MC202 - Estruturas de Dados

Lab 04 - Alocação e Passagem por Referência

Data da Primeira Chance: 04 de setembro de 2023

Link da atividade: https://classroom.github.com/a/x4_Kq2Dp

Peso: 2

A Companhia de Saneamento Ambiental do Distrito Federal (CAESB) tem enfrentado desafios significativos ao lidar com problemas de abastecimento de água em várias quadras da região. Especificamente, essas quadras são áreas urbanas densamente povoadas, onde se localizam conjuntos de prédios residenciais[1]. O abastecimento de água nessas áreas tornou-se uma preocupação crítica, já que os problemas de distribuição e pressão afetam diretamente a qualidade de vida dos moradores e a eficiência do serviço prestado.

Para solucionar essa questão complexa e garantir uma resposta mais eficaz aos problemas de abastecimento, a CAESB tomou uma iniciativa proativa. A empresa lançou um edital público e está buscando uma solução inovadora baseada em *software*, por meio de um processo de licitação. O objetivo principal é desenvolver um sistema especializado que dado um mapeamento detalhado de cada quadra afetada, com notas de urgência atribuídas que variam de -100 a 100 para cada conjunto de prédios residenciais, encontre uma área crítica a ser reparada.

Essas notas serão atribuídas conforme a prioridade de cada conjunto, levando em consideração diversos fatores, como a gravidade das falhas de abastecimento, a frequência dos problemas relatados pela comunidade local, a capacidade de suportar a falta de água, entre outros indicadores relevantes. Os conjuntos que receberem as notas mais elevadas representarão aqueles que necessitam de atendimento imediato e uma intervenção urgente por parte das equipes de reparo e manutenção.

---

[1] Caso deseje saber mais como funciona o endereçamento de Brasília recomendamos a seguinte leitura: https://www.silveiraimoveis.com/como-funcionam-os-enderecos-de-brasilia

Figura 1: Mapa Asa Norte/DF

Fonte: Secretaria de Estado de Desenvolvimento Urbano e Habitação - DF

O sistema em questão tem como foco encontrar uma região contígua para reparar. Consideramos que a quadra é um quadrado, representado por uma matriz (quadrada) com as notas dos conjuntos que a compõe, e que a região a ser reparada também será um (único) quadrado com os subconjuntos prioritários. A região escolhida para os reparos deve ser aquela com a maior soma das notas (em caso de empate fica a critério da equipe de reparação escolher, uma alternativa seria priorizar a primeira ocorrência encontrada).

Como a CAESB deseja realizar algumas simulações, é possível que o usuário queira mudar um único valor da matriz referente à região escolhida e refazer a análise. Isso pode ser feito várias vezes, **sempre em relação à matriz da região escolhida (não a matriz original)**.

Entrada

A matriz a ser considerada tem a variável 'lc', onde 'lc' representa o número de linhas e colunas da matriz quadrada. Cada elemento '$n_{i,\ j}$' da matriz corresponde a um valor inteiro que varia de -100 a 100.

Além disso, o sistema possui a opção '*alt*', uma variável inteira que assume o valor `1` caso o usuário deseje alterar algum elemento da matriz, ou `0` caso contrário. Caso o usuário opte por realizar a alteração, ele deverá fornecer o índice da linha '*i*' e o índice da coluna '*j*' do elemento que deseja alterar, bem como o novo valor '*val*' a ser atribuído a esse elemento. É importante notar que os índices '*i*' e '*j*' devem estar nos limites válidos da matriz (`0` a *lc*−`1` para linhas e colunas). Em seguida deve-se gerar uma nova submatriz de tamanho '*lc-1*'. Essas alterações podem ser realizadas quantas vezes for necessário.

Dessa forma, com base na matriz original e suas eventuais alterações, o sistema será capaz de encontrar a submatriz com a maior soma, possibilitando a alocação otimizada das equipes de reparos nos conjuntos mais necessitados de forma inteligente.

[lc]
[$n_{0,0}$ $n_{0,1}$ $n_{0,2}$ … $n_{l-1,c-1}$]
[alt] [2]
[i] [j] [$val_{i,j}$] [3]
[alt] [4]

Saída

A saída é formada pela impressão da matriz original e submatriz com a maior soma com espaçamento tabular ("`\t`") entre os valores. Reforça-se que podem existir quantas alterações forem necessárias até sobrar somente um conjunto (um ponto na matriz) e o resultado dessas eventuais alterações devem ser recalculados. Quando *'lc==1'* deve-se encerrar o programa, dessa forma, não é necessário perguntar para o usuário se ele deseja alterar algum elemento da matriz.

[2] Se a matriz fornecida ser de tamanho `2x2`, então a próxima matriz será de tamanho `1x1`. Nesse caso não é necessário pedir '`alt`'.
[3] Essas informações são pedidas caso '`alt`' seja pedido e igual a `1`.
[4] Se a última alteração resultar em uma matriz `1x1` não é necessário pedir '`alt`'.

Quadra:

$s_{0,0}$ $s_{0,1}$ ... $s_{0,c-1}$

$s_{1,0}$ $s_{1,1}$ ... $s_{1,c-1}$

$s_{2,0}$ $s_{2,1}$ ... $s_{2,c-1}$

Conjuntos que precisam de atenção:

$s_{0,0}$ $s_{0,1}$

$s_{1,0}$ $s_{1,1}$

$s_{2,0}$ $s_{2,1}$

Exemplos

Exemplo 1:

Entrada

```
3
-6 0 9 -9 5 -1 0 3 8
1
0 1 9
```

Saída

```
Quadra:
-6    0     9
-9    5     -1
0     3     8

Conjuntos que precisam de atenção:
5     -1
3     8

Conjuntos que precisam de atenção:
```

```
9
```

Exemplo 2:

Entrada

```
5
1 2 -1 -4 -20 -8 -3 4 2 1 3 8 10 1 3 -4 -1 1 7 16 94 -19 5 -100 100
1
3 3 -100
1
0 0 100
0
```

Saída

```
Quadra:
1     2     -1    -4    -20
-8    -3    4     2     1
3     8     10    1     3
-4    -1    1     7     16
94    -19   5     -100  100

Conjuntos que precisam de atenção:
-3    4     2     1
8     10    1     3
-1    1     7     16
-19   5     -100  100

Conjuntos que precisam de atenção:
4     2     1
10    1     3
1     7     16
```

```
Conjuntos que precisam de atenção:
100   2
10    1
```

Regras e Avaliação

Neste laboratório, é obrigado a usar **alocação dinâmica para as matrizes e passagem por referência, deve-se usar a passagem por referência na função que devolve a matriz.** Seu código será avaliado não apenas pelos testes do *GitHub*, mas também pela qualidade. Dentre os critérios subjetivos de qualidade de código iremos analisar:

- O uso apropriado de funções, de comentários;
- A escolha de bons nomes de funções e variáveis;
- A ausência de trechos de código repetidos desnecessariamente;
- O uso apropriado de funções;
- A eficiência dos algoritmos propostos;
- A correta utilização das Estruturas de Dados;
- O tempo de execução e uso de memória dos algoritmos projetados;
- **Vazamento e outros erros de memória (*valgrind*);**

Note, porém, que essa não é uma lista exaustiva, pois outros critérios podem ser analisados dependendo do código apresentado visando mostrar ao aluno como o código poderia ser melhor.

Submissão

Você deverá criar o arquivo `mapeamento.c` e submeter no repositório criado no aceite da tarefa. Você pode enviar arquivos adicionais caso deseje para serem incluídos por `mapeamento.c`.

Lembre-se que sua atividade será corrigida automaticamente na aba "*Actions*" do repositório. Confirme a correção e o resultado, já que o que vale é o que está lá e não na sua máquina.

Após a correção da primeira entrega, será aberta uma segunda chance, com prazo de entrega apropriado.