Trabalho Prático 2 - Escapando da floresta da neblina

Descrição do problema

O herói lendário Linque está perdido na floresta da neblina e precisa de sua ajuda para escapar. A floresta está envolta por uma neblina mágica e quanto mais o tempo passa mais densa a neblina fica. Linque sabe que em pouco tempo a magia da neblina fará com que ele fique perdido ali para sempre, mas ele já está muito cansado das batalhas que travou. Sua única chance de sobrevivência é escapar da floresta antes que a exaustão tome conta dele.

A floresta é composta por n clareiras e m trilhas que as conectam. As clareiras numeradas de 0 até $n-1$. As trilhas podem ser percorridas apenas em uma direção. Algumas clareiras possuem portais, que as conectam por um caminho mágico de distância 0. Sempre que Linque caminha por uma das trilhas, ele consome energia proporcional à distância percorrida, o que significa que atravessar os portais não consome sua energia. No entanto Linque já está cansado da viagem, possui apenas s de energia e pode utilizar os portais no máximo k vezes. Sabemos que ele começa sempre na clareira 0 e a saída está na clareira $n-1$. Seu trabalho é decidir se Linque consegue escapar da floresta, dado que para isso ele deve ser capaz de chegar na clareira $n-1$ gastando no máximo s de energia e atravessando no máximo k portais. Caso contrário ele ficará perdido na floresta para sempre...

Algoritmos

Neste trabalho utilizaremos dois algoritmos. O primeiro é o algoritmo de Djikstra, um algoritmo clássico da computação para encontrar caminhos mínimos em grafos que não possuam arestas de peso negativo. Os detalhes do algoritmo e sua implementação estão bem descritos no vídeo abaixo:

https://www.youtube.com/watch?v=Kue4UUxstoU&ab_channel=MaratonaUFMG

O segundo algoritmo a ser implementado é o A* (A-estrela). É importante que antes de tentar entender este algoritmo você compreenda bem o Djikstra, uma vez que de certa forma eles são bem semelhantes.

https://pt.wikipedia.org/wiki/Algoritmo_A*

Na implementação do A* você deve utilizar a distância euclidiana para o vértice de destino como heurística.
Observação importante: Para a entrega da VPL é essencial que na sua implementação de ambos os algoritmos que ele pare quando o vértice de destino for retirado da fila de prioridades.

O que você deve implementar

Note que a implementação padrão não resolve nosso problema. Ambos algoritmos são implementados utilizando uma fila de prioridades para escolher qual o próximo vértice a ser analisado. Além dos algoritmos citados na seção anterior, você deve implementar um

TAD de fila de prioridades modificado para satisfazer as condições adicionais do problema. A escolha de estratégia de modificação do TAD faz parte do trabalho e deverá ser descrita na documentação.

Você deve também implementar seu TAD de grafo utilizando Matriz e Lista de adjacência. (Para a entrega da VPL, utilize Lista de adjacência, mas você precisará de ambas implementações para a análise experimental).

Entrada e saída

O formato apresentado nesta seção é o que será utilizado para a avaliação de corretude em uma VPL no moodle. Sinta-se livre da alterá-la para realizar seus experimentos para a documentação caso julgue necessário.

As instancias serão lidas pela pela entrada padrão. A primeira linha consiste em três inteiros n, m e k que representam respectivamente quantos vértices, trilhas e portais a floresta possui. As próximas n linhas possuem dois racionais x e y, descrevendo as coordenadas de cada clareira em um plano. Depois, as próximas m linhas possuem dois inteiros u e v, indicando que existe uma trilha entre as duas clareiras, que só pode ser percorrida começando em u e terminando em v. Por fim as próximas k linhas possuem também dois inteiros u e v, indicando quais clareiras são conectadas por portais (os portais também só podem ser atravessados começando em u e terminando em v). A última linha contém um racional s indicando a quantidade de energia que Linque possui para caminhar e um inteiro q indicando quantos portais podem ser utilizados.

A saída serão dois inteiros. O primeiro, utilizando o algoritmo Djikstra imprima 1 caso exista caminho de custo no máximo s e que cruze no máximo q portais e 0 caso contrário. De forma análoga faça o mesmo para o algoritmo A*.

Análise experimental

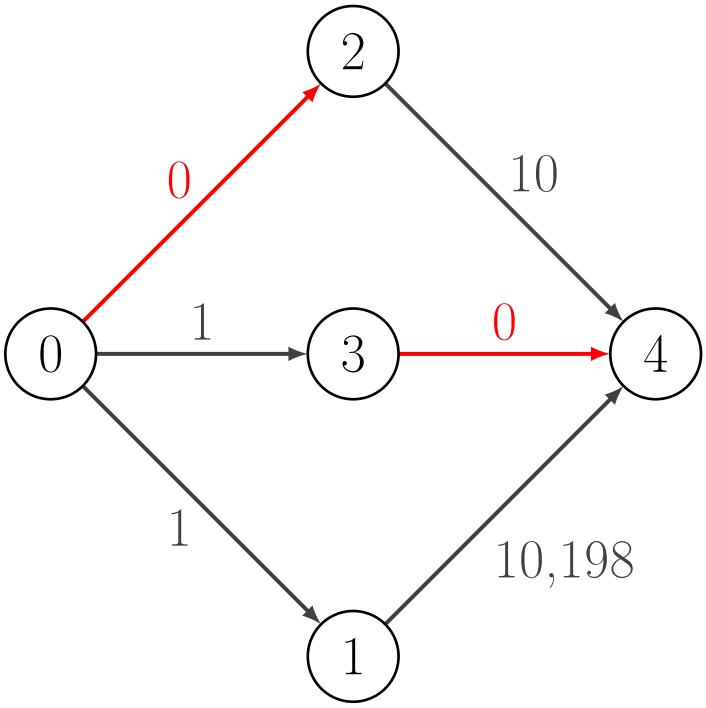
São esperadas as seguintes análises em sua documentação:

- Análises complexidade experimentais, para verificar os resultados teóricos apresentados na documentação.
- Comparação das implementações de Matriz e Lista de adjacência. Como elas impactaram a execução? Existem tipos de instancias que favorecem a utilização de alguma das implementações?
- Análises com relação a **qualidade** das soluções obtidas. Os algoritmos sempre encontram as mesmas soluções? Se você variar ou restringir alguns dos parâmetros de entrada essa resposta muda? Porque?

Exemplo

```
5  4  2
0  1
0  0
0  2
1  1
10  2
0  1
0  3
1  4
2  4
0  2
3  4
1  1
```

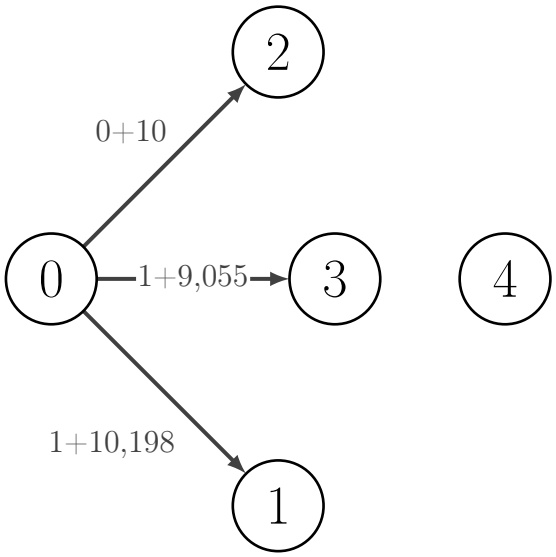
Podemos representar a entrada acima com o seguinte grafo direcionado:



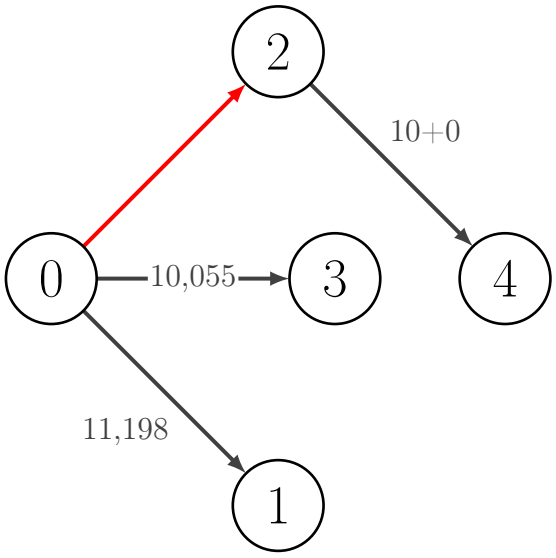
As arestas vermelhas representam os portais

Neste exemplo Linque possui 1 de energia e pode utilizar no máximo 1 portal. Então a pergunta que queremos responder é se existe caminho que leva do vértice 0 ao vértice 4 de custo no máximo 1 e que cruza no máximo 1 portal. De fato esse caminho existe e é $0 \rightarrow 3 \rightarrow 4$. Mas note que para o A* a resposta será 0. Vamos observar o passo a passo da execução do algoritmo.

Os vértices de destino dos arcos pretos são os elementos na fila de prioridade, com custo descrito no arco.



Inicialmente os três vizinhos de 0 são colocados na fila de prioridades. Note que estamos somando ao custo a heurística da distância para o vértice 4, que é o nosso vértice de destino.



Mesmo sabendo que o caminho mínimo é passando pelo vértice 3, como ele utiliza um portal seu custo seria 0, mas a heurística considera a distância, mesmo que ela não seja contabilizada no custo final. Dessa forma o menor elemento na fila de prioridades ainda é o caminho passando pelo vértice 2. Note que ao remover este elemento estamos removendo o vértice de destino da fila, então o algoritmo deve parar. A quantidade de portais que utilizamos foi 1, no entanto menor caminho encontrado então foi de custo 10, mas o custo máximo permitido pela instância é 1, logo a resposta para o A* deve ser 0.

Documentação

A documentação do trabalho deve ser entregue em formato **PDF** e também **DEVE** seguir o modelo de relatório que será postado no Moodle. Além disso, a documentação deve conter **TODOS** os itens descritos a seguir **NA ORDEM** em que são apresentados:

1. **Capa:** Título, nome, e matrícula.
2. **Introdução:** Contém a apresentação do contexto, problema, e qual solução será empregada.
3. **Método:** Descrição da implementação, detalhando as estruturas de dados, tipos abstratos de dados (ou classes) e funções (ou métodos) implementados.
4. **Análise de Complexidade:** Contém a análise da complexidade de tempo e espaço dos procedimentos implementados, formalizada pela notação assintótica.
5. **Estratégias de Robustez:** Contém a descrição, justificativa e implementação dos mecanismos de programação defensiva e tolerância a falhas implementados.
6. **Análise Experimental:** Apresenta os experimentos realizados em termos de desempenho computacional e localidade de referência, assim como as análises dos resultados.
7. **Conclusões:** A Conclusão deve conter uma frase inicial sobre o que foi feito no trabalho. Posteriormente deve-se sumarizar o que foi aprendido.
8. **Bibliografia:** Contém fontes utilizadas para realização do trabalho. A citação deve estar em formato científico apropriado que deve ser escolhido por você.
9. Número máximo de páginas incluindo a capa: 10

Não se esqueça de descrever de forma concisa sua implementação do TAD de fila de prioridade. Analise o impacto das sua escolhas na complexidade do algoritmo, compare com outras possíveis estruturas que também resolveriam o problema.

A documentação deve conter a descrição do seu trabalho em termos funcionais, dando foco nos algoritmos, estruturas de dados e decisões de implementação importantes durante o desenvolvimento.

Evite a descrição literal do código-fonte na documentação do trabalho.

Dica: Sua documentação deve ser clara o suficiente para que uma pessoa (da área de Computação ou não) consiga ler, entender o problema tratado e como foi feita a solução.

Como será feita a entrega

Você deve utilizar a linguagem C ou C++ para o desenvolvimento do seu sistema. O uso de estruturas pré-implementadas pelas bibliotecas-padrão da linguagem ou terceiros é terminantemente vetado. Você DEVE utilizar a estrutura de projeto abaixo junto ao Makefile:

```
– TP
    |– src
    |– bin
    |– obj
    |– include
    Makefile
```

A pasta **TP** é a raiz do projeto; a pasta **bin** deve estar vazia; **src** deve armazenar arquivos de código (*.c, *.cpp, ou *.cc); a pasta **include**, os cabeçalhos (headers) do projeto, com extensão *.h, por fim a pasta **obj** deve estar vazia. O Makefile deve estar na raiz do projeto. A execução do Makefile deve gerar os códigos objeto *.o no diretório **obj** e o executável do TP no diretório **bin**. O arquivo executável **DEVE** se chamar **tp2.out** e deve estar localizado na pasta **bin**. O código será compilado com o comando:

```
make all
```

O seu código será avaliado através de uma **VPL** que será disponibilizada no moodle. Você também terá à disposição uma VPL de testes para verificar se a formatação da sua saída está de acordo com a requisitada. A VPL de testes não vale pontos e não conta como trabalho entregue. Um pdf com instruções de como enviar seu trabalho para que ele seja compilado corretamente estará disponível no moodle.
A documentação será entregue em uma atividade separada designada para tal no Moodle. A entrega deve ser feita em um único arquivo com extensão .pdf, com nomenclatura nome_sobrenome_matricula.pdf, onde nome, sobrenome e matrícula devem ser substituídos por suas informações pessoais.

Avaliação

O trabalho será avaliado de acordo com:

- Corretude na execução dos casos de teste - (30% da nota total)
- Indentação, comentários do código fonte e uso de boas práticas - (10% da nota total)
- [Documentação] Definição e implementação das estruturas de dados e funções - (20% da nota total)
- [Documentação] Apresentação da análise de complexidade das implementações - (20% da nota total)
- [Documentação] Análise experimental - (20% da nota total)

Se o programa submetido não compilar[1], seu trabalho não será avaliado e sua nota será 0. Trabalhos entregues com atrasos sofrerão penalização de 2^{d-1} pontos, com d = dias de atraso.

[1] Entende-se por compilar aquele programa que, independente de erros no Makefile ou relacionados a problemas na configuração do ambiente, funcione e atenda aos requisitos especificados neste documento em um ambiente Linux.

Considerações finais

1. Comece a fazer esse trabalho prático o quanto antes, enquanto o prazo de entrega está tão distante quanto jamais estará.
2. Leia atentamente o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.
3. Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.
4. Plágio é crime. Trabalhos onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas (em relação a todos os envolvidos). Discussões a respeito do trabalho entre colegas são permitidas. É permitido consultar fontes externas, desde que exclusivamente para fins didáticos e devidamente registradas na seção de bibliografia da documentação. **Cópia e compartilhamento de código não são permitidos.**

FAQ (Frequently asked Questions)

1. Posso utilizar qualquer versão do C++? NÃO, o corretor da VPL utiliza C++11.
2. Posso fazer o trabalho no Windows, Linux, ou MacOS? SIM, porém lembre-se que a correção é feita sob o sistema Linux, então certifique-se que seu trabalho está funcional em Linux.
3. Posso utilizar alguma estrutura de dados do tipo Queue, Stack, Vector, List, e etc..., do C++? NÃO.
4. Posso utilizar smart pointers? NÃO.
5. Posso utilizar o tipo String? SIM.
6. Posso utilizar o tipo String para simular minhas estruturas de dados? NÃO.
7. Posso utilizar alguma biblioteca para tratar exceções? SIM.
8. Posso utilizar alguma biblioteca para gerenciar memória? SIM.
9. As análises e apresentação dos resultados são importantes na documentação? SIM.
10. Os meus princípios de programação ligados a C++ e relacionados a engenharia de software serão avaliados? NÃO.
11. Posso fazer o trabalho em dupla ou em grupo? NÃO.
12. Posso trocar informações com os colegas sobre a teoria? SIM.
13. Posso utilizar IDEs, Visual Studio, Code Blocks, Visual Code, Eclipse? SIM.