MC202E - ESTRUTURAS DE DADOS

Lab11: A Torre dos Enganos

A Torre dos Enganos é uma torre com mais de 8 mil metros de altura, sem janela ou qualquer coisa do tipo. No topo da torre existe uma entrada que leva a uma sala (que está no último andar). Nesta sala existem dois elevadores e apenas um computador, com a seguinte mensagem.

> Cada andar desta torre dá acesso a dois elevadores, denotados por A e B, onde
> - quando o elevador A é acionado no andar i, ele desce exatamente X_i andares de uma única vez e sobe exatamente Y_i andares de uma única vez;
> - quando o elevador B é acionado no andar i, ele desce exatamente W_i andares de uma única vez e sobe exatamente Z_i andares de uma única vez;
>
> Existe apenas uma única saída nesta torre e está situada no térreo (andar 0).
>
> Entretanto, quando algum elevador é utilizado, a saída começa a ser lacrada automaticamente.
>
> A única forma de chegar até o térreo antes da saída ser totalmente lacrada é se você minimizar o número de vezes que os elevadores são acionados.
>
> Sempre é possível "chamar" o elevador não utilizado até seu andar atual.

Considere que você acabou de acordar no topo da Torre dos Enganos. Como você sabe programar muito bem, use o computador da sala para determinar como chegar no térreo antes da saída ser totalmente lacrada.

Tarefa

Escreva um programa em C que recebe como entrada o número total de andares existentes e os valores de X_i, Y_i, W_i e Z_i de cada andar, e que imprima como saída uma sequência que indica qual elevador entrar e qual botão pressionar para se chegar até o térreo. Para isso, você deverá modelar a entrada como um **grafo**, cuja representação deverá utilizar **lista de adjacência**.

Entrada

A primeira linha da entrada contém um inteiro $\boldsymbol{m}$ que indica o número de cenários. Para cada cenário, será dado um inteiro $\boldsymbol{n}$ que corresponde ao número de andares, seguido de n linhas, onde cada linha contém **exatamente** quatro inteiros não negativos que correspondem os valores de X_i, Y_i, W_i e Z_i, para $i = n, n\text{-}1, \ldots, 1$.

Saída

A saída do seu programa deverá conter uma resposta por cenário. Para cada cenário, deverá ter uma mensagem com a identificação do cenário (i.e., `"Cenário #c"`, onde `c` é o identificador do cenário), começando a partir de 1; seguido de várias linhas, onde cada linha deverá conter o caractere `'A'` ou `'B'` para indicar qual elevador entrar, seguido do caractere `'X'`, `'Y'`, `'W'`, ou `'Z'` para indicar qual botão pressionar.

Observação

- Como pode existir mais uma sequência que minimiza o número de vezes que os elevadores são acionados, então o seu programa deverá seguir a seguinte ordem de preferência: escolher entrar primeiro no elevador A em vez do elevador B; e escolher descer os andares em vez de subir. Caso contrário, a solução dada poderá ser considerada **incorreta**.

Exemplo

A grafia da saída abaixo deve ser seguida rigorosamente por seu programa, inclusive a impressão de uma linha em branco no final da saída.

Entrada

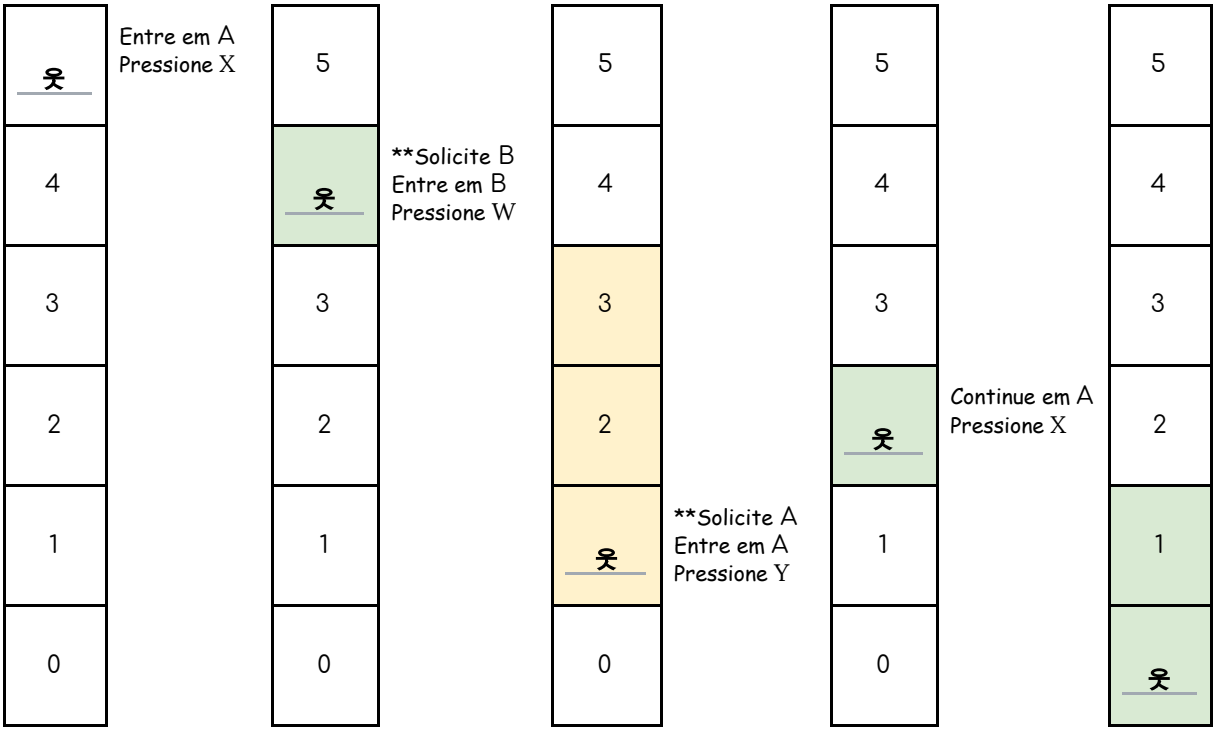
```
1
5
1 0 2 0
0 0 3 0
0 2 0 1
2 0 0 0
0 1 0 0
```

Saída

```
Cenário #1
A X
B W
A Y
A X

```

Para esclarecer melhor como a saída acima funciona, considere a ilustração abaixo que representa um esquema simplificado dos passos a serem realizadas. Nesta ilustração, a torre é representada como um vetor coluna, onde o caractere 웃 indica o andar da torre em que você se encontra. Além disso, as cores verde e amarelo servem para indicar quantos andares foram deslocados.



Obs.: Nesta solução os elevadores foram **acionados 6 vezes**.

Critérios específicos

Os seguintes critérios específicos sobre o envio, implementação, compilação e execução devem ser satisfeitos.

i. Submeter no *SuSy* os arquivos:

Obrigatórios

⇒ `lab11.c`: Deverá conter a função principal para a solução do problema.

⇒ `grafo.*`: Arquivos de cabeçalho e fonte devem conter **somente** a interface e implementação de estrutura e algoritmos de grafo. Em outras palavras, **não deve conter funções do problema**.
Espera-se que sejam implementadas as seguintes funções:
- ❖ Criar grafo (recebe o número de vértices);
- ❖ Inserir aresta (recebe dois vértices que são extremos de uma aresta);
- ❖ Obter adjacência (recebe um vértice).

Obs.: O grafo deve ser representado por meio de **lista de adjacência**.

⇒ `lista.*`: Arquivos de cabeçalho e fonte devem conter **somente** a interface e implementação de estrutura de **lista ligada**.
Espera-se que sejam implementadas as seguintes funções:
- ❖ Criar lista;
- ❖ Inserir (recebe um elemento) ⇨ Complexidade O(1);
- ❖ Obter iterador ⇨ Complexidade O(1);
- ❖ Avançar posição (recebe um iterador de lista) ⇨ Complexidade O(1);

- ❖ Eh vazia (ou similar).

Obs.: A estrutura de lista deverá ser utilizada na estrutura de grafo.

Opcionais

⇒ `*.*`: Enviar até 2 arquivos cabeçalho e 2 arquivos fonte, desde de que contribuam para a modularização da solução.

ii. **É obrigatório** implementar uma solução que utiliza **conceitos e algoritmos de grafo**.
Obs.: Faz parte da avaliação desta tarefa a **sua interpretação** do problema. Em outras palavras, a forma como a entrada é modelada e a forma como a resposta é obtida a partir dos algoritmos em grafos utilizados.

iii. Flags de compilação:
`-std=c99 -Wall -Werror -lm`

iv. Tempo máximo de execução: **1 segundo**.

Observações gerais

As tarefas possuirão os mesmos casos de testes abertos e fechados, no entanto o número de submissões permitidas e prazos são diferentes. As seguintes tarefas estão disponíveis no *SuSy*:

- ❏ **Lab11-AmbienteDeTeste**: Esta tarefa serve para testar seu programa no *SuSy* antes de submeter a versão final. Nessa tarefa, tanto o prazo quanto o número de submissões são ilimitados, porém os arquivos submetidos aqui **não serão corrigidos**.
- ❏ **Lab11-Entrega**: Esta tarefa tem limite de uma única submissão e serve para entregar a **versão final** dentro do prazo estabelecido para o laboratório. Não use essa tarefa para testar o seu programa e submeta aqui apenas quando não for mais fazer alterações no seu programa.

Avaliação

Este laboratório será avaliado conforme o número de **casos de teste fechados** em que o seu programa apresentou saída correta, menos possíveis descontos referentes aos critérios de correção e de qualidade de código, os quais estão disponíveis na **planilha de notas**. Entretanto, outros critérios podem ser incorporados na avaliação desta tarefa se for julgado pertinente; e **a nota pode ser zerada caso não atender os critérios específicos.**