CS180 — Algorithms and Complexity

HW#3 — Divide and Conquer

Due: 11:55pm Wednesday February 25, 2015

D. Stott Parker, Yuh-Jie Chen, Xiaoran Xu

stott@cs.ucla.edu, eyjchen@cs.ucla.edu, xrxu@cs.ucla.edu

Using CCLE, please upload your answers as a PDF document by the deadline.

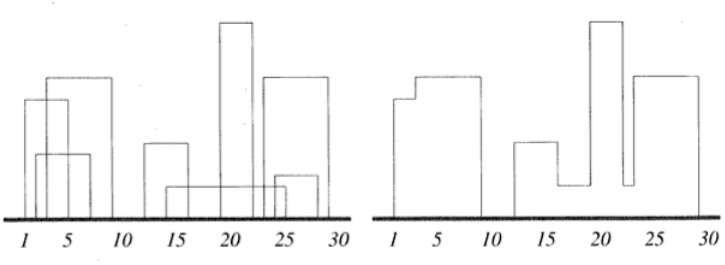
1. **Rectangles**

   Imagine that we are given a set of n rectangles resting on the x-axis. The i-th rectangle is described by 3 integer values $[\ell_i, h_i, r_i]$ specifying the 2D coordinates (ℓ_i, h_i) and (r_i, h_i) of the top two points in the rectangle. (Thus the coordinates of the other two points are $(\ell_i, 0)$ and $(r_i, 0)$.)

   These rectangles can *overlap*, in the sense that their rectangular 2D regions intersect. The *outline* of the rectangles a sequence of x-coordinates and heights giving the boundary (polygon) of their union. For example, using boldface to set off the height values, the $n = 8$ rectangles

   [1,**11**,5], [2,**6**,7], [3,**13**,9], [12,**7**,16], [14,**3**,25], [19,**18**,22], [23,**13**,29], [24,**4**,28],

   look as shown on the left in the following diagram:



   The outline of these rectangles, shown on the right of the diagram, is

   [1,**11**,3,**13**,9,**0**,12,**7**,16,**3**,19,**18**,22,**3**,23,**13**,29,**0**].

   (a) Design an $O(n \log n)$ algorithm that finds the outline of the rectangles.

   (b) Design an $O(n)$ algorithm that, given an outline, finds a rectangle of maximal area that fits within the outline. Implement your algorithm with a single left-to-right scan through the outline data.

2. **Interview Questions**

   (a) You are given a 3-pint container and a 5-pint container, and as much water as you want. Specify a sequence of filling and emptying steps that leave the containers holding exactly 7 pints of water.

   (b) There are many problems like the one above; it is from the 1916 Stanford-Binet IQ test. A common interview question uses 3, 5, 4. Design an algorithm that, given three small integers like these as input, finds a sequence with the minimum number of steps.

   (c) (Bi-Sum): You are given an array A of n integers, and another integer z, and you want to determine whether the array contains two elements a and b such that $a + b = z$.

      i. Give an algorithm that uses a min-heap and a max-heap to determine this in time $O(n \log n)$.

      ii. Give an algorithm that runs in time $O(n)$, assuming that A is given to you in sorted order.

   (d) (Tri-Sum): You are given an array A of n integers (possibly negative) and you want to determine whether the array contains three elements a, b, and c such that $a + b + c = 0$. Give an algorithm that solves this problem in $O(n^2)$ time.

   (e) (Missing Element): You are given an array of size n containing every number in $\{0, 1, 2, \ldots, n\}$ except for one. Give an algorithm to find the missing number in time $O(n)$, using only 1 memory cell that has $\lceil 2 \log_2 n \rceil$ bits. (For example, when $n = 50000$, the cell has 32 bits, and can represent numbers from 0 to $2^{32} - 1$.)

3. **Optimal Submatrix**
Suppose that there is a large square matrix of integer values that can be negative, zero, or positive. By a *submatrix* we mean a (contiguous) rectangular block of matrix elements. Please give your pseudocodes.

   (a) Find a maximal positive rectangular submatrix — i.e., a submatrix containing only positive values that has the most elements.

   (b) Find a maximum sum rectangular submatrix — i.e., a submatrix whose elements have maximal sum. (Hint: This is a generalization of the 'maxsum' problem discussed at the start of this course.).

   The matrix here is large, and so it is probably not possible (in the time you have available) to use a brute-force algorithm to find these optimal submatrices. You may need to use **dynamic programming**.

4. **Going Beyond the Master Theorem**
The Master Theorem is an excellent tool for thinking about complexity, but it is in not exhaustive or complete for solving recurrences. Here are some recurrences that do not fit the Master Theorem.

   (a) Suppose we have the recurrence

$$T(n) = \begin{cases} 1 & n = 1 \\ \sqrt{n}\, T(\sqrt{n}) + O(n) & n > 1 \end{cases}$$

   Assume $n = 2^{2^p}$, where p is a non-negative integer. Show that $T(n) = O(n \lg\lg n)$, where 'lg' $= \log_2$.
   Hint: Consider $S(n) = T(n)/n$.

   (b) Suppose we have the recurrence

$$T(n) = \begin{cases} 1 & n = 1 \\ 8 & n = 2 \\ 3T(n/2) + 4T(n/4) + 3n & n > 2 \end{cases}$$

   Give an expression for $T(2^p)$ for integer p, and determine the integer k for which $T(n) = \Theta(n^k)$.
   Hint: Consider $S(n) = T(n)/n$.

   (c) Suppose we have the recurrence $T(n) = T(n-1) + 1/(2n-1)$, and $T(0) = T(1) = 0$.
   Show that $T(n)$ has solution

$$\sum_{k=1}^{n} \frac{1}{2k-1} = \ln(\sqrt{n}) + O(1).$$

   Hint: express the sum on the left as a difference of two simpler sums: $\sum_{k=1}^{n} \frac{1}{2k-1} = \sum_{p=1}^{2n-1} \frac{1}{p} - (\cdots)$.

   (d) *General observation:* Two important linear recurrences are:

$$T(n) = a\,T(n-1) + b \quad \Rightarrow \quad T(n) = a^{n-1}\,T(1) + \left(\tfrac{a^{N-1}-1}{a-1}\right) b$$

$$T(n) = a\,T(n-1) + b\,n \quad \Rightarrow \quad T(n) = a^{n-1}\,T(1) + \left(\tfrac{(2a-1)a^{n-1}-(a-1)(n+1)-1}{(a-1)^2}\right) b$$

   To obtain the second solution, it helps to be able to derive results like the following.
   Problem: show that for $c \neq 1$,

$$\sum_{k=1}^{n} k c^{k-1} = \frac{nc^{n+1} - (n+1)c^n + 1}{(c-1)^2}.$$

   Specifically, show this by using derivatives:

$$\sum_{k=1}^{n} k c^{k-1} = \sum_{k=1}^{n} \frac{d}{dc}\left(c^k\right) = \frac{d}{dc} \sum_{k=1}^{n} c^k.$$

   General observation: as a result of this, we have a way to generate new 'Master Theorems' — a solution of a recurrence for $T(n)$ that involves a constant a can give us a solution of another recurrence for $T'(n) = \frac{d}{da} T(n)$.