INSTITUTE OF COMPUTING - UNICAMP

Graduate Program

MO417A Design and Analysis of Algorithms

2015 - Semester 1 - Jorge Stolfi

Problem Set 02 - 2015-04-05

Nome	**RA**

Item												TOT
Nota												

1. The two algorithms below solve the same problem: given an integer $n \geq 0$ and the letters $x = (x[0], x[1], .. x[n-1])$ of some word (uppercase only, '`A`' to '`Z`', without accent marks), returns a letter that occurs twice in the word; or '`*`' if all the letters are distinct.

   For example, if $x = ($'`B`', '`A`', '`N`', '`A`', '`N`', '`A`'$)$, they return '`A`'; if $x = ($'`G`', '`A`', '`R`', '`L`', '`I`', '`C`'$)$, they return '`*`'. (As before, the notation $[v]$ before a statement means "v is the number of times that this statement gets executed when the algorithm is executed once".)

```
     Algorithm A(n, x)
         for i from 1 to n − 1 do
             for j from 0 to i − 1 do;
[f]              if x[j] = x[i]
                     return x[i];
                 endif;
             endfor;
         endfor;
         return '*';
```

```
     Algorithm B(n, x)
         for i from 0 to n − 2 do
             for j from i + 1 to n − 1 do;
[f]              if x[j] = x[i]
                     return x[i];
                 endif;
             endfor;
         endfor;
         return '*';
```

   For each algorithm:

   (a) Find a "best-case" input word x with $n = 100$ letters, that minimizes the count f. What is the value of f for that word?

   (b) Find a "worst-case" input word x with $n = 100$ letters, that maximizes the count f. What is f for that word?

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

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

2. Consider the algorithm below, that finds the maximum and minimum of a list of n numbers at the same time:

```
          Algorithm MinMax(n, x)
              i ← 0;
              while i < n do
[f]               if i + 1 < n
                      if x[i] < x[i + 1]
                          a ← x[i]; b ← x[i + 1];
                      else
                          a ← x[i + 1]; b ← x[i];
                      endif;
                  else
                      a ← x[i]; b ← x[i]
                  endif;
                  if i = 0 or a < xmin
[g0]                  xmin ← a;
                  endif;
                  if i = 0 or b > xmax
[g1]                  xmax ← b;
                  endif;
                  i ← i + 2;        // ⇐ note!
              endwhile;
              return xmin, xmax;
```

Assuming that n is even (to simplify the analysis), determine the following quantities as a function of n:

(a) the number of times f that the main loop is executed.

(b) The total number h of comparisons between elements of the list, namely "$x[i] < x[j]$", "$a < xmin$", and "$b > xmax$".

(c) An upper (worst case) bound for g_0 and g_1, the number of times that *xmin* and *xmax* are updated, respectively.

(d) A lower (best-case) bound for g_0 and g_1.

(e) The expected values of g_0 and g_1, assuming that the elements of x are all distinct and all possible orders are equally likely.

3. Find upper and lower bounds for the following summations, using the integral method. Recall that $x - 1 < \lfloor x \rfloor \leq x$, $\sqrt{x} = x^{1/2}$, and $\log_a x = \ln x / \ln a$ for any x and a such that the formulas are defined. Recall also that $\int \ln x dx = x(\ln x - 1)$.

   (a) $s(n) = \sum_{i=1\ n} \sqrt{i+2}$

   (b) $s(n) = \sum_{i=1\ n} \log_2 i$

   (c) $s(n) = \sum_{i=1\ n} \lfloor n^2/i^2 \rfloor$

4. The factorial function is defined as $f(n) = n! = \prod_{i=1}^{n} i = 1 \times 2 \times 3 \times \cdots \times n$.

   (a) Write a summation formula for the natural logarithm of the factorial, $g(n) = \ln f(n)$.

   (b) Find upper and lower bounds for $g(n)$, using the integral method.

   (c) From those bounds, get upper and lower bounds for the factorial $f(n)$.

   (d) Based on these bounds, what is the asymptotic class of $f(n)$?

5. Consider the function f, from the natural numbers $\mathbb{N}$ to $\mathbb{N}$, defined recursively by

$$f(n) = \begin{cases} 0 & \text{if } n = 0, \\ (n \bmod 10) + f(\lfloor n/10 \rfloor) & \text{if } n > 0. \end{cases} \tag{1}$$

   (a) What is $f(235)$?

   (b) Determine the maximum and minimum values of $f(n)$, for any number n with k decimal digits as a function of k alone.

   (c) What is the asymptotic class of $f(n)$ as a function of k? And as a function of n?

6. Consider the function g, from the natural numbers $\mathbb{N}$ to the integers $\mathbb{Z}$, defined recursively by the homogeneous linear recurrence

$$g(n) = \begin{cases} 5 & \text{if } n = 0, \\ 0 & \text{if } n = 1, \\ g(n-1) + 6g(n-2) & \text{if } n \geq 2. \end{cases} \tag{2}$$

   (a) Tabulate the value of $g(n)$ for a few values of n.

   (b) Write the characteristic polynomial of the recurrence.

   (c) Dtermine the roots r_i of the polynomial.

   (d) Dtermine the explicit formula for $g(n)$.

   (d) What is the asymptotic class of g as a function of n?

7. The following recursive algorithm $Permute(n, x, k, use)$ generates all permutations of the elements of a list $x = (x[0]..x[n-1])$ of n arbitrary elements, leaving the first k elements fixed.

In particular, if $k = 0$ the procedure generates all permutations; if $k = n$, it generates a single permutation, namely the given list x itself. For each generated permutation, *Permute* calls the function *use*, provided by the calling program, with arguments (n, x). The elements of x are rearranged while call is in progress, but, at the end, they will be restored to their original order.

For example, if $n = 5$, $x = (07, 97, 27, 35, 45)$, and $k = 2$, then $Permute(n, x, k, use)$ will call

$$use(5, (07, 97, 27, 35, 45))$$
$$use(5, (07, 97, 27, 45, 35))$$
$$use(5, (07, 97, 35, 27, 45))$$
$$use(5, (07, 97, 35, 45, 27))$$
$$use(5, (07, 97, 45, 35, 27))$$
$$use(5, (07, 97, 45, 27, 35))$$

and the list x will again contain $(07, 97, 27, 35, 45)$.

```
          Algorithm Permute(n, x, k, use)
             if k = n
[u]              use(n, x);
             else
                 for i from k to n − 1 do
[f]                  swap x[k] ↔ x[i];
                     Permute(n, x, k + 1, use);
                     swap x[k] ↔ x[i];
                 endfor;
             endif;
```

(a) Assuming that the algorithm is correct, determine the number u of times the procedure *use* is called (including inside the recursive calls) when *Permute* is called with generic n and k.

(b) Write a recursive formula for $f(n, k)$, the total number of times that two elements are swapped (including inside the recursive calls) when *Permute* is called with generic n and k.

(c) Find a non-recursive formula for $f(n, k)$. The formula should depend only on the difference $m = n - k$, not on n an k explicitly.

(d) Prove that the formula satisfies the recursive definition, by induction on m.

8. The algorithm *Scramble* below rearranges a list $x = (x[0], .. x[n-1])$ of n elements according to a permutation p. The permutation is a list $(p[0], .. p[n])$ of n integer indices, all distinct, all between 0 and $n-1$. The effect of the algorithm is to put in $x[i]$ the value of $x[p[i]]$, for all i in $0.. n-1$.

   For example, if we have $x = (05, 15, 25, 35, 45, 55, 65)$ and $p = (3, 4, 6, 1, 0, 5, 2)$ on input, then we will have $x = (35, 45, 65, 15, 05, 55, 25)$ at the end. (Note that the "obvious" algorithm "for i in $0.. n-1$ do $x[i] \leftarrow x[p[i]]$" would not work; try it.)

```
        Algorithm Scramble(n, x, p)
            for i in 0.. n − 1 do
                if p[i] ≥ 0
                    // Start of a new cycle:
                    t ← x[i]; j ← i;
                    while p[j] ≠ i;
[f]                     k ← p[j];
                        x[j] ← x[k];
                        p[j] ← p[j] − n; // Makes p[j] negative.
                        j ← k;
                    endwhile;
                    x[j] ← t; p[j] ← p[j] − n;
                endif;
            endfor;
            // Restore the permutation p:
            for i in 0.. n − 1 do
                p[i] ← p[i] + n;
            endfor;
```

   Determine the number of times f that the inner loop is executed, as a function of n. Hint: simulate the example above to understand how the algorithm works. What is the asymptotic class of f as a function of n?