TRABALHO PRÁTICO DE AEDs III (2º Sem/ 2021)

Valor do trabalho: 40 pontos. Data de entrega: **06/12**. O trabalho pode ser feito individualmente ou em dupla.

Os modelos matemáticos e computacionais e as simulações computacionais estão se tornando cada vez mais importantes no mundo atual. A pandemia de Covid-19 está mostrando que as pesquisas científicas são fundamentais para o desenvolvimento de vacinas, novos medicamentos, para o desenvolvimento científico e tecnológico de um país. Muitas vezes um avanço na ciência ocorre a partir de simulações computacionais realizadas com um modelo construído para explicar determinado fenômeno. As simulações complementam experimentos reais e ajudam a desvendar os mecanismos por trás de diversos fenômenos do mundo real. Simulações computacionais são simulações realizadas em um computador com o objetivo de entender um fenômeno/sistema/problema de interesse.

Os modelos matemáticos e computacionais mais utilizados em diversas áreas são: Equações Diferenciais Ordinárias, Equações Diferenciais Parciais, Modelos baseados em Agentes, Autômatos Celulares, Modelos estocásticos e Redes Complexas, sendo que as Redes Complexas tem como base a estrutura de dados grafos.

O objetivo deste trabalho é desenvolver um simulador simples para estudar o espalhamento de epidemias em uma população.

O objetivo é simular epidemias ou pandemias através da utilização de um modelo clássico da literatura chamado **SIR** (**Suscetível-Infectado-Recuperado**), da utilização de uma estrutura de dados do tipo **Grafo** para representar as relações de parentesco e amizade entre indivíduos de uma sociedade, da utilização de um algoritmo de simulação guiado por eventos (chamado **event-driven simulation**), da utilização de uma estrutura de dados do tipo fila de prioridades (**heap**) para gerenciar os eventos e definir a ordem de execução deles e da utilização do formato **XML** para a representação da entrada de dados para o simulador. Resumindo, são vários conceitos importantes que vocês irão aprender e implementar na prática neste trabalho.

O modelo SIR é um modelo clássico da literatura sendo muito utilizado nas áreas de modelagem matemática e computacional para estudar o espalhamento de doenças. O modelo SIR com algumas modificações tem sido usado para estudar a pandemia de COVID-19 ajudando a fazer previsões e a responder questões importantes como:

- Como será a evolução no número de casos e mortes no estado de Minas Gerais?
- Qual é a taxa estimada de hospitalizações para a cidade de São João del-Rei para o mês de novembro?
- Qual o impacto da vacinação de 50% da população na redução do número de casos e mortes no Brasil?

É claro que todas as previsões feitas por um modelo possuem um erro associado, mas o importante é que muitas previsões conseguiram e conseguem prever com um erro relativamente pequeno vários indicadores como os mencionados anteriormente.

No modelo SIR, cada indivíduo de uma população pode estar em um determinado estado de saúde representando, por exemplo, se o indivíduo está saudável (também chamado de suscetível à doença) ou se foi infectado pelo vírus causador da epidemia, ou até

mesmo se a pessoa já se recuperou da doença e ficou “imune” a ela não sendo mais infectada. Então, o modelo SIR estuda a evolução temporal (ou evolução no tempo) da transmissão do vírus (por exemplo, Sars-CoV-2) através das mudanças de estados que ocorrem nos indivíduos da população durante um determinado tempo simulado. Para estudar isso, o modelo precisa definir algumas hipóteses e trabalhar com elas para tentar simular o comportamento real de uma epidemia. O nosso modelo trabalha com as seguintes hipóteses:

- Uma pessoa suscetível tem uma probabilidade de ser infectada se ela está próxima de pessoas infectadas. No caso do modelo a ser construído, a pessoa será infectada com uma determinada probabilidade se ela está conectada a pelo menos uma pessoa infectada. Esse evento é chamado de infecção.
- Uma pessoa infectada tem uma probabilidade de se recuperar independente das relações que tem na rede social. Essa probabilidade poderia depender da idade e de co-morbidades, por exemplo. Esse evento é chamado de recuperação.

O desenho abaixo mostra um diagrama de estados mostrando as possíveis mudanças de estado no modelo SIR:

A transição de um estado para outro pode depender de vários fatores e ser estocástica (depender de uma probabilidade como é o caso do modelo considerado aqui).

As relações de amizade e parentesco serão representadas através de um grafo não-direcionado. Cada nó do grafo representa um indivíduo (pessoa) de uma população e uma aresta ligando dois indivíduos representa uma relação de amizade ou parentesco entre eles. Poderá ser utilizada qualquer representação vista como matrizes esparsas, listas de adjacência, dicionários, ou vocês podem criar a própria representação ou utilizar uma biblioteca que já implemente a criação do grafo para vocês.

Cada nó do grafo, terá um atributo **estado** ou state para representar o estado de saúde daquela pessoa. O estado pode ser um dos três estados do modelo SIR: suscetível, infectado ou recuperado. Durante a epidemia, podem ocorrer eventos que vão provocar a mudança de um estado para outro. Cada evento tem algumas características, um conjunto de condições para que o evento ocorra e uma lista de ações a serem executadas caso as condições sejam verdadeiras. Todo evento tem as seguintes informações:

- **Nome**: o nome do evento;
- Tempo para ocorrer (chamado **time delay**): todo evento tem um atraso de tempo para que ele ocorra;
- **Probabilidade**: uma probabilidade para a ocorrência do evento;
- **Repetição**: um atributo booleano indicando se o evento poderá se repetir ao longo da simulação, ou seja, se ele poderá ser executado novamente.

Além dessas informações, um evento também pode ter uma ou mais condições onde cada condição tem as seguintes informações:

- O estado atual que será testado na condição representado por **mystate**;

- **Opcionalmente** o estado de um vértice vizinho (**neighbour_state**) que também será testado na condição. Quando o valor de neighbour_state for uma string vazia, quer dizer que não precisamos testar o estado dos vizinhos. Em outras palavras, o estado dos vizinhos poderia ser qualquer um para que a condição seja verdadeira.

Obs: Considera-se, neste modelo, que os vizinhos de um vértice são os vértices adjacentes a ele.

Um evento também é composto por uma ou mais ações, sendo que cada ação tem as seguintes informações:

- **Tipo da ação**: o tipo será sempre **update**;
- **Novo estado do vértice analisado**: a ação irá atualizar o estado do vértice com o valor dado em **newstate**.

Os eventos que podem ocorrer durante a epidemia serão representados por alguma estrutura (uma classe, struct, etc) que irá armazenar todas suas informações. Vocês devem definir e implementar essa estrutura. Os eventos serão inseridos e gerenciados por uma heap que irá implementar a fila de prioridades dos eventos. Cada evento tem um tempo associado e eles serão ordenados na fila **do menor tempo para o maior**. O evento com menor tempo estará na raiz (ou no topo) da heap. Nesse caso, a heap é chamada heap mínima. Se dois ou mais eventos tiverem o mesmo valor de tempo, eles poderão aparecer em qualquer ordem na heap em relação aos outros eventos com o mesmo tempo.

A ideia do algoritmo de simulação dado abaixo é executar um conjunto de passos no tempo até ser atingido o tempo de final de simulação e para cada passo executar os eventos que irão ocorrer naquele passo de tempo. Serão executados os eventos que estão na fila de prioridades cujo tempo desses eventos é menor ou igual ao tempo atual de simulação. O tempo do evento é calculado com base no tempo atual e no time delay do evento. Abaixo, é dado um exemplo de código em C++ que mostra o loop temporal e a utilização da fila de prioridades (eventQueue) para obter e remover da fila os eventos que serão executados. A cada passo de tempo, devem ser salvas as informações atuais do grafo (ver seção sobre isso a seguir), isto é, os valores dos estados de cada nó.

```
void run(){
    std::cout << "Running the simulation " << endl;
    save(0);
    for (double t = dt; t <= tfinal; t += dt) {
        if (eventQueue.empty()) break;
        Event *e = eventQueue.top();
        while (!eventQueue.empty() && e->getTime() <= t){
            //ex: e->execute(...): aqui vai a lógica para executar o evento
            eventQueue.pop();
            if (e->getRepeat()){
                e->setTime(t + e->getTimeDelay());
                eventQueue.push(e);
            }
            e = eventQueue.top();
        }
        if (isReferenceTime(refTimes,t))
            save(t);
    }
    std::cout << "Simulation ended " << endl;
}
```

A implementação dos eventos deverá fornecer um método para execução deles. A **execução** de um evento envolve percorrer todos os vértices do grafo verificando para cada vértice:

- Se um número aleatório gerado é menor ou igual a probabilidade do evento;

  Obs: Deve ser gerado um número aleatório a cada novo vértice visitado.

- Se as condições são satisfeitas;
- Caso as condições sejam satisfeitas, as ações são executadas e o estado do vértice atual é atualizado;
- Por último, se um evento irá se repetir, devemos calcular o novo tempo dele e inseri-lo novamente na fila.

  Destaca-se que uma condição pode ser testada antes da probabilidade. O importante é que ambas sejam verificadas para decidir se a ação será executada ou não.

  Antes de ser iniciada a simulação, os eventos lidos do arquivo XML devem ser inseridos na fila de prioridades definindo o tempo deles com base no time delay de cada.

Entrada de dados

As informações sobre estados, eventos e parâmetros da simulação podem ser definidas diretamente no código (alocando e preenchendo as estruturas utilizadas) ou **opcionalmente** podem ser lidas de um arquivo no formato XML. Um exemplo de entrada através de um arquivo XML é dado na figura abaixo.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<model name="pandemia">
    <simulation_params tfinal="20" dt="1"/>
    <states>
        <state name="S"/>
        <state name="I"/>
        <state name="R"/>
    </states>
    <events>
        <event name="infection" timedelay="1" probability="0.3" repeat="true">
            <conditions>
                <condition mystate="S" neighbour_state="I" />
            </conditions>
            <actions>
                <action type="update" newstate="I" />
            </actions>
        </event>
        <event name="recovery" timedelay="5" probability="0.7" repeat="true">
            <conditions>
                <condition mystate="I" neighbour_state="" />
            </conditions>
            <actions>
                <action type="update" newstate="R" />
            </actions>
        </event>
    </events>
</model>
```

O XML mostra que um modelo definido pela tag model tem um nome, dois parâmetros (tfinal e dt), um conjunto de estados definido pela tag states e um conjunto de eventos definidos pela tag events. Dentro de states, temos a definição dos estados do modelo SIR e dentro de eventos temos as definições dos eventos de infecção (infection) e recuperação (recovery). As informações presentes em um evento já foram explicadas anteriormente. Os estados lidos podem ser armazenados, por exemplo, em um vetor. Fica a critério de cada um, a forma como os estados serão definidos e armazenados.

Todo evento (event) tem um conjunto de condições (conditions) e ações (actions), mas no caso do modelo SIR clássico só precisamos de uma condição e uma ação para implementar cada evento.

O tempo de simulação dado em tfinal se refere ao tempo que irá se passar no sistema modelado. Considere que um tempo final igual a 100 corresponde a 100 dias acompanhando a evolução temporal do sistema modelado. Considere também que o passo de tempo (ou o incremento no tempo) será igual a 1 representando um incremento de um dia no tempo. Então, o passo de tempo (dt) será igual a 1 (um dia).

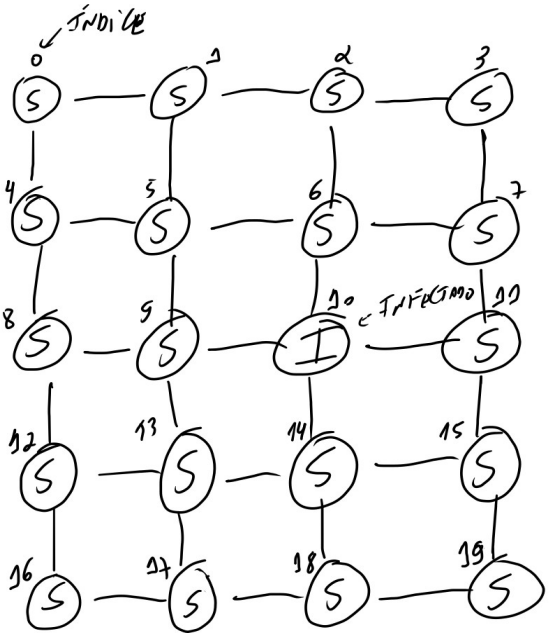
Criação do grafo

O grafo representando a rede social deve ser criado de duas formas diferentes:

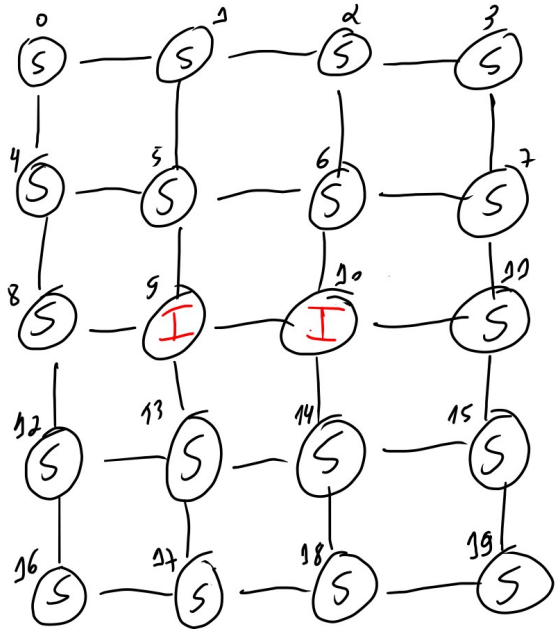
1) O grafo é criado como se fosse uma grade (grid) regular bidimensional como mostrado nas figuras abaixos. Abaixo, são mostrados os grafos para os tempos t = 0

(condição inicial), t = 1 e t = 2. Você pode criar o grafo definindo as arestas uma por uma ou uma outra forma é associar a cada nó uma posição (x,y) no espaço bidimensional e criar as arestas entre os vértices com base na posição.

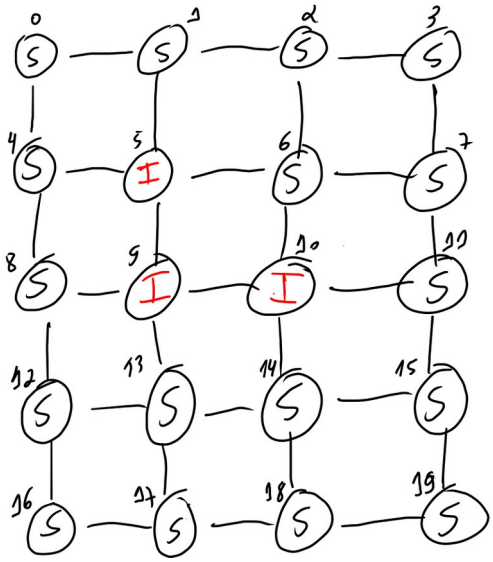
t = 0:



t = 1:

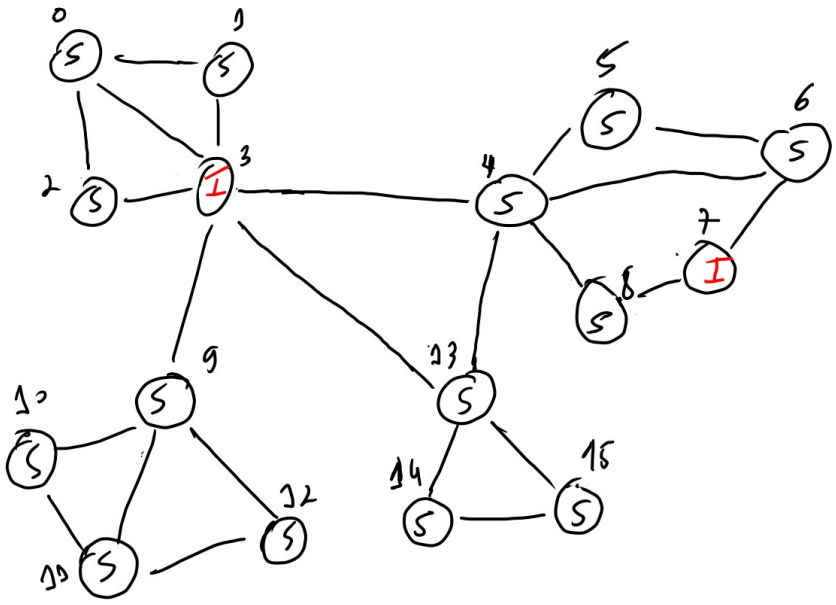


t = 2:

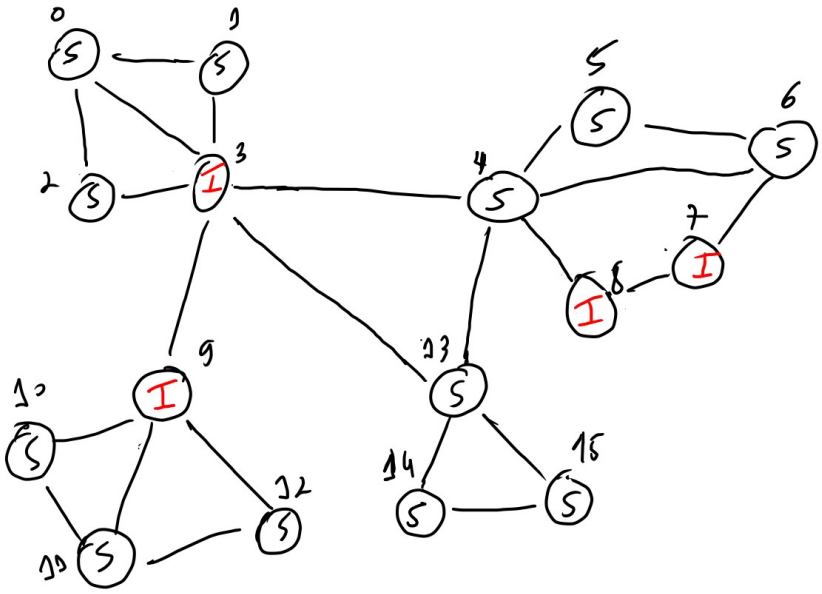


2) O grafo é criado dividindo-se os vértices em grupos, onde cada grupo pode ter de 3 a 6 vértices, por exemplo, e o número de grupos pode variar de 3 a 6. Os vértices de um grupo estão mais conectados entre si do que vértices de grupos diferentes. Até no máximo um ou dois vértices de um grupo são escolhidos para se ligar a outros grupos. O algoritmo de criação deve tentar não deixar vértices isolados. Algumas poucas ligações são criadas entre os grupos tentando obter uma rede do tipo Small World (mundo pequeno) onde com poucos “pulos” chegamos em qualquer lugar da rede/grafo. Abaixo, são mostradas três figuras de um grafo desse tipo para os tempos t = 0, t = 1 e t = 5. Repare que no tempo 5 já podem ocorrer eventos de recuperação considerando os times delays dados no XML de entrada mostrado anteriormente.

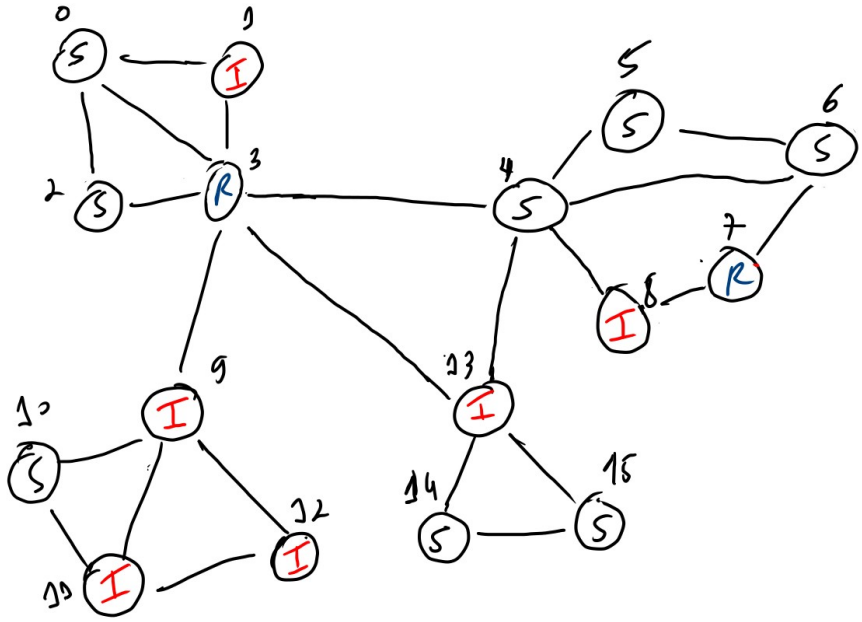
t = 0:



t = 1:



t = 5:



Impressão e exibição dos resultados

Os resultados de cada passo da simulação devem ser impressos em um arquivo de texto. Nos arquivos de texto com os resultados, deverão ser impressos sempre as **ligações do grafo no início**, posteriormente as **informações do grafo para cada passo de tempo** simulado e no final devem ser impressas as **porcentagens de infectados e recuperados ao final da simulação**. Quando o número de passos de tempo foi maior do que 10, você pode escolher salvar os resultados apenas para alguns passos de tempo. A exigência é salvar no mínimo 10 resultados diferentes. Por exemplo, com um tfinal igual a 100, poderíamos escolher salvar os passos 0, 10, 20, 30, 40, 50, 60, 70, 80, 90 e 100.

Opcionalmente, os resultados podem ser exibidos de forma visual usando alguma biblioteca para visualização de grafos.

Considerando um grafo em forma de grade regular com a mesma condição inicial do grafo da figura dada para t = 0, um exemplo de saída é mostrada abaixo considerando tfinal = 10 e dt = 1, onde primeiro são exibidas as arestas do grafo, depois os resultados da simulação para alguns passos de tempo e por último as porcentagens de infectados e recuperados.

Ligações do grafo:

0: {1, 4}

1: {0, 2, 5}

2: {1, 3, 6}

3: {2, 7}

4: {0, 5, 8}

5: {1, 4, 6, 9}

6: {2, 5, 7, 10}

7: {3, 6, 11}

8: {4, 9, 12}

9: {5, 8, 10, 13}

10: {6, 9, 11, 14}

11: {7, 10, 15}

12: {8, 13, 16}

13: {9, 12, 14, 17}

14: {10, 13, 15, 18}

15: {11, 14, 19}

16: {12, 17}

17: {13, 16, 18}

18: {14, 17, 19}

19: {15, 18}

Resultados da simulação:

t = 0:

Estado (0) = S

Estado (1) = S

Estado (2) = S

Estado (3) = S

Estado (4) = S

Estado (5) = S

Estado (6) = S

Estado (7) = S

Estado (8) = S

Estado (9) = S

Estado (10) = I

Estado (11) = S

Estado (12) = S

Estado (13) = S

Estado (14) = S

Estado (15) = S

Estado (16) = S

Estado (17) = S

Estado (18) = S

Estado (19) = S

t = 1:

Estado (0) = S

Estado (1) = S

Estado (2) = S

Estado (3) = S

Estado (4) = S

Estado (5) = S

Estado (6) = S

Estado (7) = S

Estado (8) = S

Estado (9) = I

Estado (10) = I

Estado (11) = S

Estado (12) = S

Estado (13) = S

Estado (14) = I

Estado (15) = S

Estado (16) = S

Estado (17) = S

Estado (18) = S

Estado (19) = S

t = 2:

Estado (0) = S

Estado (1) = S

Estado (2) = S

Estado (3) = S

Estado (4) = S

Estado (5) = I

Estado (6) = I

Estado (7) = I

Estado (8) = I

Estado (9) = I

Estado (10) = I

Estado (11) = I

Estado (12) = I

Estado (13) = I

Estado (14) = I

Estado (15) = I

Estado (16) = S

Estado (17) = S

Estado (18) = S

Estado (19) = S

…

t = 5:

Estado (0) = S

Estado (1) = I

Estado (2) = I

Estado (3) = I

Estado (4) = R

Estado (5) = I

Estado (6) = I

Estado (7) = R

Estado (8) = I

Estado (9) = I

Estado (10) = R

Estado (11) = I

Estado (12) = I

Estado (13) = R

Estado (14) = I

Estado (15) = R

Estado (16) = R

Estado (17) = I

Estado (18) = I

Estado (19) = I

t = 10:

Estado (0) = I

Estado (1) = I

Estado (2) = I

Estado (3) = R

Estado (4) = R

Estado (5) = I

Estado (6) = I

Estado (7) = R

Estado (8) = I

Estado (9) = I

Estado (10) = R

Estado (11) = R

Estado (12) = I

Estado (13) = R

Estado (14) = I

Estado (15) = R

Estado (16) = R

Estado (17) = I

Estado (18) = R

Estado (19) = R

Porcentagem de infectados = 50%

Porcentagem de recuperados = 50%

Linguagens e bibliotecas

O simulador pode ser desenvolvido em qualquer linguagem de programação e poderá utilizar as estruturas de dados e bibliotecas já disponíveis na linguagem utilizada ou outras bibliotecas que desejarem. Por exemplo, utilizar a estrutura priority_queue da STL (biblioteca padrão) da linguagem C++ para implementar a heap, ou utilizar a biblioteca NetworkX para trabalhar com o grafo na linguagem Python ou utilizar alguma biblioteca da linguagem Java e, assim por diante. Para a leitura do XML, é recomendável utilizar bibliotecas prontas disponíveis na linguagem de programação com a qual vocês estão trabalhando.

Simulações a serem feitas

Deverão ser realizadas simulações com o simulador construído a fim de que vocês verifiquem os comportamentos que podem ser obtidos com o nosso modelo de espalhamento de epidemias. Com base nos resultados, vocês poderiam até ir pensando e propondo melhorias para o modelo com o objetivo de deixar o comportamento dele mais próximo de uma epidemia ou pandemia real.

As simulações a serem realizadas serão separadas em cenários. **Para cada cenário simulado, deverá ser explicado qual foi o comportamento observado**. Deverá ser entregue para cada cenário o resultado dele juntamente com a análise feita do resultado.

Para as simulações, considere sempre que as probabilidades de infecção e recuperação devem ser maiores do que zero e menores ou iguais a 1 e considere o **número de nós do grafo entre 15 e 30**. Opcionalmente, você pode testar com grafos maiores.

Cenário 1

Considerando uma topologia do grafo em forma de uma grade regular e considerando uma pessoa infectada mais ou menos no meio do grafo (posicionada no meio da grade), tente determinar uma combinação de valores para as probabilidades de infecção e de recuperação a partir da qual pelo menos 90% da população é infectada em 30 dias. Faça o mesmo para 50 dias. Considere que o número inicial de recuperados é 0, que o timedelay da infecção é 1 e que o timedelay da recuperação é 5.

Cenário 2

Pegue os valores das probabilidades obtidos no cenário 1 e considere como condição inicial um grafo onde 50% das pessoas (50% dos nós) estão no estado **recuperado**

representando uma vacinação de 50% da população. Faça simulações para 30 e 50 dias e analise as mudanças no resultado em relação ao Cenário 1. Considere que o timedelay da infecção é 1 e que o timedelay da recuperação é 5.

Cenário 3

Considerando uma topologia do grafo em forma de uma grade regular e considerando uma pessoa infectada mais ou menos no meio do grafo (posicionada no meio da grade), fixando os valores das probabilidades, tente determinar uma combinação de valores para os timedelay tal que pelo menos 90% da população é infectada em 30 dias. Faça o mesmo para 50 dias.

Cenário 4

Refaça as análises do cenário 1 para o grafo Small World (grafo com grupos de vértice). Considere, inicialmente, que **duas** pessoas estão infectadas cada uma em um grupo de vértice diferente. Tente determinar uma combinação de valores para as probabilidades de infecção e de recuperação a partir da qual pelo menos 90% da população é infectada em 30 dias. Faça o mesmo para 50 dias. Considere que o número inicial de recuperados é 0, que o timedelay da infecção é 1 e que o timedelay da recuperação é 5.

Cenário 5

Pegue os valores das probabilidades obtidos no cenário 4 e considere como condição inicial um grafo onde 50% das pessoas (50% dos nós) estão no estado **recuperado** representando uma vacinação de 50% da população. Faça simulações para 30 e 50 dias e analise as mudanças no resultado em relação ao Cenário 1. Considere que o timedelay da infecção é 1 e que o timedelay da recuperação é 5. Considere o grafo com grupos de vértices (Small world).

Cenário 6

Considerando o grafo com grupos de vértices e que **duas** pessoas estão infectadas cada uma em um grupo de vértice diferente, fixando os valores das probabilidades, tente determinar uma combinação de valores para os timedelay tal que pelo menos 90% da população é infectada em 30 dias. Faça o mesmo para 50 dias.

Obs: Caso não consiga obter 90% de infecção da população, mostre e analise o resultado que chegou mais próximo disso.

O que entregar

Além do **código fonte** do trabalho, deverão ser entregues os seguintes documentos:

- Uma **documentação** explicando:
  1) Como o grafo foi implementado;
  2) A representação e implementação dos eventos;
  3) A implementação da fila de prioridades;
  4) Como foi feita a leitura do XML (**opcional**);
  5) Como os resultados foram salvos.

6) Arquivo README explicando como compilar e executar o código no sistema operacional Ubuntu 20.04.

- Pelos menos **dois resultados** (dois arquivos de texto de saída) **para cada cenário pedido**. **Cada resultado, deverá ser acompanhado de uma análise do comportamento observado**. A análise pode ser feita em um ou dois parágrafos.