COMS21103: Problems set 6

Dynamic programming

The starred problems below are **optional**, more challenging and hopefully interesting. If any of the problems seems unclear, please post a question on the forum.

1. Write out a formal proof in words of the "proof by picture" given in lecture of the recurrence for the size of the largest empty square in a monochrome image.

2. Write out the table generated by the dynamic programming algorithm IterED for computing the edit distance between the strings `SPAM` and `PROMO`. Hence write down an optimal sequence of edits changing the first string into the second.

3. Prove the claim made in lecture that the runtime of RecFib(n) is exponential in n. Hint: one way to do this is based on first proving that, for any n, the runtime of RecFib($n-1$) is at least as long as the runtime of RecFib($n-2$).

4. Let A be an array containing n integers. A subarray of A is a contiguous subset of elements of A. The maximum subarray sum $MSS(A)$ is defined to be the maximum, over all subarrays of A, of the sum of the elements in the subarray. For example, if $A = (1, -2, 3, -1, 2)$, $MSS(A) = 4$, achieved by the subarray $(3, -1, 2)$.

   (a) Let $MSS_j(A)$ be the maximum subarray sum that can be achieved by any subarray of A that finishes at position j. Give a recurrence expressing $MSS_j(A)$ in terms of $MSS_{j-1}(A)$.

   (b) Use your recurrence to give a memoized recursive dynamic programming algorithm for computing $MSS(A)$. Your algorithm should run in time $O(n)$.

   (c) Give a "bottom-up" algorithm which computes $MSS(A)$ without any recursive calls.

5. ($\star$) Give an algorithm which computes the n'th Fibonacci number F_n using $o(n)$ integer additions.

6. ($\star$) The Solitary Drinking Problem is defined as follows. There are n seats in a line at a bar, and a sequence of drinkers comes in to use them. The first drinker sits in seat 1. All subsequent drinkers sit in a seat which is as far as possible from anyone else (if there is more than one, picking the lowest-numbered one). To avoid having to make small talk, drinkers do not want to sit next to anyone else. When this can no longer be achieved, any new drinkers give up and leave.

   For example, with 7 seats and numbering drinkers in order of appearance, the situation at the end of this process, when no more drinkers can sit down, is as follows:

   

   For a bar with n seats, let $B(n)$ be the number of filled seats at the end of this process. The first values of $B(n)$, starting from $n = 1$, are $1, 1, 2, 2, 3, 3, 3, \ldots$

   Give a dynamic programming algorithm which efficiently computes $B(n)$. Code up your algorithm and use it to compute $B(1000000)$.

For more practice, there are many more dynamic programming exercises by Jeff Erickson online at `http://www.cs.uiuc.edu/~jeffe/teaching/algorithms/notes/05-dynprog.pdf`.