UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA
NÚCLEO DE COMPUTAÇÃO - NUCOMP

ALGORITMOS E ESTRUTURAS DE DADOS II 2024/2
PROF. FLÁVIO JOSÉ MENDES COELHO

PROJETO PRÁTICO 3 (versão 1)

1 Objetivos

Este **Projeto Prático 3** – **PP3**, tem o objetivo de exercitar e avaliar suas habilidades em:

- Aplicar os algoritmos e estruturas de dados abordados em AED 2 em um problema contextualizado apresentado neste enunciado;
- Aplicar e codificar um TAD **grafo ponderado** com **lista de adjacência** em C++20 (veja os slides das aulas de AED 2), para o problema proposto nesse enunciado;
- Aplicar e codificar um TAD **fila de prioridade mínima** com um **heap binário mínimo** em C++20 (veja os slides das aulas de AED 2 e a referência [**?**]), para o problema proposto nesse enunciado;
- Aplicar e codificar o algoritmo de **Dijkstra para caminhos mínimos** com um TAD **fila de prioridade mínima** em C++20 (veja os slides das aulas de AED 2 e a referência [**?**]), para o problema proposto nesse enunciado;
- Aplicar e codificar o algoritmo de **Kruskal** (veja os slides das aulas de AED 2 e a referência [**?**]) para obter uma **árvore geradora mínima** (*minimum spanning tree – MST*) utilizando um TAD **Union-Find** (veja a referência [**?**]) em C++20, para o problema proposto nesse enunciado;
- Solucionar diversos subproblemas relacionados ao problema principal;
- Avaliar seu domínio sobre o código que você próprio desenvolveu neste projeto, a partir dos assuntos abordados na disciplina AED 2.

2 Descrição do problema

Ano: **2045**. O mundo está assombrado com a notícia vinda de um país cujos governantes não valorizam seus professores, nem seus cientistas! Pesquisadores brasileiros desenvolvem uma técnica fantástica para tratar células cerebrais cancerígenas! A técnica consiste em injetar nano-robôs no cérebro doente que caminham pela rede de neurônios do cérebro, curando blocos de neurônios doentes. Um *neurônio* é um célula nervosa que se conecta a outros neurônios por meio de ligações denominadas *sinapses* (os *dendritos* são ramos do corpo de um neurônio que se ligam a dendritos de outro neurônio por meio das sinapses, que são como "pontos de ligação" entre os dendritos de neurônios distintos). São as sinapses que permitem que sinais químicos/elétricos sejam transmitidos entre um neurônio e outro da imensa rede de 86 bilhões de neurônios do cérebro humano[1]. Para explicar o processo de forma simplificada, os pesquisadores criaram um modelo do cérebro representado por um grafo, onde blocos de neurônios são representados por vértices do grafo, e sinapses ligando neurônios (e ligando blocos de neurônios) são as arestas do grafo, como mostrado na Figura 1. Nesta figura, os vértices 11 e 5 representam, respectivamente, os blocos de entrada e saída do robô. As setas em vermelho indicam o caminho a ser percorrido pelo robô. Vértices em cinza representam blocos de neurônios doentes. Um bloco é doente se contém pelo menos um neurônio doente; ou é sadio, em caso contrário. Em um bloco doente, neurônios doentes estão marcados em cor preta (veja a Figura 2). O processo executado por um nano-robô é descrito à seguir:

1. O robô inicia em um bloco de neurônios (ponto de entrada no cérebro);
2. O robô calcula um percurso pela rede de neurônios até alcançar o bloco de saída do cérebro. Este percurso deve ser mínimo, pois o robô dispõe de pouca energia. Assim, utilize o algoritmo de Dijkstra com uma fila de prioridade mínima;
3. O robô caminha pelo percurso calculado visitando os blocos de neurônios do caminho;
4. Ao visitar um bloco de neurônios o robô identifica se está diante de um bloco doente ou de um bloco sadio;
5. Se o bloco for doente, o nano-robô (para economizar tempo e energia), determina a MST (*minimum spanning tree*) dentro do bloco de neurônios doentes, e caminha na MST injetando uma enzima sintética no núcleo de cada neurônio do bloco (doente ou não), que corrige o código genético do neurônio, tornando-o sadio (a enzima é inócua às células sadias). Se o bloco não for doente o robô não faz nada e segue para o próximo passo (observe que somente serão calculadas as MSTs dos blocos doentes do percurso do robô). Para determinar cada MST, implemente o algoritmo de Kruskal com um Union-Find;
6. O robô abandona o bloco visitado e parte em direção do próximo bloco do percurso.
7. Os passos de 4-6 se repetem até o robô concluir o percurso chegando ao bloco que representa o ponto de saída do cérebro.

[1]Azevedo F. A., Carvalho L. R., Grinberg L. T., Farfel J. M., Ferretti R. E., Leite R. E., Jacob Filho W., Lent R., Herculano-Houzel S. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. J Comp Neurol. 2009 Apr 10;513(5):532-41. doi: 10.1002/cne.21974.

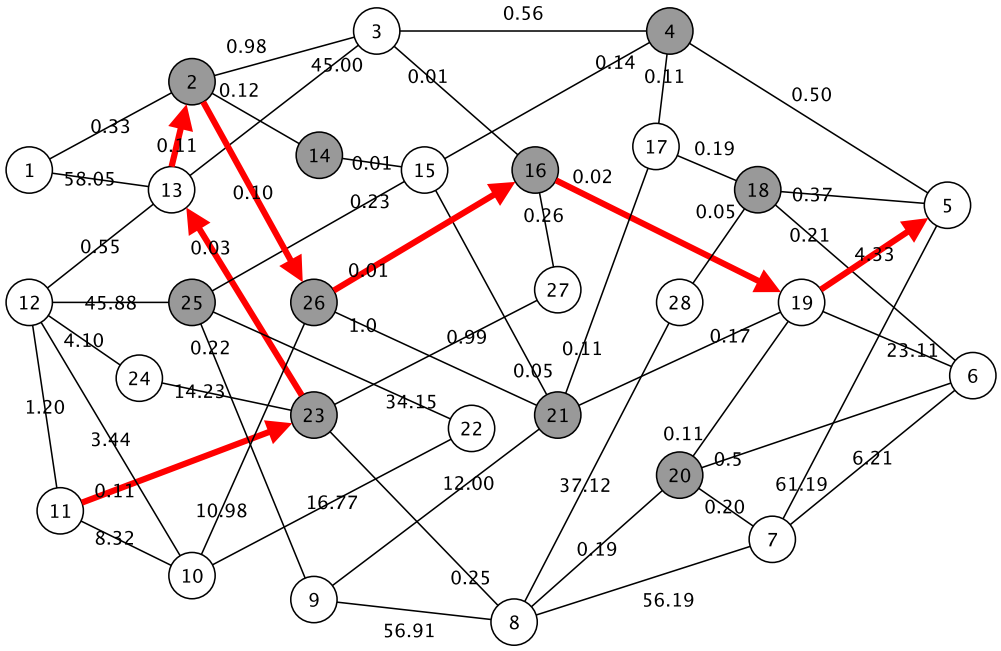

Figura 1: Grafo representando blocos de neurônios do cérebro e o percurso de um nano-robô.

Durante o percurso, nem todos os blocos identificados pelo robô são doentes. Para tentar "forçar" o robô a passar por alguns blocos doentes para curá-los, os cientistas injetam uma solução no cérebro do paciente que faz o robô interpretar as sinapses que se conectam a um neurônio doente como tendo um valor de comprimento muito pequeno. Assim, ao processar o cálculo do percurso, haverá maior chance do nano-robô passar por esse neurônio doente (pois a "distância" da sinapse até ele terá um valor menor do que as sinapses que se ligam a neurônios sadios). **Você e sua equipe devem desenvolver o software que controla o nano-robô!**

3 Entradas e saídas do problema

Entradas. As entradas serão lidas a partir de um juiz online, e compreendem as seguintes sequências:

1. O grafo do cérebro, seguido dos blocos de entrada e saída do cérebro. Exemplo:

```
28 52 <= ordem e tamanho do grafo do cérebro
1 2 0.33   <= 1a. aresta ligando os blocos 1 e 2, com valor 0.33
2 14 0.12  <= 2a. aresta ligando os blocos 2 e 14, com valor 0.12
etc...
```

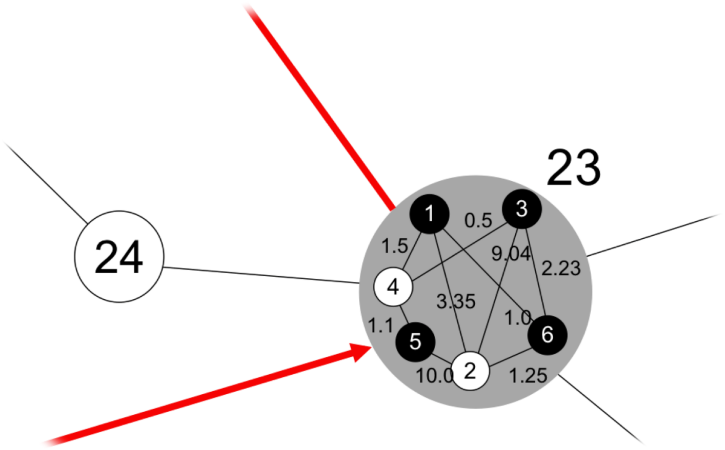

Figura 2: Bloco com neurônios doentes (em preto).

```
8 28 37.12 <= 52a.  aresta ligando os blocos 8 e 28, com valor 37.12
11 5       <= Blocos de entrada e saída
```

2. Os grafos de cada bloco de neurônios, na ordem de cada bloco dada no grafo do cérebro. Exemplo:

```
6 4  <= ordem e tamanho do grafo do bloco 1
0 <= número de neurônios doentes
1 3 12.5  <= aresta ligando o neurônio 1 ao neurônio 3, e sua distância
4 5 9.6   <= aresta ligando o neurônio 4 ao neurônio 5, e sua distância
5 7 0.3
7 8 1.8
5 6   <= ordem e tamanho do grafo do bloco 2
3 <= número de neurônios doentes
1 3 6 <= neurônios doentes
1 2 1.5  <= aresta ligando o neurônio 1 ao neurônio 2, e sua distância
1 5 1.9
2 3 7.0
2 4 8.46
2 5 12.66
4 5 2.1
9 8   <= ordem e tamanho do grafo do bloco 3
etc...
6 9 <= ordem e tamanho do grafo do bloco 23
4 <= número de neurônios doentes
1 6 3 5 <= neurônios doentes
1 4 1.5
1 2 3.35
1 6 1.0
```

```
2 5 10.0
2 3 9.04
2 6 1.25
3 4 0.5
3 6 2.23
4 5 1.1
etc... segue até o bloco 28
```

Saída. Deve ser apresentado a soma dos pesos de cada MST visitada no caminho percorrido pelo nano-robô (na verdade, o nano-robô não irá percorrer cada MST, apenas calcular seu valor).

4 Requisitos do projeto

1. **Equipes**. Este projeto deve ser desenvolvido por duplas ou individualmente. **Não serão aceitas equipes com mais de dois participantes**.

2. **Ferramentas e técnicas**. O projeto deve ser codificado em C++20. Os contêineres da STL permitidos são `std::list` e `std::pair`. O uso de qualquer outra biblioteca não é permitido, com exceção de `iostream`, `list`, `cstdlib`, `limits` e `iomanip`. Pelo menos os TADs das estruturas de dados devem ser codificadas com programação orientada a objetos. Como compilador local, sugerimos o GCC (Linux e Windows) ou o Clang (Mac OS). Para codificação local, sugerimos o editor/IDE `https://www.sublimetext.com/` ou `https://code.visualstudio.com/` combinado com o compilador local C++20 sugerido acima. Para codificação online, sugerimos o IDE `https://cpp.sh/` com suporte ao C++20.

3. **Correção e pontuação**. A correção e pontuação do PP3 é dividida em duas etapas:

   Correção funcional - CF. Essa correção é realizada de forma automática pelo juiz on-line ao qual a equipe submeterá o solução/código do PP3. Essa correção tem pontuação máxima igual a $10,0$ pts. A equipe ou aluno pode submeter o código ao juiz on-line quantas vezes quiser, até a data e hora do prazo final. Ao submeter uma solução, o juiz on-line executará N casos de teste para testar a corretude da solução submetida. Portanto, pontuação da solução submetida corresponde ao número de casos de teste corretos, ou seja, $10,0/N * C$, onde N é o número de casos de teste do juiz on-line e C é o número de casos de teste corretos. Os detalhes sobre a submissão serão repassados pelo professor, por e-mail ou pelo Classroom.

   Correção de Inspeção de código - CIC. Essa correção é realizada manualmente pelo professor que examinará a última versão do código submetido ao juiz on-line para verificar se esse código atende aos requisitos desse enunciado. Em caso positivo, o professor confirma os pontos obtidos na CF. Em caso contrário, se um ou mais requisitos não forem seguidos, a pontuação obtida na CF sofrerá uma série de penalizações reduzindo o valor obtido na CF. Isso é necessário porque uma solução pode obter a nota máxima nas submissões ao juiz on-line, mas não atender a um ou mais requisitos solicitados em um enunciado. Por exemplo, um requisito

pode pedir a implementação de uma algoritmo ou estrutura de dados X e, como solução, a equipe implementa um algoritmo ou estrutura de dados Y, diferente do que foi solicitado. Dessa forma, apresentar quaisquer outras soluções (algoritmos, estruturas de dados, linguagem de programação, etc) que não sejam as especificadas na tabela abaixo, resulta nas penalidades (acumulativas) sobre a CF para cada requisito não atendido:

Tabela 1: Requisitos do PP3 que devem ser atendidos

Tipo	**Requisito**	**Penalidade**
AED	TAD grafo ponderado com lista de adjacência em C++20, conforme referência e sem usar uma estrutura de dados pronta de qualquer biblioteca	-20% do CF
AED	TAD fila de prioridade mínima com um heap binário mínimo em C++20, conforme referência e sem usar uma estrutura de dados pronta de qualquer biblioteca	-20% do CF
AED	Algoritmo de Dijkstra para caminhos mínimos com um TAD fila de prioridade mínima em C++20, conforme referência e sem usar uma estrutura de dados pronta de qualquer biblioteca	-20% do CF
AED	Algoritmo de Kruskal para obter uma árvore geradora mínima (*minimum spanning tree – MST*) utilizando um TAD Union-Find em C++20, conforme referência e sem usar uma estrutura de dados pronta de qualquer biblioteca	-20% do CF
Linguagem	C++20	-70% do CF
Linguagem	Uso incorreto de funções e parâmetros (ponteiros, constantes, referências, etc), templates e classes C++; uso de variáveis globais	-10% do CF
Técnicas	POO (classes encapsuladas) e programação estruturada	-10% do CF
Estilo de código	Indentação K&R ou Allman correta (veja https://en.wikipedia.org/wiki/Indentation_style) e nomeação de variáveis com Camel case ou Snake case ou Pascal case (veja https://en.wikipedia.org/wiki/Naming_convention_(programming))	-10% do CF

Datas

- Emissão deste enunciado: 17/10/2024 às 15:00 (hora local).
- Abertura do juiz online: 19/10/2024 às 16:00 (hora local).
- Fechamento do PP3 no juiz on-line: 06/11/2024 às 22:00 (hora local).

CÓDIGO DE ÉTICA

Este projeto é uma avaliação acadêmica e deve ser concebido, projetado, codificado e testado pela equipe, com base nas referências fornecidas neste enunciado ou nas aulas de Algoritmos e Estruturas de Dados, ou por outras referências indicadas pelo professor, ou com base em orientações do professor para com a equipe, por solicitação desta. Portanto, não copie código pronto da Internet para aplicá-lo diretamente a este projeto, não copie código de outras equipes, não forneça seu código para outras equipes, nem permita que terceiros produzam este projeto em seu lugar. Isto fere o código de ética desta disciplina e implica na atribuição da nota mínima ao trabalho.

Referências

[1] COELHO, Flávio. Slides das aulas de *Algoritmos e Esruturas de Dados II*. Disponível no Google Classroom da disciplina. Universidade do Estado do Amazonas, Escola Superior de Tecnologia, Núcleo de Computação - NUCOMP. Semestre letivo 2016/2.

[2] C++. In: *WIKIPÉDIA, a enciclopédia livre*. Flórida: Wikimedia Foundation, 2016. Disponível em: <https://pt.wikipedia.org/w/index.php?title=C%2B%2B&oldid=45048480>. Acesso em: 17 abr. 2016.

[3] C++. In: *cppreference.com*, 2016. Disponível em <http://en.cppreference.com/w/>. Acesso em: 17 abr. 2016.

[4] CORMEN, T. H., Leiserson, C. E., Rivest, R. L., Stein C. *Introduction to Algorithms*, 3rd edition, MIT Press, 2010

[5] R. Sedgewick, K. Wayne. Algorithms. 4th edition, Addison-Wesley Professional, 2011

[6] STROUSTRUP, Bjarne. *The C++ Programming Language*. 4th. Edition, Addison-Wesley, 2013.

[7] STROUSTRUP, Bjarne. *A Tour of C++*. Addison-Wesley, 2014.

[8] SZWARCFITER, Jayme Luiz et. alii. *Estruturas de Dados e seus Algoritmos*. Rio de Janeiro. 2a. Ed. LTC, 1994.

[9] WIRTH, Niklaus. *Algoritmos e Estruturas de Dados*. Rio de Janeiro. 1a. Ed. Prentice - Hall do Brasil Ltda., 1989.

[10] ZIVIANI, Nívio. *Projeto de Algoritmos com Implementação em Java e C++*. 2a. Edição. Cengage Learning, 2010.

[11] ZIVIANI, Nívio. *Projeto de Algoritmos com Implementação em Pascal e C*. 3a. Ed. São Paulo: Cengage Learning, 2012.