MC558A - Projeto e Análise de Algoritmos II

Lehilton Pedrosa
Murilo de Lima (PED C)

Exercício de Programação V

- **Prazo de submissão:** 9 de junho às 23:59:59
- O exercício deve ser implementado em C ou C++
- Número máximo de submissões: 10
- Tempo máximo de execução: 10s

Matrizes Reorganizáveis

Uma matriz quadrada de zeros e uns é dita *reorganizável* se, após uma sequência de permutações de linhas ou colunas, a matriz resultante possui apenas números 1 na diagonal principal.

Por exemplo, considere a seguinte matriz.
0 1 1
0 0 1
1 0 0
Se trocarmos as linhas 1 e 2, obtemos a seguinte matriz.
0 0 1
0 1 1
1 0 0
Em seguida, se trocramos as colunas 1 e 3, obtemos a seguinte matriz.
1 0 0
1 1 0
0 0 1
Note que essa matriz possui apenas 1s na diagonal principal, portanto a matriz original é reorganizável.

Dada uma matriz quadrada de 0s e 1s, você deve escrever um programa que decida se ela é ou não reorganizável. Você deve **obrigatoriamente** modelar o problema como um problema de fluxo máximo. **Implementações que utilizem outras estratégias, mesmo dentro da complexidade solicitada, receberão nota zero no exercício**.

Entrada: na primeira linha da entrada é dado um número inteiro n. Você pode supor que $\mathbf{1 \leq n \leq 1000}$.
A seguir são dadas n linhas, cada linha contendo n números 0 ou 1 separados por espaço, representando a matriz de entrada.

Saída: uma única linha com a resposta "SIM" caso a matriz seja reorganizável, e "NAO" caso contrário.

Exemplos:
Entrada:
3
0 1 1
0 0 1
1 0 0
Saída:
SIM

Entrada:
3
0 0 1
0 0 1

1 1 0
Saída:
NAO

Dica: para testar se seu algoritmo de fluxo máximo está correto, observe o corte definido pelo conjunto de vértices alcançáveis no grafo residual.

Relatório: você deve incluir, no cabeçalho do arquivo do seu código, um comentário com 100 a 300 palavras, explicando sua solução. Não é necessário fazer uma prova formal, mas você deve argumentar por que sua solução funciona.
Sugestão: escreva o relatório antes de implementar o código; isso vai te ajudar a pensar melhor no problema. Após concluir a implementação, revise o relatório para checar se algo precisa ser modificado.

Observações:

- O SuSy utiliza o GCC 4.4.7 20120313 (Red Hat 4.4.7-17). São utilizadas as seguintes flags para compilação:
  - C99: `-std=c99 -pedantic -Wall -lm`
  - ANSI C: `-ansi -pedantic -Wall -lm`
  - C++: `-ansi -pedantic -Wall -lm`
- Você deve implementar estruturas de dados eficientes, com consumo de memória O(n + m), onde m é o número de 1s na matriz.
  Implementações com complexidade de memória fora do especificado receberão uma penalidade de 3 pontos.
- Seu algoritmo deve executar em tempo O(nm²). Lembre-se que alocação de memória (mesmo memória estática alocada na pilha) influi na complexidade de tempo.
- A nota do exercício depende do número de casos de teste que você acertar; são dados 10 casos de teste, sendo 7 abertos e 3 fechados, valendo o seguinte:
  - 1 caso correto: nota 0,5
  - 2 casos corretos: nota 1,0
  - 3 casos corretos: nota 1,5
  - 4 casos corretos: nota 2
  - 5 casos corretos: nota 3
  - 6 casos corretos: nota 4
  - 7 casos corretos: nota 5,5
  - 8 casos corretos: nota 7
  - 9 casos corretos: nota 8,5
  - 10 casos corretos: nota 10
- No entanto, **implementações com complexidade de tempo fora do especificado receberão nota zero no exercício**.
  Isto poderá ser verificado através de casos de teste fechados adicionais, com tamanho de entrada maior, executados fora do SuSy pelo monitor.
- Você pode utilizar as bibliotecas-padrão do C e as estruturas de dados da biblioteca-padrão do C++.
- **Trechos de código copiados da Internet ou dx coleguinha configuram plágio.**
- Sugerimos que você use scanf para fazer a leitura da entrada, a fim de garantir que seu código execute no tempo especificado.
- Seu código deve estar identado, modularizado e bem comentado. Identifique-se e deixe claro quais estruturas de dados e algoritmos foram utilizados.
