CS 341, Winter 2022 R. Peng, A. Storjohann

ASSIGNMENT 2

DUE: Wednesday, January 26, 11:59pm. DO NOT COPY. ACKNOWLEDGE YOUR SOURCES.

Please read `http://www.student.cs.uwaterloo.ca/~cs341` for general instructions and policies. **Note:** All logarithms are base 2 (i.e., $\log x$ is defined as $\log_2 x$).

Exercises. The following exercises are for practice only. You may ask about them in office hours. Do not hand them in.

1. In class you learned the Master Theorem for solving recurrences of the form $T(n) = aT(\frac{n}{b}) + cn^k$. Choose values of $a, b, k \in \{\frac{1}{2}, 1, 2, 4\}$ so that:
   (a) $T(n) \in \Theta(n^2)$.
   (b) $T(n) \in \Theta(\sqrt{n} \log n)$.

Problems. To be handed in.

1. [10 marks] **Divide & Conquer.** The number of bits in the binary representation of an integer a is given by
$$\lg a = \begin{cases} 1 & \text{if } a = 0; \\ 1 + \lfloor \log |a| \rfloor, & \text{otherwise.} \end{cases}$$
Consider a software library for multi-precision nonnegative integers (whose type we will call `LongInt`) that includes the following two functions:
   - `LongInt(x)` – takes as input a primitive type integer $x, 0 \leq x < 2^{64}$, and returns a `LongInt` with the value x.
     Run-time: $O(1)$.
   - `Mul(A,B)` – takes as input two `LongInt`'s and returns a `LongInt` with the value $A \times B$.
     Run-time: $O(\max(\lg A, \lg B) \min(\lg A, \lg B)^{\log_2 3 - 1})$ (Karatsuba's algorithm)

   In this question you will consider the problem of multiplying together n primitive type integers to obtain a single large integer.

   As input you are given an array $a[1 \ldots n]$ of length $n \geq 1$, each entry being a primitive integer (e.g., fits into 64 bits). Your task is to write a procedure that computes the product of all the integers in a and returns the answer as a `LongInt`. Here is one solution that simply accumulates the product:

```
MultiMul(a[1...n], n)
    P := LongInt(a[1])
    for i from 2 to n do
        P := Mul(P, LongInt(a[i]))
    return P
```

To estimate the run-time of algorithms using the multi-precision library we sum the cost of the calls to the functions defined above. Note that operations with primitive types like array indices and loop variables are assumed to have unit cost.

(a) [2 marks] Analyse the run-time of procedure `MultiMul`, that is, derive a tight big-O bound for the run-time in terms of n.

(b) [8 marks] Design and analyse a divide and conquer algorithm that does the same job as `MultiMul` but with run-time $O(n^{\log_2 3})$.

2. [10 marks] **Upper Envelopes of Rectangles Touching the x-axis**

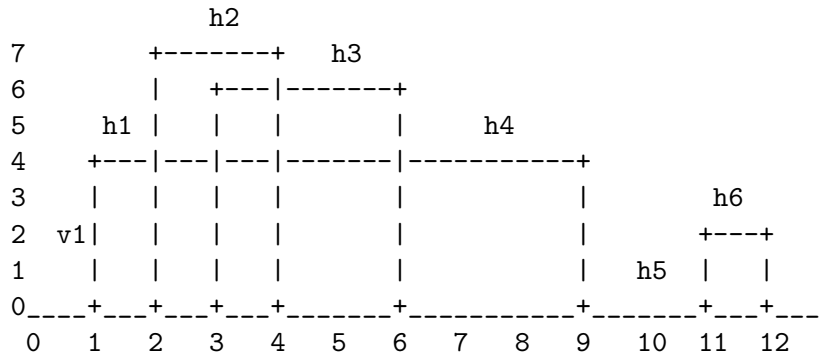
You have n rectangles, each with one side touching the x-axis, but above it. The goal is to find the upper envelope of these rectangles.

Formally, you should devise an algorithm that takes as input n and three length n arrays:

- $left[i]$ is the $x-$coordinate of the left edge of the i^{th} rectangle.
- $right[i]$ is the $x-$coordinate of the right edge of the i^{th} rectangle.
- $height[i]$ is the height of the i^{th} rectangle.

The algorithm should output all the corners of the upper envelope, starting with the leftmost point on the x-axis, ending with rightmost point on the x-axis, in order along the upper envelope. Note that the upper envelop can be considered to be an alternating sequence of vertical and horizontal line segments.

For example, consider four rectangles, $[1, 9]$ with height 4, $[2, 4]$ with height 7, $[3, 6]$ with height 6, and $[11, 12]$: with height 2.



The upper envelope should be: $(1, 0) - (1, 4) - (2, 4) - (2, 7) - (4, 7) - (4, 6) - (6, 6) - (6, 4) - (9, 4) - (9, 0) - (11, 0) - (11, 2) - (12, 2) - (12, 0)$. The six contiguous horizontal segments of the envelope are labelled $h_1, h_2, \ldots, h_6$. There are seven vertical line segments, but because of space restrictions, only the first vertical line segment v_1 of the envelop is labelled.

(a) [3 marks] Show that the upper envelope of n such rectangles contains at most $6n$ segments.

(b) [4 marks] Show that given two upper envelopes of such rectangles, each specified by at most n points, we can find the upper envelope of the union of the corresponding rectangles in $O(n)$ time.

(c) [3 marks] Give an $O(n \log n)$ time algorithm for computing the upper envelope of n such rectangles.

Challenge Question This is for fun and enrichment only. Do not hand it in.

1. For question 2, what if instead of rectangles, we have triangles with bases touching the x-axis? You may want to first prove that the size of the upper envelope is still at most $O(n \log n)$.