Trabalho Prático 1

Como será a entrega?

Existem 5 tarefas no Moodle, uma para cada um dos problemas. Os alunos deverão enviar **UM APENAS UM ARQUIVO .jl** em cada tarefa. A submissão de qualquer outro formato de arquivo ou de mais de um arquivo implicará em **ZERO**. O arquivo também deve ter uma nomenclatura especifica. Para cada tarefa, o arquivo submetido pelo aluno de matricula xxxxxx deve se chamar `tp1_xxxxxx.jl`.

Programas com o nome errado levarão nota 0!

O que faremos?

Neste trabalho prático os alunos são convidados a implementarem os modelos que eles criaram na primeira prova da disciplina. Para isso nós usaremos a linguagem de programação Julia e o solver Gurobi disponível através do pacote JuMP da linguagem.

Os arquivos enviados pelos alunos para cada problema serão testados para 3 instâncias de testes (que são semelhantes as instâncias de testes que foram disponibilidades aos alunos, mas não são iguais). Os arquivos serão testados com a seguinte linha de comando no prompt de comando:

```
julia tp1_xxxxxx.jl <instancia>
```

Onde `instancia` é o caminho para o arquivo que será testado. O programa deve então executar e imprimir na **TELA**:

```
TP1 xxxxxx = <valor>
```

Onde xxxxxx é o número de matrícula do aluno e `<valor>` é o valor da solução ótima obtida pelo método.

Cada execução será limitada a 5 minutos, tome os devidos cuidados para que seu código execute dentro do tempo limite.

Os alunos terão acesso a 3 instâncias de testes para realizar validações em seus códigos. Os arquivos estão disponíveis no Moodle e as soluções ótimas, assim como a formatação e interpretação das instâncias de testes estão indicadas abaixo:

Quais são os problemas?

Empacotamento

Considere um conjunto de objetos $O = \{o_1, o_2, \ldots, o_n\}$ cada objeto com um peso w_i. Dispomos de um várias caixas de papel, cada uma delas com o limite de peso 20Kg. Desejamos empacotar nossos objetos, utilizando o menor número de caixas possíveis, dado que em nenhuma caixa o valor da soma dos pesos dos objetos ultrapasse seu limite de peso.

0.0.1 Formatação da Entrada

A primeira linha do arquivo é no formato

`n <num>`

onde `<num>` é um inteiro representando o número de objetos. As demais linhas são no formato

`o <id> <peso>`

onde `<id>` é um inteiro para representar o objeto e `<peso>` é um ponto flutuante que representa o peso do objeto em Kg.

Observe que o separador é uma tabulação e não um espaço.

Clique Máxima

Dado um grafo $G = (V, E)$, uma *clique* é um conjunto de vértices dois a dois adjacentes, ou seja, com arestas entre eles. Desejamos determinar um subgrafo induzido que seja uma clique de cardinalidade máxima (maior tamanho em número de vértices).

0.0.2 Formatação da Entrada

A primeira linha do arquivo é no formato

`n <num>`

onde `<num>` é um inteiro representando o número de vértices no grafo. Assumimos que os vértices do grafo estão nomeados de 1 até `<num>`. As demais linhas são no formato

`e <v> <u>`

onde `<v>` e `<u>` são números inteiros representando vértices e essa linha no arquivo significa que temos uma aresta entre os vértices v e u.

Observe que o separador é uma tabulação e não um espaço.

Lotsizing com Backlog

Estamos auxiliando um produtor a planejar sua produção. Essa produtor quer que planejemos sua produção para um horizonte de tempo com n períodos. O produtor produz um único produto, conhece as demandas dos clientes para cada período de tempo i (d_i), o custo de produzir uma unidade do produto no tempo $i(c_i)$ e o custo de armazenar uma unidade do tempo i para o tempo $i + 1(h_i)$. Entretanto, devido a sazonalidade de seu produto, pode ser que os pedidos dos clientes não sejam satisfeitos em um período, esse caso, podemos entregar o produto atrasado para o cliente, mas pagamos uma multa de p_i por unidade de produto pedida pelo cliente e ainda não entregue no período i.

0.0.3 Formatação da Entrada

A primeira linha do arquivo é no formato

`n <num>`

onde `<num>` é um inteiro que representa o tamanho do horizonte de planejamento. As demais linhas são no formato

`<id> <num> <valor>`

onde `<valor>` é um ponto flutuante, `num` é um inteiro e `<id>` pode assumir 4 valores:

Se `<id>` **= c** : `valor` é o custo de produção no período `<num>`

Se `<id>` **= d** : `valor` é a demanda pelo produto no período `<num>`

Se `<id>` **= s** : `valor` é o custo de estocagem no período `<num>`

Se `<id>` **= p** : `valor` é o valor da multa no período `<num>`

Observe que o separador é uma tabulação e não um espaço.

Coloração de Grafos

Dado um grafo $G = (V, E)$, uma coloração própria é uma atribuição de cores aos vértices do grafo de tal forma que vértices adjacentes recebem cores diferentes. Desejamos determinar o menor número de cores necessárias para colorir de maneira própria um grafo dado de entrada.

A primeira linha do arquivo é no formato

`n <num>`

onde `<num>` é um inteiro representando o número de vértices no grafo. Assumimos que os vértices do grafo estão nomeados de 1 até `<num>`. As demais linhas são no formato

`e <v> <u>`

onde `<v>` e `<u>` são números inteiros representando vértices e essa linha no arquivo significa que temos uma aresta entre os vértices v e u.

Observe que o separador é uma tabulação e não um espaço.

A-Coloração de Grafos

Dado um grafo $G = (V, E)$, uma coloração própria é uma atribuição de cores aos vértices do grafo de tal forma que vértices adjacentes recebem cores diferentes. Se além disso, nos temos a propriedade que para cada conjunto de vértices coloridos com a cor i temos na vizinhança combinada desses vértices todas as demais cores, ou seja, se $C_i \subseteq V(G)$ é o conjunto dos vértices que receberam a cor i, temos que para toda cor i, $N(C_i) \cap C_j \neq \emptyset$ para toda cor j, com $j \neq i$. Diremos que a coloração é uma A-coloração.

Desejamos determinar o maior número de cores possíveis para se colorir de maneira própria um grafo dado de entrada e garantir que a coloração obtida é uma A-coloração.

A primeira linha do arquivo é no formato

`n <num>`

onde `<num>` é um inteiro representando o número de vértices no grafo. Assumimos que os vértices do grafo estão nomeados de 1 até `<num>`. As demais linhas são no formato

`e <v> <u>`

onde `<v>` e `<u>` são números inteiros representando vértices e essa linha no arquivo significa que temos uma aresta entre os vértices v e u.

Observe que o separador é uma tabulação e não um espaço.

Como será a avaliação?

Os códigos serão executados em um sistema com as seguintes configurações:

- Ubuntu 24.04.2 LTS.
- Processador Intel Core i7-12700.
- Julia versão 1.10.5.
- JuMP versão 1.23.1.
- Gurobi (do JuMP) versão 1.3.0
- Gurobi versão 11

Seu código deve executar nessa configuração e via terminal, caso ele não execute ou produza uma erro será atribuída a nota 0.

Cada problema valerá ao todo 2 pontos e cada execução de uma instância valerá $\frac{2}{3}$ ponto. Essa pontuação será atribuída da seguinte forma, considerando uma execução para uma instância:

Programa não finalizou a execução: 0.

Erro de execução: 0.

Impressão errada: 0.

Execução correta, mas valor errado:

Valor maior que o ótimo em problema de maximização: 0.

Valor maior que o ótimo em problema de minimização: 0.2.

Valor menor que o ótimo em problema de maximização: 0.2.

Valor menor que o ótimo em problema de minimização: 0.

Execução correta e valor certo: $\frac{2}{3}$.