

- The instructions are the same as in Homework 1, 2 and 3.

There are 6 questions for a total of 100 points.

1. (15 points) (*Rising Stock Prices*): When digging through archives, the daily price of stocks for n days of a long forgotten company was found and made available in an array $A[1, \dots, n]$. It is known that the stock price was stable during the initial few days, and then due to some global event, the price started rising significantly every day until the last day. We are trying to go over this data and figure out the trigger day (*i.e., the date on which the price started rising significantly*). You are asked to design an algorithm for this task.

You are given an array $A[1 \dots n]$, array size $n > 3$, and a stability parameter Δ which is a positive number. The array is such that for an (unknown) index $1 < j < n$, $|A[i] - A[i-1]| \leq \Delta$ for all $1 < i \leq j$ and $|A[i] - A[i-1]| > \Delta$ for all $i > j$. Note that $|\cdot|$ denotes absolute value, and j is the trigger day or index. Design an algorithm for finding the trigger index j . Give a running time analysis for your algorithm.

2. (15 points) (*Minimum product path*) You are given a pointer to the root r of a binary tree, where each vertex v has pointers $v.lc$ and $v.rc$ to the left and right child, and a value $Val(v) > 1$. The value NIL represents a null pointer, showing that v has no child of that type. You wish to find the path from r to some leaf that minimizes the product of values of vertices along that path. Give an algorithm to find the minimum product of vertices along such a path, along with a proof of correctness and runtime analysis.
3. (15 points) (*Skyline*) In a video game, a city's buildings are rectangles along an axis, specified as $B_i = (s_i, f_i, h_i)$, for $1 \leq i \leq n$, i.e., building i starts at $x = s_i$ and goes to $x = f_i$ at height h_i . When seen from a distance, the graphic is supposed to plot the skyline, which at each vertical point x has the height of the tallest building B_i with $s_i \leq x \leq f_i$. The goal is to plot the skyline as a series of points connected by line segments. See Figure 1 for an example. Give an $O(n \log n)$ time algorithm for this problem. Discuss the running time of your algorithm. You may assume n is a power of 2 for this question.

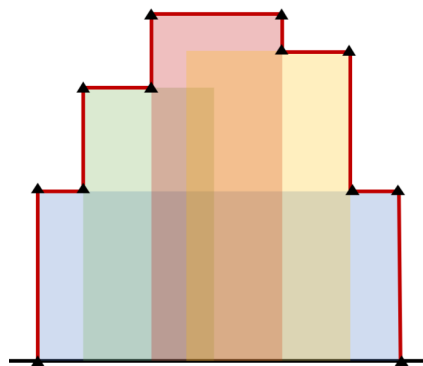
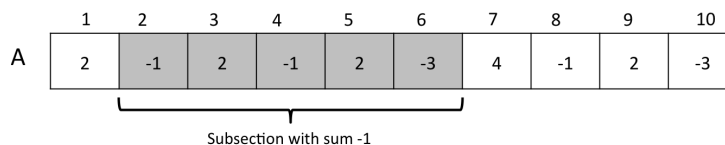


Figure 1: An example for the skyline problem with 4 buildings. The goal is to output the sequence of points (*shown as black triangles*), which, when connected with line segments, gives the skyline (*shown in red*).

4. (15 points) (*Maximum subsection*): Given a sequence of integers (positive or negative) in an array $A[1..n]$, the goal is to find a *subsection* of this array such that the sum of integers in the subsection is maximized. A subsection is a contiguous sequence of indices in the array.

For example, consider the array and one of its subsections in the figure below. The sum of integers in this subsection is -1 .



You have to design an algorithm that takes as input an array $A[1..n]$ and outputs a pair of indices (i, j) such that i and j are the starting and ending indices of the maximum subsection of the array A .

- There is a simple algorithm that, for every pair (i, j) , calculates the sum $\sum_{k=i}^j A[k]$ and outputs the pair that maximizes the sum. What is the running time of this algorithm?
- The previous algorithm can be made more efficient by first computing the cumulative sum in linear time. That is, compute an array $B[1..n]$ such that $B[i] = \sum_{k=1}^i A[k]$. The sum of subsection $A[i..j]$ can then be calculated as $B[j] - B[i-1]$ in constant time. Give a linear time algorithm for computing the cumulative sum array B . What is the algorithm's running time that uses array B to find the maximum subsection?
- Design a divide-and-conquer algorithm with running time $O(n \log n)$. Discuss the correctness and running time of your algorithm.

(There is an $O(n)$ time algorithm for this problem! That algorithm uses another algorithm design technique called *Dynamic Programming*, which is an upcoming topic.)

5. An array $A[1..n]$ is said to have a majority element if more than half (i.e., $> n/2$) of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form “is $A[i] \geq A[j]$?” (Think of the array elements as GIF files, say.) However you can answer questions of the form: “is $A[i] = A[j]$?” in constant time.

- (15 points) Show how to solve this problem in $O(n \log n)$ time. Provide a runtime analysis and proof of correctness.

(*Hint: Split the array A into two arrays A_1 and A_2 of half the size. Does knowing the majority elements of A_1 and A_2 help you figure out the majority element of A ? If so, you can use a divide-and-conquer approach.*)

- (15 points) Design a linear time algorithm. Provide a runtime analysis and proof of correctness.

(*Hint: Here is another divide-and-conquer approach:*

- Pair up the elements of A arbitrarily, to get $n/2$ pairs (say n is even)
- Look at each pair: if the two elements are different, discard both of them; if they are the same, keep just one of them
- Show that after this procedure, there are at most $n/2$ elements left, and that if A has a majority element, then it's a majority in the remaining set as well)

6. (*MIS revisited*): Consider the following backtracking algorithm intended to solve the MIS problem.

```
MISBTE1( $G = (V, E)$ )
· If ( $|V| \leq 1$ ) return( $|V|$ )
· Let  $v \in V$  be the least degree vertex in  $G$ 
·  $include = 1 + \text{MISBTE1}(G \setminus (\{v\} \cup N(v)))$ 
· if ( $\deg(v) \leq 2$ ) return( $include$ )
·  $exclude = \text{MISBTE1}(G \setminus \{v\})$ 
· return( $\max(include, exclude)$ )
```

- (a) (4 points) Prove or disprove: The algorithm works correctly for any graph.
- (b) (4 points) Prove or disprove: The algorithm works correctly for any degree-2 graph, i.e., a graph where the maximum degree of a node is 2.
- (c) (2 points) What is the running time of the algorithm on degree-2 graphs?