CS 577: Introduction to Algorithms Spring 2017

Practice Midterm Exam

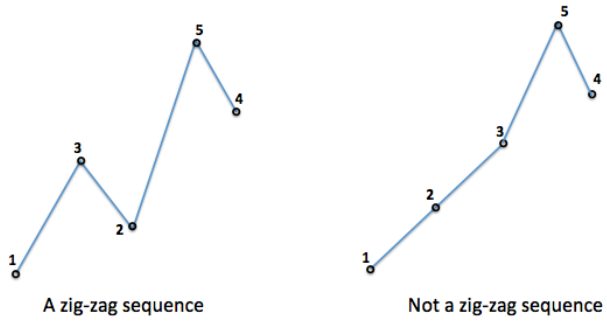
Instructor: Shuchi Chawla March 16, 7:15–9:15 PM

Guidelines:

- This exam is 2 hours long. You are required to answer each of the three questions.
- You can use all the results we showed in class or in the homework. Clearly state the results you use. Substantiate all other claims you make.
- The exam is closed book and closed notes, but you are allowed to bring a "cheat-sheet"—one page of notes (front and back).

GOOD LUCK!

1. A zig-zag sequence is a sequence A where, for every index i between 2 and $|A| - 1$, $A[i]$ is either larger than both of its neighbors $A[i-1]$ and $A[i+1]$, or smaller than both. For example, $1, 3, 2, 5, 4$ is a zig-zag sequence, whereas $1, 2, 3, 5, 4$ is not (see figure below). In this problem your goal is find the length of the longest zig-zag subsequence of a given sequence. For example, for the sequence $1, 2, 3, 5, 4$, the answer is 3 (corresponding to, e.g., $1, 5, 4$).



   Define a recursive function for computing the length of the longest zig-zag subsequence. Your equation should lead to an efficient (polynomial time) dynamic program, but you do not need to provide the algorithm, a proof of correctness, or the running time. Please remember to give a formal definition for your recursive function(s), and include any base cases.

2. You are given a graph G with weights w_e on edges and the MST T of the graph. Suppose that the weight of an edge e in the graph increases from w_e to w'_e, with all the other weights remaining the same. In this problem your goal is to design a linear time algorithm to recompute a new MST. You may assume that all weights in the graph before and after the change are distinct.

(a) Describe your algorithm.

(b) Give a brief (3-4 line) argument for correctness.

(c) Give a brief (1-2 line) argument for bounding the running time of your algorithm.

3. Let $n = 2^k - 1$ for some integer $k \geq 1$. You are given an unsorted array $A[1 \cdots n]$, which contains all but one of the integers in the range $0, \cdots, n$, so exactly one of these elements is missing from A. The problem is to determine the missing integer in $O(n)$ time. Each element of A is represented in binary using k bits, and the only way to access the elements is to call the function **bit**(i, j), which returns the value of the jth bit of $A[i]$. (In other words, you cannot directly access an element $A[i]$.)

   (a) Give a divide and conquer algorithm for this problem that finds the missing number while making at most $O(n)$ calls to the function **bit**.

   Hint: Can you reduce the problem to half its original size after reading at most n of the bits?

   (b) Give a brief (3-4 line) argument of correctness for your algorithm.

   (c) Give a recurrence for the the number of times your algorithm calls **bit**. Solve it to obtain the running time in asymptotic notation. No explanation is necessary.