Instituto de Computação da UNICAMP

MC558: Projeto e análise de algoritmos II

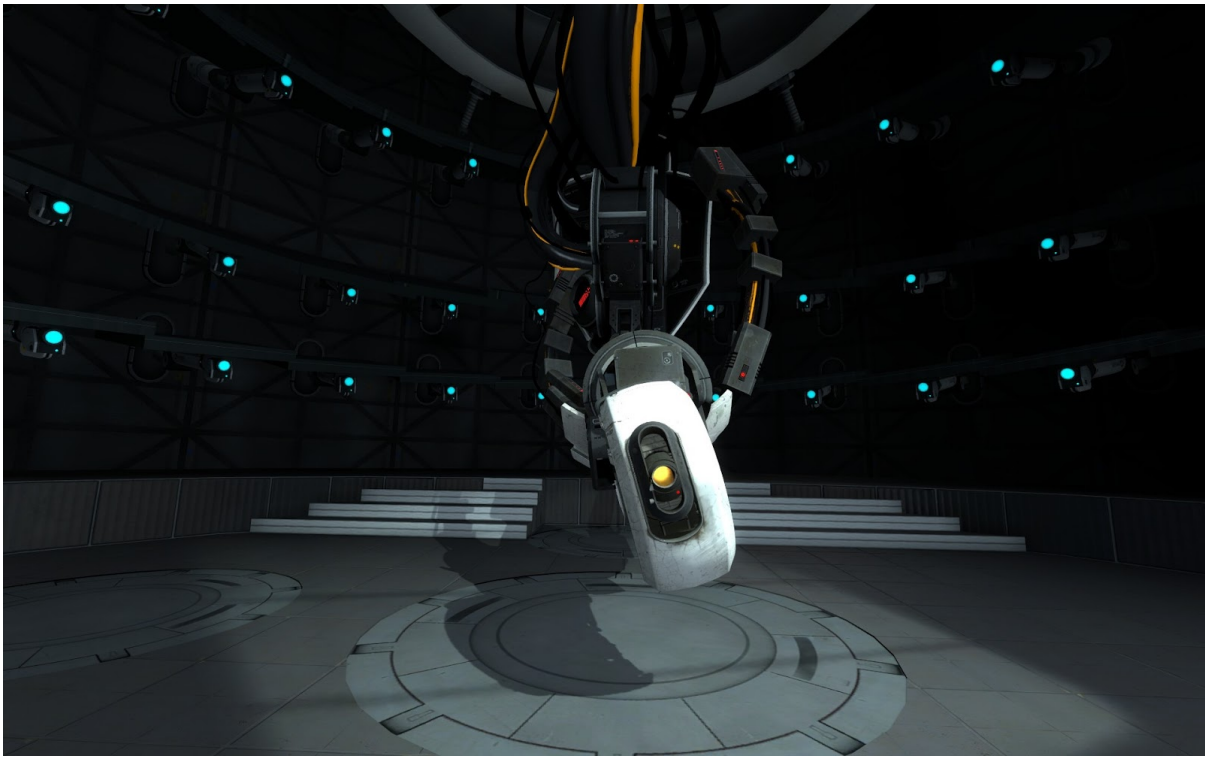
Segundo Semestre de 2019 - Turma A

Laboratório Nº 02 - Puzzle

Prazo de entrega: **19/11/2019 às 23:59:59**

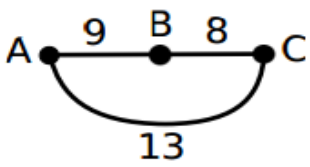
Docente: C. N. Campos
Monitor: Léo Vieira Peres

Contexto:



"The Enrichment Center regrets to inform you that this next test is impossible."

Você, como o mais novo cientista maluco da Aperture Laboratories, foi encarregado por GLaDOS de medir o nível de dificuldade dos puzzles do recém-inaugurado Aperture Science Computer Aided Enrichment Center. O objetivo do Enrichment Center é prover um ambiente seguro e lúdico no qual cobaias possam testar a invenção revolucionária do laboratório: o Hand Held Portal Device (HHPD). A área de teste pode ser descrita como um grafo conexo. Para passar de uma área x para uma área y, e vice-versa, uma cobaia deve resolver um puzzle de dificuldade $w(x,y)$ com a ajuda do HHPD. A figura a seguir ilustra uma área de testes com três áreas e três puzzles. De A, uma cobaia pode resolver um puzzle de dificuldade 9 e ir para B ou resolver um puzzle de dificuldade 13 e ir para C.



O objetivo das cobaias é sair de sua área inicial *s* e chegar a outra área *t*, onde serão recompensadas pela sua participação com bolo e aconselhamento psiquiátrico. Como níveis de dificuldades maiores implicam em maior risco de neurointoxicação, as cobaias tentarão percorrer o grafo de forma a evitar puzzles mais difíceis, isto é, dentre todos os caminhos de *s* a *t*, escolherão aquele que minimiza o puzzle de dificuldade máxima no caminho. Sua tarefa é determinar a dificuldade deste puzzle.

No exemplo acima, uma cobaia em A pode chegar em C por dois caminhos distintos: A->C e A->B->C. O custo de uma aresta máxima no caminho A->C é 13. No caminho A->B->C, o custo máximo é max{9, 8} = 9. Logo, a menor dificuldade máxima que uma cobaia enfrentará será min{13, 9} = 9, pois ela evitará o caminho A->C.

Entrada:

A primeira linha da entrada contém dois inteiros *n* e *m*, indicando, respectivamente, o número de áreas e o número de ligações entre estas áreas. Em seguida, haverá *m* linhas com três inteiros *x*, *y* e *w(x,y)*, indicando que uma cobaia pode transitar entre as áreas *x* e *y*, se resolver um puzzle de dificultade *w(x,y)*. As áreas estão numeradas de 0 a *n*-1. A próxima linha contém um inteiro *k*. As próximas *k* linhas contêm um par de áreas *s*, *t*, para os quais a cobaia deve determinar a menor dificuldade máxima de um puzzle em um caminho de *s* a *t*. A entrada deve ser lida da entrada padrão.

Saída:

Para cada um dos *k* pares de vértices *s* e *t*, imprima uma linha com a menor dificuldade máxima que uma cobaia enfrentará para ir de *s* a *t*. A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq n \leq 500$
- $1 \leq m \leq (n*(n-1))/2$
- $1 \leq w(x,y) \leq 1000000$
- $1 \leq k \leq 50$

Exemplo de execução1:

```
4 4
0 1 6
0 2 1
1 3 2
2 3 2
2
0 1
0 3
2
2
```

Exemplo de execução2:

```
4 4
0 1 1
1 2 7
1 3 3
2 3 4
4
0 2
```

```
0 3
2 1
3 1
4
3
4
3
```

Notas: Textos em `azul` designam dados de entrada, isto é, que devem ser lidos pelo seu programa.
Textos em `preto` designam dados de saída, ou seja, que devem ser impressos pelo seu programa.

Observações gerais:

- O número máximo de submissões é 30.
- O seu programa deve estar completamente contido em um único arquivo denominado `puzzle.c` ou `puzzle.cpp`.
- Para a realização dos testes automáticos, a compilação se dará de uma das seguintes formas:

```
gcc -Werror -pedantic -Wall -lm -o puzzle puzzle.c
g++ -Werror -pedantic -Wall -lm -o puzzle puzzle.cpp
```

- Não se esqueça de incluir no início do programa o seu nome e o seu RA.

Critérios importantes:

Não deixe de ler esta seção porque poderá incluir informações extras sobre penalidades.