Algoritmos e Complexidade

MIEI, LCC 2º ano

3 de Novembro de 2018 – Duração: 90 min

1. Considere a função que determina se dois *arrays* de inteiros são iguais.

   (a) Determine um invariante `I` e um variante `V` apropriados para provar a **correcção total** do algoritmo.

   (b) Mostre que o programo é correcto apresentando **apenas** as condições de verificação do programa.

```
// 0 <= n
int eq (int u[], int v[], int n)
{
    i = 0;
    result = 1;
    while (i<n && result == 1)
       // I, V
    {
      if (u[i] != v[i]) { result = 0; }
      i = i+1;
    }
}
// (result == 0 || result == 1) &&
// (result == 1 <==>
//  forall x:: 0 <= x < n ==> u[x] == v[x])
```

   (c) De forma a analisar o tempo de execução desta função:

      i. Identifique o melhor e pior casos do tempo de execução.

      ii. Assumindo (i) que ambos os arrays se encontram preenchidos de forma aleatória, e (ii) que o tipo `int` corresponde aos inteiros de K bits (e que por isso a probabilidade de dois inteiros aleatórios serem iguais é de $\frac{1}{2^k}$), apresente, justificando, um somatório que permita estimar o **caso médio** do número de comparações entre elementos dos dois *arrays*.

2. Considere a seguinte definição de uma função que calcula o quadrado de um número natural (inteiro não negativo). Analise a complexidade desta função em função do número de bits necessários para representar o argumento. Note que se são necessários N bits para representar um número, então esse número pertence ao intervalo $[2^{N-1}..2^N - 1]$.
   Faça a sua análise contando o número de operações aritméticas (adições, divisões e multiplicações) efectuadas. Para isso escreva uma recorrência que traduza o comportamento da função **no pior caso** e apresente uma solução dessa recorrência.

```
int square (int x) {
  int r, m=x/2;
  if (x==0)
    r = 0;
  else if (x%2 == 0)
    r = 4* square (m);
  else
    r = 4*(square (m) + m) + 1;

  return r;
}
```