Fourth written examination of

Algoritmos e Estruturas de Dados

November 28, 2016 **Duration: no more than 45 minutes**

Name:

Student number:

6.0 $\boxed{\textbf{1:}}$ [Divide and conquer] Let $T(n)$ be the effort required to solve a problem of size n, let $f(n)$ be the effort needed to perform steps divide and combine (again for a problem of size n), and let a be the number of subproblems of size n/b that are done in the conquer step. Assigning an effort of 1 to problems of size $n \leq n_0$, we have

$$T(n) = \begin{cases} 1 & \text{for } n \leq n_0; \\ aT\left(\frac{n}{b}\right) + f(n) & \text{otherwise.} \end{cases}$$

The master theorem states that:

- If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
- If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = O(n^{\log_b a} \log n)$
- If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ and if $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta\big(f(n)\big)$.

Consider the recursion $T(n) = 3T\left(\frac{n}{2}\right) + 10n \log n$. What can we say about how $T(n)$ grows?

Answer:

7.0 **2:** The data type `T` implements a data container that stores pointers to binary tree nodes. It has a method `put` to put a tree node pointer in the container, a method `get` that retrieves (and removes) a tree node pointer from the container, and a method `isEmpty` that returns `false` if and only if the data container is not empty. The following C++ function visits all nodes of a binary tree.

```
void visit_all(tree_node *root)
{
  tree_node *n;
  T c;  // this creates an empty container

  c.put(root);
  while(c.isEmpty() == false)
    if((n = s.get()) != nullptr)
    {
      visit(n);
      s.put(n->left);
      s.put(n->right);
    }
}
```

2.0 **2a)** If the data container is a stack, by what order are the tree nodes visited?

2.0 **2b)** If the data container is a queue, by what order are the tree nodes visited?

3.0 **2c)** If the data container is a priority queue, by what order are the tree nodes visited?

Answer:

7.0 $\boxed{\textbf{3:}}$ The following function computes recursively the number of paths starting at $(0,0)$ and ending at (X,Y). The paths always stay inside the rectangle $0 \le x \le X$, $0 \le y \le Y$. Steps are made using knight moves (a chess piece). **<u>At most</u>** B steps can move in a negative x or y direction.

```
static long count = 0;

void count_paths(int x,int y,int b)
{ // to count all paths: count = 0; count_paths(0,0,0);
  if(x == X && y == Y)
    count++;                              // count
  if(x + 1 <= X && y + 2 <= Y)
    count_paths(x + 1,y + 2,b);           // move in the +1,+2 direction
  if(x + 1 <= X && y - 2 >= 0 && b + 1 <= B)
    count_paths(x + 1,y - 2,b + 1);       // move in the +1,-2 direction
  if(x - 1 >= 0 && y + 2 <= Y && b + 1 <= B)
    count_paths(x - 1,y + 2,b + 1);       // move in the -1,+2 direction
  if(x - 1 >= 0 && y - 2 >= 0 && b + 1 <= B)
    count_paths(x - 1,y - 2,b + 1);       // move in the -1,-2 direction
}
```

Write code that counts the number of paths using dynamic programming. What is the computational complexity of your solution?

Answer: