Tópicos Avançados em Algoritmos - Prova III

1. **Verdadeiro ou falso?**
   Diga se cada uma das seguintes afirmações é verdadeira ou falsa (não é necessário justificar)

   (a) **O produto de matrizes é associativo.**
   (b) **Sendo A, B e C matrizes[1], o custo (número de multiplicações elementares) do produto $(A\times B)\times C$ é igual ao custo do produto $A\times(B\times C)$.**
   (c) **O comprimento máximo de uma sequência consecutiva comum às "strings" s e t pode ser determinado (no pior caso) em tempo polinomial.**
   (d) **O comprimento máximo de uma sequência (não necessariamente consecutiva) comum às "strings" s e t pode ser determinado (no pior caso) em tempo polinomial.**
   (e) **Usando convenientemente a Programação Dinâmica conseguem-se por vezes algoritmos polinomiais para problemas completos em NP (como o problema de decisão da mochila).**
   (f) **O número de comparações efectuadas pelo algoritmo de "merge" de 2 vectores com m e n elementos está necessariamente compreendido entre $\min\{m,n\}$ e $m+n-1$.**
   (g) **Não existe qualquer algoritmo de ordenação baseado em comparações que tenha um tempo de execução no pior caso de ordem $O(n\sqrt{\log n})$, onde n é o número de elementos do vector.**

2. **Multiplicação óptima de matrizes**
   O número de formas de colocar parêntesis num produto $M_1M_2\ldots M_n$ de n matrizes pode ser definido pela recorrência $f(1)=1$, $f(n)=\sum_{1\le p<n} f(p)f(n-p)$ para $n\ge 2$.

   (a) **Usando indução matemática em n mostre que[2] para $n\ge 7$, $f(n)\ge 2^n$. Sugestão. Pode usar o facto $f(7)=132$.**
   (b) **A utilização da Programação Dinâmica permite obter um algoritmo polinomial em n para determinar a "parentização" óptima de um produto de n matrizes. Descreva em linhas gerais esse algoritmo e mostre que a sua eficiência é $O(n^3)$.**

3. **Procurar x**
   Considere o problema de procurar um valor x num vector `v[1..n]` (não necessariamente ordenado) com n elementos distintos. A resposta é o índice `i` tal que `v[i]`$=x$, ou `-1` se x não ocorre em `v`. Utiliza-se o modelo externo dos dados com a análise no pior caso; todos os acessos aos dados são comparações entre x e elementos do vector (cada comparação tem 3 resultados possíveis). Seja $c(n)$ o número de acessos aos dados (comparações) no pior caso.

   (a) **Usando o Princípio da Informação necessária, determine um minorante de $c(n)$.**
   (b) **Determine o majorante seguinte: $c(n)\le n$.**
   (c) **Mostre que se verifica o minorante[3] $c(n)\ge n$.**
   (d) **Suponha agora que o vector está ordenado e que se usa a pesquisa binária. Mostre que o número de comparações $c(n)$ satisfaz $c(n)\le\log(n+1)$, onde se supõe que n é da forma $n=2^p-1$. Que conclui neste caso (pesquisa binária) do minorante obtido na alínea 3a?**

4. **Comparações no "merge"**
   Considere o algoritmo de `merge(u[1..m],v[1..n])` onde todos os $m+n$ elementos dos vectores são distintos. Usa-se o modelo externo dos dados, com a análise no pior caso; todos os acessos aos dados têm a forma "`u[i]<v[j]?`".

   (a) **Explique porque é que o número de respostas possíveis é igual ao número de sequências de $m+n$ caracteres `u` ou `v`, com m caracteres `u` e n caracteres v.**
   Um exemplo (com $m=4$ e $n=3$): `merge([10,20,80,100],[30,40,90])` $\Rightarrow$ `uuvvuvu`
   (b) **Usando o Princípio da Informação Necessária determine um minorante do número de comparações efectuadas entre elementos de `u` e elementos de `v`.**
   (c) **Supondo que $m=n$ e usando a fórmula de Stirling, $n!\approx\sqrt{2\pi n}(n/e)^n$, determine uma forma fechada (sem usar factoriais nem combinações) para o minorante obtido na alínea anterior.**

5. **Transformando uma palavra noutra**
   Suponha que uma "string" `s` (com m caracteres) é transformada numa uma "string" `t` (com n caracteres) usando apenas as operações *inserir caracter* e *eliminar caracter*. Por exemplo, `ele -> bele -> abele -> abel -> abela -> bela`. O custo desta transformação é de 5. O *custo mínimo*, ou *distância*, entre estas palavras é de 3, $d(\texttt{"ele"},\texttt{"bela"})=3$, pois existe uma transformação de custo 3.
   Mostre que a distância entre duas "strings" s e t é dada por

   $$d(s,t)=(m-\texttt{ms}(s,t))+(n-\texttt{ms}(s,t))=m+n-2\times\texttt{ms}(s,t)$$

   onde $\texttt{ms}(s,t)$ é o comprimento da maior sequência (não necessariamente consecutiva) comum às "strings" s e t. Sugestão. Considere a história de cada caracter (sobrevivência, eliminação, inserção) na transformação óptima $s\to t$.

---

[1] Tais que o número de colunas de A é igual ao número de linhas de B e o número de colunas de B é igual ao número de linhas de C.

[2] Este resultado implica que qualquer algoritmo que analise todas as formas de colocar parêntesis com vista a determinar a "parentização" óptima é exponencial em n.

[3] Este minorante é, portanto, exacto. Assim, o minorante obtido na alínea 3a é muito fraco.