Trabalho Prático 18

Avenidas

Algoritmos e Estruturas de Dados I

Prazo: **03** de **dezembro** de 2017

1 Introdução

As estradas da cidade de Joãozinho e Zezinho estão severamente danificadas, devido ao intenso fluxo de veículos pesados criado pelo desenvolvimento econômico da cidade. Para resolver o problema, o prefeito decidiu construir novas avenidas. Após grande planejamento, ficou definido que:

- todas as avenidas construídas terão mão única, e ligarão exatamente dois bairros;
- nenhum par de avenidas se intersectará — serão construídos viadutos, túneis e pontes conforme necessário;
- deve ser possível, partindo de qualquer bairro, chegar a qualquer outro bairro usando só as novas avenidas, sempre respeitando a mão das avenidas.

Os engenheiros da cidade desenharam uma proposta de mapa viário e o prefeito verificou que o plano satisfaz as três primeiras restrições, mas não conseguiu verificar a última.

2 Tarefa

Lembrando da ajuda que deu quando resolveram dividir a cidade, o prefeito pediu que você escrevesse um programa que determina se o plano de avenidas permite viajar de qualquer bairro até qualquer outro da cidade.

3 Entrada

A primeira linha de cada caso de teste contém dois inteiros, N (N nunca será maior que 100) e A, indicando, respectivamente o número de bairros e avenidas. Cada uma das A linhas seguintes descrevem uma avenida: a linha contém dois inteiros D e P que indicam que existe uma avenida de mão única ligando o bairro D a outro bairro, P (os bairros são numerados de 1 a N).

4 Saída

Para cada par de bairros, você deverá imprimir o seguinte texto: `"D -> P: E"`, sem aspas, onde D é o bairro origem, P é o bairro destino e E é um caractere `S` caso exista um caminho de D para P ou `N` caso contrário. Você não deverá imprimir as linhas onde $D = P$. A saída deverá estar ordenada primeiro por D e depois por P. Na dúvida, siga o exemplo abaixo.

5 Exemplo

Entrada

```
3 4
1 2
2 3
1 3
3 2
```

Saída

```
1 -> 2: S
1 -> 3: S
2 -> 1: N
2 -> 3: S
3 -> 1: N
3 -> 2: S
```

Outros exemplos estão disponíveis na página deste trabalho no Moodle.

6 Avisos

Avisos mandatórios para o envio do trabalho:

- O código deve ser escrito em linguagem `C`
- As entradas que serão utilizadas para teste não conterão erros, então não será necessário testar a validade das mesmas
- Não utilize chamadas da função `system` (por exemplo, `system("pause")`) pois essas podem variar de acordo com o sistema operacional e os *softwares* instalados da máquina onde o programa está rodando
- Deixe seu código bem comentado para facilitar a correção

- Não utilize espaços ou caracteres especiais nos nomes dos arquivos. Utilize apenas os caracteres de `A` a `Z` (tanto maiúsculas como minúsculas) sem acento, os números de `0` a `9` e os caracteres `-` (hífen), `_` (*underscore*) e `.` (ponto final)
- Utilize a extensão `.c` para arquivos de código e `.h` para arquivos de cabeçalho, quando aplicar
- Se for submeter os arquivos via upload, não envie um arquivo comprimido (por exemplo, `.zip`, `.rar`, etc.). Utilize os diversos campos da aba "Submissão", um para cada arquivo
- Envie apenas os arquivos `.c` e `.h`
- Não copie o trabalho de algum colega ou baixe da internet. Lembre-se que o prejudicado será você pois o aprendizado obtido nessa disciplina será utilizado durante diversas outras etapas do seu curso

7 Dicas

Dicas importantes para o desenvolvimento deste trabalho:

- Utilize a função `scanf` para ler as entradas do usuário
- Utilize a função `printf` para imprimir os resultados das operações
- Muito cuidado com a ordenação da saída!
- Modele as conexões entre as bairros como uma matriz
- Recursividade pode ser sua amiga na resolução deste trabalho! Lembre-se apenas de salvar quais bairros já foram visitadas.
- Qualquer dúvida que tiver, utilize o fórum de dúvidas no Moodle. Inicie o assunto do tópico com a tag `[TP18]`
- Caso prefira, participe das monitorias toda quarta das 17h às 18h na sala 2012

8 Checklist

Checklist não exaustiva de passos até a entrega do trabalho:

1. Pesquise o funcionamento das funções citadas acima para facilitar o uso
2. Implemente e compile o programa
3. Teste para o exemplo dado acima. Compare as saídas para garantir o funcionamento correto

4. Faça o mesmo do item anterior para os exemplos disponibilizados na página deste trabalho no Moodle

5. Envie o trabalho pelo Moodle, onde ele será testado automaticamente para todos os casos disponíveis

6. Caso algum teste dê errado, volte ao passo 2

Bom trabalho e divirta-se!