Teste 9 Novembro 2019

Algoritmos e Complexidade

Universidade do Minho

Questão 1 [5 valores]
Considere a função `maxarray` em baixo. Sejam

- $\phi \equiv 0 \leq a \leq b$
- $\psi \equiv a \leq r \leq b \ \ \wedge \ \ \forall\, k.\, a \leq k \leq b \rightarrow v[k] \leq v[r]$

Apresente um **invariante e um variante para o ciclo**, que permitam provar a correcção total da função face à pré-condição ϕ e à pós-condição ψ, em que r denota o valor devolvido pela função.

```
int maxarray(int v[], int a, int b) {
  int i = a+1, r = a;
  while (i <= b) {
    if (v[i] > v[r]) { r = i; }
    i = i+1; }
  return r;
}
```

Questão 2 [3 valores]
Considere o cálculo de máximos de um array de comprimento N por uma *sliding window* (janela deslizante) de comprimento k. A ideia é calcular os máximos de todas as subsequências (contíguas) de comprimento k, guardando-os num array de resultados.
Por exemplo com N=5, k=3, o resultado para o array [50, 10, 30, 20, 0] seria [50, 30, 30], correspondente aos máximos de [50, 10, 30], [10, 30, 20], e [30, 20, 0].
A função seguinte resolve recursivamente este problema, utilizando como auxiliar a função `maxarray` da Questão 1.

```
void MaxWindowRec(int v[], int r[], int N, int k) {
  if (N >= k) {
    r[0] = v[maxarray(v, 0, k-1)];
    MaxWindowRec(v+1, r+1, N-1, k);
  }
}
```

Escreva uma especificação (**pré- e pós-condição**) para esta função.

Questão 3 [5 valores]
(i) Escreva e resolva uma **recorrência** para o número de comparações $T(N, k)$ efectuadas entre elementos do array, em função de N e de k.

(ii) Tendo agora em conta os diferentes valores que k pode tomar, e assumindo que todos esses valores ocorrem com igual probabilidade, apresente um somatório que permita calcular o **caso médio** do tempo de execução $T(N)$ de `MaxWindowRec`.

Questão 4 [5 valores]
Considere um algoritmo de inserção ordenada numa lista (crescente), com uma particularidade: são apagados os nós iniciais da lista contendo valores inferiores ao que está a ser inserido. Por exemplo, a inserção de 30 na lista [10, 20, 40, 50] resulta na lista [30, 40, 50]. A função seguinte implementa este algoritmo em C.

```
node *insert_rem (node *p, int x) {
  node *new = malloc(sizeof(node)); new->value = x;
  while (p && x > p->value)
    { aux = p; p = p->next; free (aux); }
  new->next = p;
  return new;
}
```

(i) Analise o tempo de execução assimptótico de `insert_rem`, identificando o **pior e o melhor caso**.

(ii) Em termos amortizados a operação de inserção da questão anterior executa em tempo constante. Efectue a sua **análise agregada** considerando a sequência de inserções 20, 70, 60, 30, 40, 50, 10, 80 (partindo de uma lista vazia). Considere que o custo real de cada inserção/remoção efectuada à cabeça da lista é 1, por isso a inserção de 30 na lista [10, 20, 40, 50] tem custo 3. Apresente ainda uma **função de potencial** apropriada sobre as listas, e calcule a partir dela o custo amortizado constante desta operação `insert_rem`.

Questão 5 [2 valores *]
Pretende-se reimplementar o cálculo de máximos em janela deslizante (Questões 2 e 3), agora em tempo $\mathcal{O}(N)$, não dependendo de k, mas podendo para isso utilizar-se uma estrutura de dados auxiliar ocupando espaço em $\mathcal{O}(k)$. Descreva a sua solução de forma clara, justificando o tempo de execução no pior caso (pode utilizar pseudo-código se assim entender).