Problema “Sob pressão”

```
SOBPRESSAO()
    ler(n);
    ler_prefs(prefs,n);
    ler(p);
    Enquanto (p > 0) fazer
        ler_proposta(atrib);
        g ← construir_grafoPressoes(atrib, prefs, n);
        pressao ← caminho_maximo_DAG(g, n);
        Se (pressao = −1) então
            escrever(”Indeterminado (nao Pareto-optima)”);
        senão
            escrever(pressao);
        p ← p − 1;
```

Assumimos que $prefs$ é uma matriz com pelo menos n linhas e n colunas. Como as pessoas e as tarefas são identificadas por números consecutivos de 1 a n (cf., enunciado) e a biblioteca disponibilizada para suportar grafos assume também que os nós são numerados de 1 a n, pode ser útil $prefs$ ter pelo menos $n+1$ linhas. Assim, evitamos ter de efetuar correção de identificadores. A linha i ficará com as preferências da pessoa i. Do mesmo modo, $atrib$ será um vetor de inteiros, com $n+1$ posições, sendo $atrib[i]$ a tarefa que a pessoa i desempenha na proposta.

A função **construir_grafoPressoes(**$atrib$**,** $prefs$**,** n**)** usa $atrib$ e $prefs$ para construir o grafo de pressões (definido no enunciado). É dito no enunciado que esse grafo é um DAG se a atribuição corresponder a uma solução pareto-ótima. Caso contrário, não é um DAG. Notar que a tarefa $atrib[i]$ exerce pressão sobre todas as tarefas que estão antes de $atrib[i]$ na lista de preferências da pessoa i (isto é, na linha i da matriz $prefs$). Basta determinar j tal que $prefs[i][j] = atrib[i]$ e inserir arcos de $atrib[i]$ para $prefs[i][k]$, para $k < j$.

O valor que o programa vai ter de calcular é o comprimento do caminho máximo no DAG (se o grafo for um DAG).

A função **caminho_maximo_DAG(**g**,** n**)** deverá ser uma adaptação do algoritmo dado nas aulas para determinar o comprimento do caminho máximo num DAG. Como g pode não ser um DAG, é preciso verificar se há ciclos. Para isso basta *contar o número de nós que passam pela fila* (isto é, quantos tinham grau de entrada zero inicialmente ou ficaram com grau de entrada zero durante a pesquisa). Se o número não for n, então o grafo tem ciclos e a função deve retornar -1. Esta propriedade resulta do facto de um DAG ter sempre algum nó com grau de entrada zero. Quando se retiram arcos a um DAG obtém-se um DAG. No algoritmo apresentado, não se retiram os arcos explicitamente, mas decrementa-se o grau de entrada de cada vértice w adjacente a v (sendo v o nó que saiu da fila). Os nós que não passam na fila definem um subgrafo de g que tem ciclos. Se todos passam na fila, então o grafo não tem ciclos.

O algoritmo implementado para o problema "Sob Pressão" deve ter complexidade $O(n^2 + pn^2)$, sendo p o número de propostas. Notar que o número de arestas do grafo construído para cada proposta é $O(n^2)$, mas, espera-se que o algoritmo implementado para determinar o caminho máximo tenha complexidade **linear na dimensão do grafo** (a qual é dada pelo número de nós e de ramos).

Na implementação em C, é importante libertar o espaço alocado dinamicamente quando deixar de ser necessário (para evitar esgotar a *heap*). Por exemplo, quando se termina a análise de uma proposta, o espaço reservado para o grafo deve ser libertado antes de se criar um novo grafo.