Departamento de Ciência de Computadores FCUP

Algoritmos (CC4010) 2021/22

Exam (28.01.2022) *2 hours*

1. [5.0] Consider the function CHECK(x, n), where x is an array of integers and n is a positive integer. Assume $x[1]$ is the first element.

```
CHECK(x, n)
1  i = 2
2  while i ≤ n do
3      j = i − 1
4      while j > 0 do
5          if x[j] = x[i] then return 0
6          j = j − 1
7      i = i + 1
8  return 1
```

a) State the decision problem that CHECK(x, n) solves.

b) State an invariant for loop 4-6 and an invariant for loop 2-7 that allow us to conclude that the function is correct. Explain how they allow it.

c) Which is the time complexity of loop 4-6 in the best case and in the worst case for a given i? You must state it as a function of i. Explain, giving a characterization of best case and worst case instances.

d) Explain why the time complexity of CHECK(x, n) is not $\Theta(n)$ neither $\Theta(n^2)$ and why it could be described as $O(n^3)$ but we would better not do it that way.

2. [3.0] Consider the *minimum cardinality vertex cover* problem in undirected graphs $G = (V, E)$.

a) Using pseudocode, write a polynomial-time algorithm that, given $G = (V, E)$, finds a vertex cover C such that $|C|/|C^\star| \leq 2$, where $C^\star$ is an optimal vertex cover for G. Explain why it is a polynomial time algorithm, why it yields a vertex cover, and why it guarantees that approximation factor.

b) Explain why for **acyclic graphs** (i.e., either trees or forests), an optimal vertex cover can be found in polynomial time (giving such an algorithm), whereas for generic undirected graphs it cannot unless P = NP.

3. [4.0] Consider the function PARTITION, x being an *array* of n integers, $x[1], \ldots, x[n]$, and a and b integers such that $1 \leq a \leq b \leq n$.

```
PARTITION(x, a, b)
1.  z = x[a]
2.  j = a
3.  for i = a + 1 to b do
4.      if x[i] < z then
5.          Exchange x[i] with x[j + 1]
6.          j = j + 1
7.  Exchange x[a] with x[j]
8.  return j
```

a) Write the function QUICKSORT(x, a, b), that calls PARTITION and sorts the $x[a..b]$ in **increasing** order, that is, in the end $x[a] \leq x[a + 1] \leq \ldots \leq x[b]$.

b) How does PARTITION(x, a, b) change $x[a..b]$? Which is the invariant of loop 3-6 that allows us to conclude that? Explain why QUICKSORT(x, a, b) is correct.

c) How can we change the code to implement *randomized quicksort* instead?

4. [1.0] Explain how the function *quickselect(x,a,b,k)* introduced in the course can be used to find the median of an array of n integers, for odd n. Which would be its expected time complexity?

Solve three of the remaining problems

5. Consider the problem of guarding an n-vertex simple polygon with guards on vertices, introduced in the course. It is known that the number of pieces of the partition of the polygon induced by the visibility regions of all vertices is $O(n^3)$.

a) Explain the main steps of the reduction of this problem to the *minimal cardinality set cover* proposed by Ghosh and why the reduction is exact (i.e., preserves the optimal solutions).

b) Explain why the $(1 + \log_e N)$-approximation algorithm for minimum cardinality **set cover**, where N is the number of elements in its universe, gives a $O(\log_e n)$-approximation algoritm for the guarding problem. Here $e = 2.718281828\ldots$ is the Euler number (or Neper constant).

6. We want to compute the product of two numbers X and Y, represented in binary with n bits. The product will be represented with $2n$ bits. Assume that n is a power of 2. Recall that $X = 2^{n/2}A + B$ and $Y = 2^{n/2}C + D$, where A, B, C and D have $n/2$ digits.

a) Explain the idea of **Karatsuba's algorithm** that runs in $o(n^2)$ time (i.e., sub-quadratic time) .

b) Justify its correctness and its time complexity, using the **Master Theorem**. You should explain the details.

7. Given two strings x and y, whose length is n and m, consider the **Minimum Edit Distance** problem that rewrites x into y, using insertions, deletions and substitutions. The cost per operation is $c \in \mathbb{Z}^+$.

a) Write and explain the recurrence that supports a dynamic programming (DP) algorithm for the problem.

b) Using pseudocode, write a function based on DP to find the optimal cost and return a solution.

8. Consider the **interval scheduling** problem described in class.

a) Explain what is given and what we are looking for in this problem.

b) Why the strategy *minimum duration first* can produce sub-optimal solutions?

c) Using pseudocode write an $O(n \log n)$ time algorithm that yields an optimal solution always, where n is the number of intervals.

d) Justify the time complexity and the correctness of the algorithm.

9. Prove that the **amortized cost per operation** when we increment a binary counter from 0 to n by adding one unit each time is $O(1)$. The counter is defined by an array of k bits and the operations should be carried out modulo 2^k. The real cost of each flip is 1 (either a flip from 0 to 1 or a flip from 1 to 0). The total cost is the number of flips in the n increments. Use either the *aggregation method* or the *accounting method*.

Master theorem:

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence $T(n) = aT(n/b) + f(n)$, where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log_2 n)$.
3. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$, for some constant $\varepsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.