Teste 29 Novembro 2021

Algoritmos e Complexidade

Universidade do Minho

Questão 1 [12 valores]
Considere a função `firstOdd` em baixo, que calcula o índice da *primeira ocorrência de um número ímpar* num array.

```
int firstOdd (int v[], int N) {
  //pre: N>=0
  int i=0;
  while (i<N && v[i]%2==0) i++;

  if (i==N) r = -1;

  else r = i;
  //pos: (r==-1 && forall_{0 <= k < N} ...) || (0<=r<N  && ...
)
  return r;
}
```

1. Complete a definição da pós-condição da função, preenchendo os espaços `...` .

2. Apresente um invariante para o ciclo que seja adequado para provar a correcção parcial da função face à especificação acima.

3. Apresente a análise de melhor e pior caso do tempo de execução, contando o número de comparações `v[i]%2==0` efectuadas.

4. Efectue agora a análise do caso médio de execução da função, calculando o valor esperado do número de acessos ao array assumindo aleatoriedade no preenchimento do array.

Questão 2 [6 valores]

A seguinte função `diferentes` conta de forma recursiva o número de elementos diferentes presentes num array.

```
int elem (int x, int u[], int N) {
  // pre: ...
  if (N==0) r = 0;
  else if (x==u[0]) r = 1;
       else r = elem(x, u+1, N-1));
  // pos: ...
  return r;
}

int diferentes (int v[], int N) {
  int r = 0;
  if (N!=0) {
    r = diferentes (v+1, N-1);
    if (!elem (v[0], v+1, N-1) r++;
  }
  return r;
}
```

1. Apresente uma especificação (i.e. uma pré-condição e uma pós-condição) para a função auxiliar `elem`.

2. Efectue a análise do tempo de execução da função `diferentes` no melhor e no pior caso, escrevendo e resolvendo todas as recorrências necessárias.

Questão 3 [2 valores **]

Considere a seguinte função:

```
void loop (int v[], int N, int k) {
    int tmp, cont = 1;
    while (cont) {
        cont = 0;
        tmp = v[N-1];
        for (i=N-1; i>0; i--)
            if (v[i] != v[i-1]) {
                v[i] = v[i-1];
                if ((!cont) && (i != N-1)) cont = 1;
            }
        if (v[0]==tmp)  v[0] = (v[0]+1)%k;
    }
}
```

1. Mostre que a execução pode não terminar no caso geral, e que termina sempre se k>N (apresentando para isso um variante adequado).

2. Considerando que k>N, efectue a análise de melhor e de pior caso do tempo de execução da função.