

## PROJETO PRÁTICO 2

### 1 Objetivos

Este **Projeto Prático 2 – PP2**, tem o objetivo de exercitar e avaliar suas habilidades em:

- Codificar uma ou mais das seguintes estruturas de dados ou uma combinação delas: LISTA, PILHA, FILA e GRAFO (ou outras que você mostre serem eficientes para o problema) na linguagem de programação exigida neste enunciado, e apresentar soluções/algoritmos simples e eficientes para o problema;
- Codificar e aplicar corretamente os algoritmos para caminhos mínimos e MST (*minimum spanning tree*) no vistos em sala de aula, no contexto do problema do projeto.
- Demonstrar seu domínio sobre o código que você desenvolver neste projeto e mostrar que sabe realizar modificações locais no código do projeto, caso seja pedido pelo avaliador;
- Apresentar argumentos lógicos, razoáveis, para questões relativas ao código e às soluções empregadas no projeto.

### 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, há séculos! 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 uma 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 que “os 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<sup>1</sup>. Para explicar o processo de forma

---

<sup>1</sup>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.

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 (algoritmo de Dijkstra), pois o robô dispõe de pouca energia);
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 (algoritmo de Kruskal), 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);
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.

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:

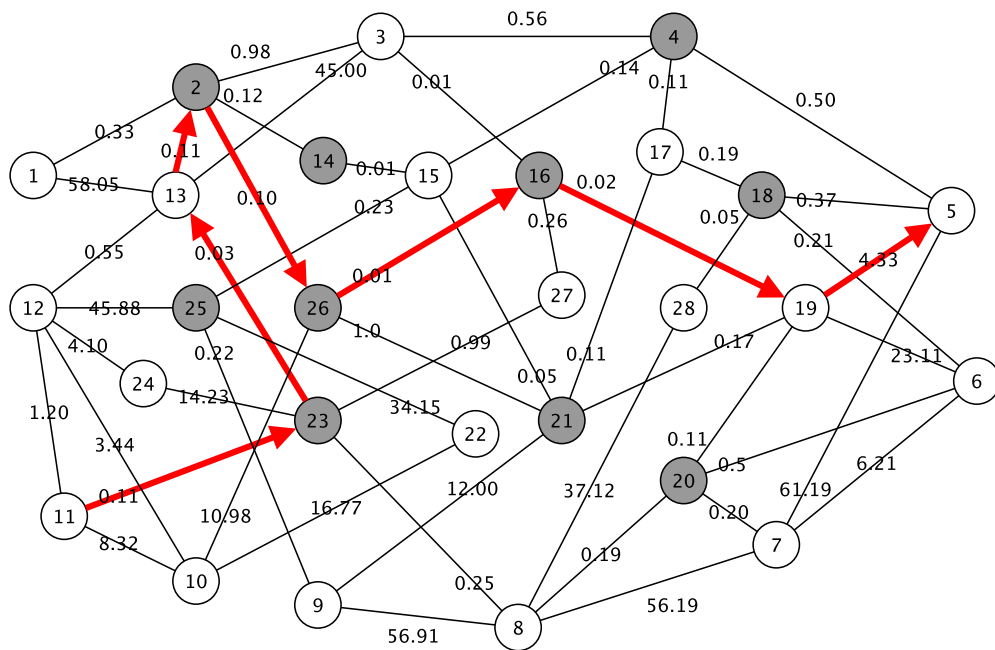


Figura 1: Grafo representando blocos de neurônios do cérebro e o percurso de um nano-robô.

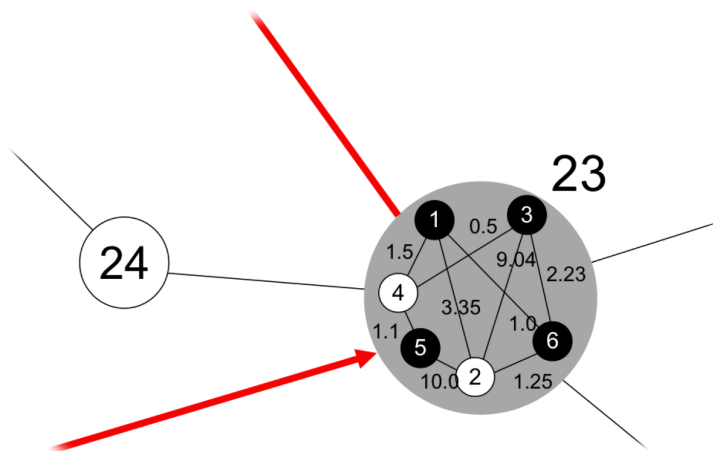


Figura 2: Bloco com neurônios doentes (em preto).

```

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 <= 2a. aresta ligando os blocos 2 e 14, com valor 0.12
etc...
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 sistema 28

```

**Saída.** Deve ser apresentado o total das somas dos custos de cada MST do caminho percorrido pelo nano-robô (o nano-robô não irá percorrer cada MST, apenas calcular seu valor).

## 4 Requisitos do projeto

1. **Equipes.** Este projeto deve ser desenvolvido por uma equipe de três ou dois estudantes. Não serão aceitas equipes com menos de dois ou mais de três participantes. A violação deste requisito do projeto penalizará a equipe com a subtração de 50% dos pontos totais obtidos no projeto.
2. **Ferramentas e técnicas.** O projeto deve ser codificado em C++11 utilizando programação orientada a objetos com encapsulamento, pelo menos nos TADS empregados.
3. **Pontuação.** O PP1 vale de 0,0 (nota mínima) a 10,0 (nota máxima), e será avaliado em duas fases:

**Fase 1 - Avaliação funcional.** Pontuação mínima: 0,0; pontuação máxima: 5,0.

O código do projeto será submetido ao juiz online **run.codes**, e obterá a nota máxima desta fase, se passar em todos os casos de teste (cada caso terá uma pontuação correspondente a uma fração da nota máxima desta fase). Os detalhes sobre a submissão serão repassados pelo professor, por e-mail.

**Fase 2 - Inspeção de código.** Pontuação mínima: 0,0; pontuação máxima: 5,0.

Nesta etapa, o professor escolherá um membro da equipe para defender o projeto (qualquer membro ausente no momento da escolha do membro defensor receberá a nota mínima integral no projeto). Na inspeção de código, serão feitas perguntas sobre detalhes de implementação do projeto de acordo com os seguintes critérios:

- **Legitimidade** (critério eliminatório). A constatação de que o projeto é plágio implica na atribuição automática da nota mínima integral para o projeto incluindo as duas etapas de avaliação. Será considerado legítimo o projeto que tiver similaridade de código entre equipes menor do que 40% (comparador do juiz online do run.codes).
- **Segurança na defesa** ao responder as perguntas do avaliador e explicar as estratégias utilizadas.
- **Cumprimento de requisitos:** o projeto atende ao que foi solicitado neste enunciado.
- **Qualidade de código:** indentação, uso de TADS orientados a objetos (encapsulados), uso do C++11\*, programação genérica (quando aplicável), qualidade de código (implementação simples e eficiente das estruturas de dados, boa nomeação de identificadores, escolha das estruturas de dados e algoritmos mais eficientes em desempenho), e criatividade. **Importante:** remova todos os comentários do seu código.
- **Uso de bibliotecas:** o projeto poderá empregar somente os *containers* *pair* e *vector* da STL. Qualquer outro recurso da STL ou de outras bibliotecas não deve ser utilizado, com exceção de: *iostream*, *cstring*, *cstdlib* e *limits*. O uso de qualquer outro arquivo de cabeçalho padrão deve ser verificado com o professor.

\* Compile seu projeto em vários compiladores online (além do compilador do juiz online) para garantir que o mesmo não apresente problemas sintáticos/semânticos (nem *warnings*). Localmente, aplique as diretivas de compilação mais restritas do seu compilador.

## 5 Datas

- Emissão deste enunciado: 12/11/2017 às 19h15min (hora local).
- Abertura do juiz online: 24/11/2017 às 6h (hora local).
- Fechamento do juiz online: 03/12/2017 às 23h (hora local).
- Inspeção de código: 04/12/2017 de 9h às 19h e 06/12/2017 das 9h às 19h (hora local).

As datas podem ser modificadas para atender demandas da turma ou do professor. Neste caso, as mudanças serão comunicadas ao inscritos no grupo de AED2 por e-mail.

---

### 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 Estruturas de Dados II*. Disponível em <https://est.uea.edu.br/fcoelho>. 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] KNUTH, Donal E. *Fundamental Algorithms*, 3rd.ed., (vol. 1 de The Art of Computer Programming), Addison-Wesley, 1997.
- [6] KNUTH, Donal E. *Seminumerical Algorithms*, 3rd.ed., (vol. 2 de The Art of Computer Programming), Addison-Wesley, 1997.
- [7] KNUTH, Donal E. *Sorting and Searching*, 2nd.ed., (vol. 3 de The Art of Computer Programming), Addison-Wesley, 1998.

- [8] STROUSTRUP, Bjarne. *The C++ Programming Language*. 4th. Edition, Addison-Wesley, 2013.
- [9] STROUSTRUP, Bjarne. *A Tour of C++*. Addison-Wesley, 2014.
- [10] SZWARCFITER, Jayme Luiz et. alii. *Estruturas de Dados e seus Algoritmos*. Rio de Janeiro. 2a. Ed. LTC, 1994.
- [11] WIRTH, Niklaus. *Algoritmos e Estruturas de Dados*. Rio de Janeiro. 1a. Ed. Prentice - Hall do Brasil Ltda., 1989.
- [12] ZIVIANI, Nívio. *Projeto de Algoritmos com Implementação em Java e C++*. 2a. Edição. Cengage Learning, 2010.
- [13] ZIVIANI, Nívio. *Projeto de Algoritmos com Implementação em Pascal e C*. 3a. Ed. São Paulo: Cengage Learning, 2012.