Armando Matos, 2008-2009

Tópicos Avançados em Algoritmos - Folha de exercícios 03

Nesta folha incluem-se 2 problemas para os quais existem algoritmos muito simples e eficientes, mas em que há tendência para implementar algoritmos mais complicados e ineficientes... O primeiro problema é o conhecido "problema da celebridade" (este problema é formulado de outro modo no exercício 2); o segundo problema é o da "maioria absoluta". Inclui-se ainda uma outra questão (problema 4) que consiste basicamente no "desenho" de um código binário em que há apenas um bit que muda ao passar de um elemento para o seguinte.

1. **A celebridade**
   O problema da celebridade é o seguinte: numa sala estão n pessoas, uma das quais é uma "celebridade". Uma *celebridade* é uma pessoa que todas as outras conhecem, mas que não conhece nenhuma das outras (imagine-se, por exemplo, o presidente Bush na turma teórica desta disciplina). Em termos da matriz de adjacências m do grafo associado ao predicado "conhece" é $m[x, y] = 1$ sse x conhece y. Então procura-se um inteiro x tal que a linha x de m só contenha 1's (a menos do valor correspondente à coluna x que é irrelevante) e tal que a coluna x só contenha 1's (a menos do valor correspondente à linha x).

   – Mostre que há no máximo uma celebridade.

   – Sabe-se que existe uma celebridade. Uma pessoa externa (que não conhece nenhuma das pessoas que estão na sala) entra na sala e pode fazer perguntas a uma qualquer pessoa x, perguntando se conhece outra qualquer pessoa y; por outras palavras, pode perguntas "$m[x, y] = 1$?". Especifique um algoritmo que indica como a pessoa externa vai proceder para descobrir em não mais de $n - 1$ perguntas do tipo indicado qual é a celebridade.

2. **Outra vez a celebridade**
   Esta é uma reformulação do exercício anterior em termos da teoria dos grafos. Um "poço" num grafo dirigido $G = (V, E)$ é um vértice u tal que

   $$[\forall x \in V,\ x \neq u \Rightarrow (x, u) \in E] \ \wedge \ [\forall x \in V,\ x \neq u \Rightarrow (u, x) \notin E]$$

   Seja $|V| = n$.

   – Mostre que qualquer grafo dirigido tem no máximo um poço.

   – Sabe-se que G tem um poço. Especifique um algoritmo de ordem $O(n)$ que determina esse poço efectuando "acessos" aos ramos do grafo apenas com questões do tipo "$(a, b) \in E$?"; o número de questões efectuadas será necessariamente de ordem $O(n)$.

3. **Maioria de votos**
   Um vector contém uma sequência de n votos em determinados candidatos especificados por inteiros. Existe uma maioria absoluta quando um candidato tem mais que $n/2$ (divisão nos reais) votos. Por exemplo, em

   $$v = [4, 4, 5, 5, 5, 4, 6, 5, 5, 6, 5]$$

   existe uma maioria, pois o candidato 5 tem 6 votos e $6 > 11/2$, mas na lista de votos $[4, 5, 4, 2, 5, 5]$ não existe maioria. Pretende-se determinar, dado o vector dos votos, se existe maioria ou não e, caso afirmativo, qual é o correspondente candidato.

   Um modo simples de resolver o problema é ordenar o vector e determinar o comprimento da maior lista de elementos iguais consecutivos. Mas este método é, na melhor das hipóteses, de ordem $\Theta(n \log n)$. Mas pretende-se um algoritmo mais eficiente, de ordem n. A seguinte observação pode ajudar.

($\star$) Se existe um candidato maioritário e $v[i] \neq v[j]$, então na lista que resulta de eliminar em v os elementos i e j, o candidato maioritário existe e é o mesmo. Por exemplo, como $v[2] \neq v[3]$ (índices a começar em 1) em

$$v = [4, 4, 5, 5, 5, 4, 6, 5, 5, 6, 5]$$

se há maioria, ela é a mesma que em $[4, 5, 5, 4, 6, 5, 5, 6, 5]$ (mas pode não haver maioria em v e haver no vector com os elementos retirados, dê um exemplo!).

(a) Demonstre a observação ($\star$).

(b) Usando a observação ($\star$) implemente um algoritmo que faz apenas uma passagem pelo vector (sendo portanto de ordem n) que, a partir de um vector de votos v dado, determine se existe ou não uma maioria e no caso afirmativo qual o correspondente candidato.

4. **Mudam-se os bits...**
Ao passar de n para $n+1$, o número de bits que mudam na representação binária de n pode ser grande. Por exemplo, na transição `0110111` $\rightarrow$ `0111000` há 4 bits que se modificam (os 4 últimos). Este facto pode ser inconveniente em certas aplicações. Pretende-se, para cada $p \geq 0$, uma sequência de todas as $n = 2^p$ palavras de p bits

$$s_0,\ s_1,\ \ldots,\ s_{n-1}$$

por forma a que

1) Para $0 \leq i < n$: s_i e s_{i+1} difiram apenas num bit.
2) s_0 e s_{n-1} difiram apenas num bit.

Por exemplo, para $n = 8$, a sequência

$$\texttt{000} \longrightarrow \texttt{010} \longrightarrow \texttt{011} \longrightarrow \texttt{111} \longrightarrow \texttt{110} \Longrightarrow \texttt{101} \longrightarrow \texttt{001} \Longrightarrow \texttt{100} \longrightarrow \texttt{000}$$

não tem a propriedade indicada, pois nas transições " $\Longrightarrow$ " modifica-se mais que 1 bit.

(a) Procure (por tentativas) uma solução para $p = 2$ e para $p = 3$.

(b) Caracterize sistematicamente para cada p uma sequência com as propriedades indicadas. **Sugestão.** Comece com o caso $p = 1$ ($n = 2$) e caracterize um método de se passar da sequência que corresponde a p (que tem 2^p palavras de p bits) para a sequência que corresponde a $p+1$ (que tem 2^{p+1} palavras de $p+1$ bits).

(c) Escreva uma função recursiva, baseada no método definido na alínea anterior, que receba p como parâmetro e retorne a lista (ou vector) de palavras correspondente

$$[s_0,\ s_1,\ \ldots,\ s_{2^p-1}]$$

(d) Escreva uma função que receba p e m como parâmetros, sendo $0 \leq m < 2^p$ e retorne a palavra s_m.