INSTITUTE OF COMPUTING - UNICAMP

Graduate Program

MO417A Design and Analysis of Algorithms

2015 - Semester 1 - Jorge Stolfi

Problem Set 03 - 2015-05-12

Nome	**RA**

Item													TOT
Nota													

1. The following algorithm returns the list y of the m smallest elements in a list $x = (x[0], x[1], \ldots, x[n-1])$ of n numbers (or all those numbers, if $m \geq n$), in increasing order. It also returns the number t of such elements (which is the smallest of m and n).

```
                Algorithm Smallest: given (n, x, m) returns (t, y)
                    t ← 0;
                    for i from 1 to n − 1 do
                        if t < m or ( t = m and x[i] < y[t − 1]) then
                            // Find the right place y[j] for x[i]:
                            j ← t;
                            while j > 0 and y[j − 1] > x[i] do;
                                if j < m then
[g]                                 y[j] ← y[j − 1];
                                endif;
                                j ← j − 1;
                            endwhile;
                            y[j] ← x[i];
                            if t < m then t ← t + 1; endif;
                        endif;
                    endfor;
                    return (t, y);
```

   (a) Describe the "best-case" and "worst-case" input list x, for general n and m, in the sense that they yield the minimum and maximum values for the count $g(n, x, m)$.

   (b) Give upper and lower bounds for the count $g(n, x, n)$ in the general case, as a function of n and m only.

   (c) What is the asymptotic class (in the $O/\Omega/\Theta$ notation) of $g(n, x, m)$, as a function of n and m only?

2. The following is Strassen's matrix multiplication algorithm, that computes the product C of two square matrices A and B, with $N = 2^k$ rows and columns. The matrix elements are denoted by $A[i, j]$ with i and j in $\{0..N-1\}$.

Algorithm *StrassenMM*: given (N, A, B) returns C
  if $N = 1$ then
    $C[0,0] \leftarrow A[0,0]\, B[0,0]$
  else
    split A into 4 matrices $A_{00}, A_{01}, A_{10}, A_{11}$ with $N/2$ rows and columns;
    split B into 4 matrices $B_{00}, B_{01}, B_{10}, B_{11}$, likewise;
    $M_1 \leftarrow \mathit{StrassenMM}(N/2, A_{00} + A_{11}, B_{00} + B_{11})$;
    $M_2 \leftarrow \mathit{StrassenMM}(N/2, A_{10} + A_{11}, B_{00})$;
    $M_3 \leftarrow \mathit{StrassenMM}(N/2, A_{00}, B_{01} - B_{11})$;
    $M_4 \leftarrow \mathit{StrassenMM}(N/2, A_{11}, B_{10} - B_{00})$;
    $M_5 \leftarrow \mathit{StrassenMM}(N/2, A_{00} + A_{01}, B_{11})$;
    $M_6 \leftarrow \mathit{StrassenMM}(N/2, A_{10} - A_{00}, B_{00} + B_{01})$;
    $M_7 \leftarrow \mathit{StrassenMM}(N/2, A_{01} - A_{11}, B_{10} + B_{11})$;
    $C_{00} \leftarrow M_1 + M_4 - M_5 + M_7$;
    $C_{01} \leftarrow M_3 + M_5$;
    $C_{10} \leftarrow M_2 + M_4$;
    $C_{11} \leftarrow M_1 - M_2 + M_3 + M_6$;
  join $C_{00}, C_{01}, C_{10}, C_{11}$ into a single matrix with N rows and columns;
  endif;
  return C;

(a) Count the total number $\alpha(N)$ of additions or subtractions of matrix elements executed by *one* call of *StrassenMM*(N, A, B), ignoring the recursive calls (but considering the computation of the arguments passed to the recursive calls). For example, to compute the arguments $A_{00} + A_{11}$ and $B_{00} + B_{11}$ of the call that computes M_1, the procedure must do $2(N/2)^2 = N^2/2$ element additions, when $N > 0$. Note that the number $\mu(N)$ of multiplication of matrix elements, again ignoring the recursive calls, is 1 if $N = 1$, and 0 if $N > 0$.

(b) Let $\alpha^*(N)$ be the total number of element additions or subtractions executed by *one* call of *StrassenMM*(N, A, B), *including* the additions executed inside the recursive calls. Write a recursive formula for $\alpha^*(N)$.

(c) Similarly, let $\mu^*(N)$ be the total number of element multiplications executed by *one* call of *StrassenMM*(N, A, B), *including* the multiplications executed inside the recursive calls. Write a recursive formula for $\mu^*(N)$.

(d) Find non-recursive formulas for $\alpha^*(N)$ and $\mu^*(N)$. (Hint: define $\hat{\alpha}(k) = \alpha^*(2^k)$, and similarly for μ^*.)

(c) What is the asymptotic class (in the $O/\Omega/\Theta$ notation) of $\alpha^*(N)$ and $\mu^*(N)$, as a function of N?

3. Consider a heap structure of n numbers, stored in a list $x = (x[0], x[1], \ldots, x[n-1])$, where $x[i] \le x[2i+1]$ and $x[i] \le x[2i+2]$, for any index i such that the elements exist. Write the procedures described below, that modify the heap preserving the above invariant. For each procedure, give the maximum number of comparisons performed by the procedure, as a function of n only.

   (a) *HAdd* that adds one new value v to the heap, incrementing n. (Assume that there is space in the list x for one more element.)

   (b) *HDelete* that, given an index k in $\{0..n-1\}$, deletes element $x[k]$ from the heap, decrementing n.

   (c) *HModify* that, given an index k in $\{0..n-1\}$, and a new value u, replaces element $x[k]$ by u, without changing n. (Try to find a method that is more efficient than $n \leftarrow \mathit{HDelete}(n, x, k)$ and $n \leftarrow \mathit{HAdd}(n, v, v)$.)

   (d) *HRank* that, given an index k in $\{0..n-1\}$, counts how many elements $x[j]$ are strictly less than $x[k]$

4. Consider a *ternary heap* structure with n numbers $x = (x[0], x[1], \ldots, x[n-1])$, where $x[i] \le x[3i+1]$, $x[i] \le x[3i+2]$, and $x[i] \le x[3i+3]$, for any index i such that the elements exist. Show how to modify the *HeapSort* algorithm and its associated routines to use a ternary heap instead of a binary one. Which version would be faster for (say) a few thousand input numbers?

5. Consider the *QuickSort* algorithm with the following modification *M3Split* to the *Split* procedure: instead of picking one element of the array to be the separating value v, *M3Split* takes three elements $x[r]$, $x[s]$, and $x[t]$ (say, $x[0]$, $x[n-1]$, and $x[\lfloor n/2 \rfloor]$), and lets v be their median. (I. e., it finds the maximum and minimum of the three, and lets v be the one that is left out.)

   It can be shown that, if all the input elements are distinct and all input orderings are equally likely, the probablity that v will end up at some index m is $\Pr(m) = m(n-1-m)/A_n$ where $A_n = n(n^2-3n+2)/6$. (Note that this function is a quadratic function of m that is zero when $m = 0$ or $m = n-1$, and is maximum when $m = \lfloor n/2 \rfloor$.)

   Modify the analysis of expected number of element comparisons $t^*(n)$ performed by the average-case of the *Quicksort* algorithm, assuming this change to the *Split* routine. Compare it with the same number $t(n)$ of the original algorithm. Note that the expected number of comparisons in the recursive calls (not counting the comparison inside *M3Split*) will be $\sum_{m=1}^{n-2} \Pr(m)(t^*(m-1) + t^*(n-1-m))$ instead of $(1/n)\sum_{m=0}^{n-1}(t(m-1) + t(n-1-m))$. You may assume that $t^*(n) \le c^* n \log n$, while $t(n) \le cn \log n$, for some constants c^* and c and for sufficiently large n.