CS166 HW1

Colin Man (colinman@stanford.edu), Kenny Xu (kenxu95@stanford.edu)

April 6, 2016

Problem 1 In order to calculate the largest value of k for which $2^k \leq j - i + 1$, we can create a table and go through each of the indices sequentially from 1 to $n+1$ since those are the only values that $j-i+1$ can take on $(0 \leq i, j \leq n)$. As we go through these indices, we keep a counter for the k value that the numbers correspond to and insert into the map, starting from $k = 0$ for 1 and incrementing k every time we pass a power of 2. We can then retrieve the k for any $j - i + 1$ with an $O(1)$ map lookup.

Problem 2

Problem 3

i We can construct hybrid structures of depth $k-1$ by breaking them up into block sizes $b_1, b_2, b_3, \ldots, b_{k-1}$ and use a sparse table RMQ on each level to achieve $< O(n \log n), O(1) >$ for that level. The preprocessing time of the hybrid structure is then

$$O(n+p(\frac{n}{b_1})+(\frac{n}{b_1})p(\frac{b_1}{b_2})+(\frac{n}{b_1})(\frac{b_1}{b_2})p(\frac{b_2}{b_3})+\cdots+(\frac{n}{b_{k-2}})p(\frac{b_{k-2}}{b_{k-1}})+(\frac{n}{b_{k-1}})p(b_{k-1}))$$

If we let

$$\begin{aligned} b_1 &= \log n \\ b_2 &= \log b_1 \\ &\ldots \\ b_{k-1} &= \log b_{k-2} \end{aligned}$$

Since $p(x) = x \log x$ using a sparse table RMQ, we can use the trick mentioned in lecture to reduce every term but the last as follows:

$$\begin{aligned} O((\frac{n}{b_{i-1}})p(\frac{b_{i-1}}{b_i})) &= O((\frac{n}{b_{i-1}})\frac{b_{i-1}}{b_i}\log(\frac{b_{i-1}}{b_i})) \\ &= O((\frac{n}{b_{i-1}})\frac{b_{i-1}}{\log b_{i-1}}\log b_{i-1}) \\ &= O((\frac{n}{b_{i-1}})b_{i-1}) \\ &= O(n) \end{aligned}$$

Upon reduction, we have a preprocessing time of:

$$\begin{aligned} O(n+(\frac{n}{b_{k-1}})(b_{k-1})\log b_{k-1}) &= O(n \log b_{k-1}) \\ &= O(n \log\log b_{k-2}) \\ &= O(n \log^{(k-1)} b_1) \\ &= O(n \log^{(k-1)} \log n) \\ &= O(n \log^{(k)} n) \end{aligned}$$

Lookup is performed similar to a two level hybrid query: at each of the k levels, we perform RMQ queries one level deeper on the blocks that contain the two boundary indices and perform RMQ on the blocks between them (at the current level). This gives a query time of $O(k) = O(1)$ since k is constant.

Thus, the time complexity of our hybrid structure is $< O(n \log^{(k)} n), O(1) >$ as required.

ii The increased query time arises from the fact that the query time is based on k as shown in part (i): every level of depth that we add requires another lookup per query. Thus, as k increases, the runtime increases. However, since k is a constant, all the hybrid structures still have a query time of $O(1)$, which does not contradict our result in (i).