- **Do I need to submit a different solution for 2.2 and 2.3?**

  No. A correct solution to 2.3 also solves 2.2, so you may choose to skip 2.2 by writing "Please refer to my solution to 2.3. Of course, if that solution is flawed, you will be marked down in both 2.2 and 2.3.

Question 3

You are given an array A containing each integer from 1 to n exactly once. Your task is to compute $f(A)$, the sum of $\max(A[\ell..r])$ over all pairs of indices (ℓ, r) such that $\ell \le r$.

3.1 [**2 marks**] Suppose $n = 3$ and the array is $A = [2, 1, 3]$. Determine the value of $f(A)$.

For 3.2 and 3.3, suppose i and j are indices such that $1 \le i \le j \le n$, and let $g(i, j)$ be the number of subarrays $A[\ell..r]$ where $r > j$ and the maximum value is $A[i]$.

3.2 [**4 marks**] For a given pair of indices (i, j), under what conditions is $g(i, j)$ nonzero? In other words, what is the criterion for $A[i]$ to be the maximum of some subarray with its right endpoint at an index greater than j?

3.3 [**6 marks**] Given an index j, design an algorithm which runs in $O(n)$ time and determines the values of $g(i, j)$ for all $i < j$.

3.4 [**8 marks**] Design an algorithm which runs in $O(n \log n)$ time and determines the value of $f(A)$.

Refer to the problem of counting inversions from Lecture 3.

Clarifications:

- **Wait, this question looks different to what I'm solving!**

  This question was updated on 3rd June, as per this Moodle announcement. Please make sure that you are solving the correct question.

- **How do I explain the criterion for 3.2?**

  Specify a test so that $A[i]$ is the maximum of at least one such subarray if (i, j) passes the test and $A[i]$ is not the maximum of any such subarray if (i, j) fails it.

- **Do the elements of a subarray have to be contiguous in the original array A?**

  Yes.

- **In 3.2 and 3.3, can $i = j$?**

  It's been brought to our attention that the indexing isn't entirely consistent, but it doesn't make a great deal of difference since the cases where $i = j$ are trivial. Any interpretation is fine here.

Question 4

Your friend has constructed an array A of n distinct integers, where $n \geq 2$. However, you cannot access the elements of the array directly; instead, they instead only allow you to ask questions of the form "What is the maximum value amongst $A[l], A[l+1], \ldots, A[r-1], A[r]$?", where you may choose any valid indices l and r such that $l \leq r$. You may assume any questions you ask are answered in constant time.

Your goal is to determine the value of the **second largest** element in the array.

4.1 **[2 marks]** How can you find the value of the largest element in the array using only one question?

4.2 **[2 marks]** If you know that the largest element occurs at index i, how could you then find the value of the second largest element using only two questions?

4.3 **[11 marks]** Design an algorithm which runs in $O(\log n)$ time and determines the value of the second largest element in the array.

4.4 **[5 marks]** Now your friend imposes an extra restriction: for each question you ask except the first, the value of $r - l$ should be no larger than the value of $r - l$ in the previous question. Subject to this restriction, design an algorithm which runs in $O(\log n)$ time and determines the value of the second largest element in the array.

You may choose to skip 4.3, in which case we will mark your submission for 4.4 as if it was submitted for 4.3 also.

Ask for the the maximum of the entire array, then of the first half of the array, then of the second half of the array. What information does this give you?

Clarifications:

- **Can I access an array element indirectly by asking for the maximum of a range containing only one index?**

  Yes, you can ask for the maximum of a singleton array, but it costs you a question. You can't look up $A[i]$ without it counting to your questions asked.