Change log:

- Problem 6 (version 2): added bounds on the numbers to ensure all intermediate numbers are non-negative integers \( \leq n^2 \).

This problem set has a total of 9 written problems, plus 1 coding problems, on 2 pages.

Written solutions should be submitted to Canvas by 2pm Wednesday Apr 14, 2021. Coding solutions should be submitted separately.

While there are explanations of some of these problems available on the internet. Please refrain from directly copying code you find.

1. Show that a sequence of \( n \) of the following operations on an array \( A[1\ldots n] \) can be answered in \( O(n \log n) \) time:

   (a) Add \( x \) (possibly positive or negative) to all \( A[l\ldots r] \).

   (b) Query for the max value in some range \( [l, r] \).

2. Use the data structure from part (a), show that given a set of \( n \) axis parallel rectangles \( ([x_1, x_2] \times [y_1, y_2]) \) on the plane, with associated weights, one can find the point contained in the max total weight of these rectangles in \( O(n \log n) \) time.

3. Consider a collection of \( n \) quadratics \( a_1 x^2 + b_1 x + c_1, a_2 x^2 + b_2 x + c_2, \ldots, a_n x^2 + b_n x + c_n \). Under a consistent tie breaking rule (e.g. smaller index), partition \( x \) into ranges by the index of the maximizer function. Prove that the number of ranges is at most \( O(n \log n) \).

4. Show that a collection of quadratics can be efficiently maintained under insertion and max value queries. That is, a sequence of \( n \) of the following operations can be answered in \( O(n^{1.8}) \) time:

   (a) Add a new quadratic \( a_i x^2 + b_i x + c_i \)

   (b) Query the max value of a quadratic at \( x \):

   \[
   \max_i a_i x^2 + b_i + c
   \]

   You may assume the result of the problem above.

5. Same as problem above, but in \( O(n^{1.1}) \) time.
6. Show that a sequence of \( n \) of the following operations on an array \( A[1 \ldots n] \), initially all 0, can be answered in \( O(n^{1.1}) \) time:

(a) Add some positive integer \( x \) of value at most \( n \) to all \( A[l \ldots r] \).
(b) Integer divide (rounding down) each of \( A[l \ldots r] \) by 2.
(c) Query for the sum of some range \([l, r]\).

7. Show that a sequence of \( n \) of the following operations on an array \( A[1 \ldots n] \) can be answered in \( O(n^{1.1}) \) time:

(a) Add \( x \) (possibly positive or negative) to all \( A[l \ldots r] \).
(b) Query, for some value \( b \), the maximum of \( b \cdot i + A[i] \) over all \( i \) in some range \([l, r]\) \( (l \leq i \leq r) \).

You may find split / merge binary trees useful [https://cstheory.stackexchange.com/questions/2132/split-or-merge-binary-search-trees-in-olog-n](https://cstheory.stackexchange.com/questions/2132/split-or-merge-binary-search-trees-in-olog-n), and also assume that ‘create a new copy of a binary tree’ is free.

8. Show that a sequence of \( n \) of the following operations on an array \( A[1 \ldots n] \) can be answered in \( O(n^{1.1}) \) time:

(a) Add \( x \) (possibly positive or negative) to all \( A[l \ldots r] \).
(b) Query for the max prefix sum of the subarray \( A[l \ldots r] \):

\[
\max_{j:l \leq j \leq r} \sum_{k=l}^{j} A[k].
\]

**CODE** Separate submission on Canvas, 5 points.

[https://dmoj.ca/problem/ccol17p2](https://dmoj.ca/problem/ccol17p2)

9. Given an algorithm that takes dimensions of a rectangle \((n, m)\), and outputs the minimum number of (axis-parallel) rectangle with side length ratio \(2:1\) that it can be partitioned into, in \( O((n + m)^{0.95}) \) time or faster.