Trabalho Trimestral de Grafos

Grupo 1

Alien Vs Ripley

Turma: 203

Integrantes:

João Pedro Moreira Smolinski - 2021951450

Thales Eduardo Dias de Souza - 2021951469

Izabela Maria da Silva Fonseca - 2021951493

gabriela souza gomes da costa - 2021951582

Sumário

EXPLICAÇÃO DO PROBLEMA - O ROTEIRO DO PROGRAMA : Por Gabriela

No futuro, em 2122, as viagens intergalácticas são possíveis, além de serem práticas comuns. Ripley faz parte de uma tripulação que trabalha em uma nave rebocadora, conhecida como Nostromos, que acaba recebendo um chamado de ajuda vindo de um planeta na trajetória da nave. Esse chamado acaba acordando a tripulação, que estava em câmaras de hibernação.

Após aterrissar nesse planeta, uma parte da equipe se dirige à origem do chamado, encontrando um ser alienígena morto por lá. Um dos tripulantes é, então, contaminado por um embrião depositado em sua garganta por uma pequena forma de vida desconhecida e semelhante a uma lagosta. Infelizmente, ninguém percebeu esse fato, fazendo com que o embrião utilizasse esse tripulante como hospedeiro, culminando com a morte do hospedeiro e nascimento de um Alien dentro da nave.

O Alien cresceu rapidamente, em progressão geométrica, causando sérios problemas dentro da nave da Ripley e resultando na morte de todos os seus colegas, deixando-a sozinha com o Alien, que passou a caçá-la. Ela se encontra na entrada da nave e começa a pensar em maneiras de sair viva dessa situação.

Sua intenção é chegar ao outro lado da nave, onde está localizada a saída e onde há uma nave auxiliar que pode levá-la de volta à Terra. Mas, isso não será uma tarefa fácil, pois embora possua uma visão primitiva, o Alien tem um alto instinto predatório e continuará se movimentando e sondando os corredores em procura de Ripley. Com tudo isso em mente, Ripley conseguiu formular um plano simples, porém eficiente.

Ela deve sair da entrada da nave e andar pelos corredores até achar a saída. Ela não pode se encontrar com o Alien de forma alguma. Por isso, se ele estiver em alguns dos

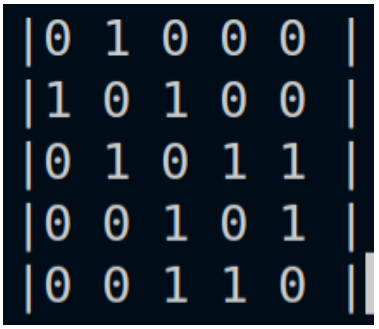
corredores em que Ripley deseja passar, ela terá de procurar outro caminho para chegar ao à saída ou esperar que o Alien saia do seu caminho.

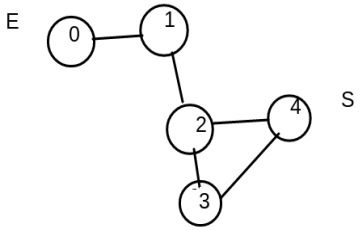
Durante sua viagem na Nostromos, ela não teve a chance de conhecer bem a nave porque estava em animação suspensa. Logo, ela não sabe o melhor caminho até a saída. Ela terá, então, que confiar em seus instintos e memória para não ser morta pelo Alien e evitar passar pelos lugares onde ela já esteve. Um verdadeiro jogo de estratégia, cujo prêmio é a sua própria vida!

Modelagem do Problema: Por Thales

Estrutura utilizada:

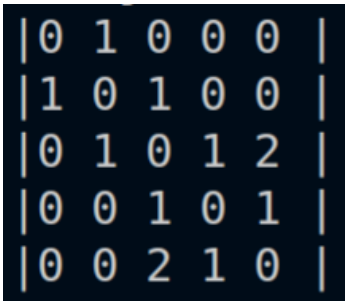
Para representar a nave em que Ripley se situa, utilizamos um grafo, para assim ser possível controlar as salas e os caminhos que as conectam. Dessa forma, a melhor forma encontrada para criar o grafo foi com uma **Matriz de Adjacências**, sendo a posição (i, j) da matriz o caminho entre o vértice i para o j. Como o grafo é não-orientado, o valor de (i, j) deve ser igual a (j, i), se ripley pode ir por um caminho ela também pode utilizar o mesmo para voltar. Representa-se 1 quando existe a conexão e 0 quando não existe.





Quando i == j, o valor será necessariamente 0, já que o vértice não pode estar conectado consigo mesmo, sendo a diagonal principal da matriz sempre nula.

Após representar a nave a partir da Matriz, é necessário colocar o alien, que interdita o caminho por um determinado tempo e depois aparece entre outros vértices, a aresta em que ele se localiza deve ser representada de modo que Ripley seja impedida de atravessar. Para isso se utiliza o número 2, quando (I, j) = 2, o alien está ali e permanecerá até o próximo movimento de Ripley.



A conexão entre o terceiro e o último vértice, que é a saída, está bloqueada pelo alien, assim se Ripley estivesse nessa posição teria que andar mais até encontrar a saída.

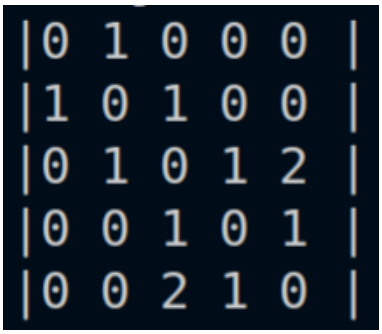
Assim como na criação da nave e de seus caminhos, o alien aparece de forma randomizada, podendo interromper Ripley ou não.

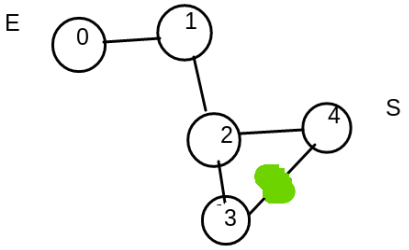
Busca utilizada:

Como Ripley necessita mover entre os vértices até achar a saída, é necessário a implementação de um algoritmo de busca. Nesse algoritmo, Ripley possui preferência pelo vértice de maior número que esteja conectado à sua localização atual e que o alien não a impeça. Assim, repete esse procedimento enquanto ela não acha a saída e quando não há mais grafos conectados, Ripley volta, movendo para os outros grafos que ainda não visitou, pois possuíam valores menores.

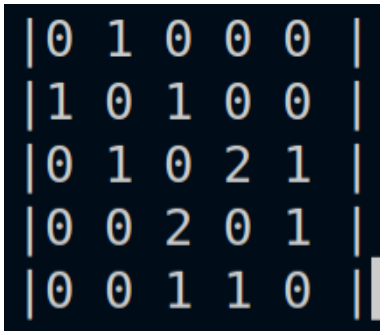
Quando Ripley está sendo interrompida pelo Alien e já visitou os outros vértices, a personagem deverá voltar, mantendo sua preferência pelos vértices de maior índice e que ainda não visitou, quando possível.

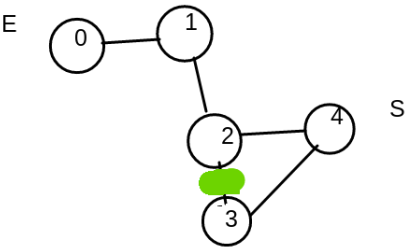
Quando essa situação ocorre e Ripley não pode voltar, um vértice com apenas 1 aresta a qual o alien ocupa, ela deve esperar.



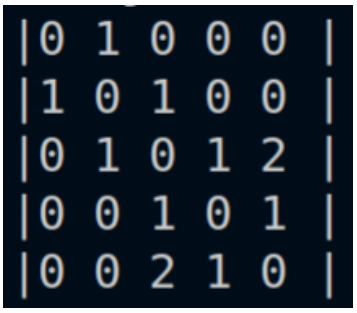


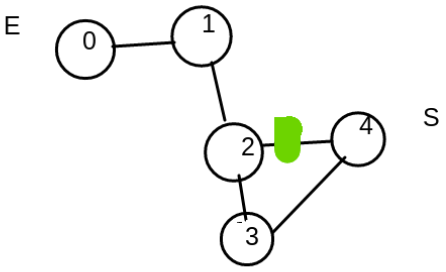
Ripley Começa em 0, então seguindo o algoritmo de busca, como só está ligado a 1, Ripley vai para a sala 1.



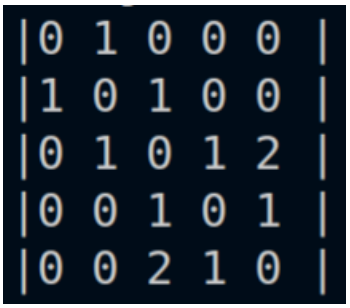


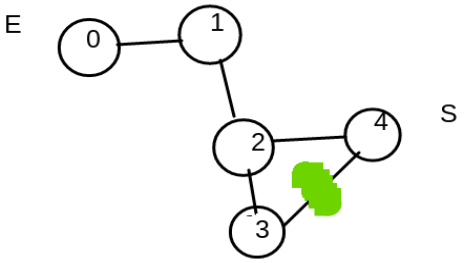
Como ainda não está sendo atrapalhada, e já visitou a sala 0, Ripley move para o vértice 2.



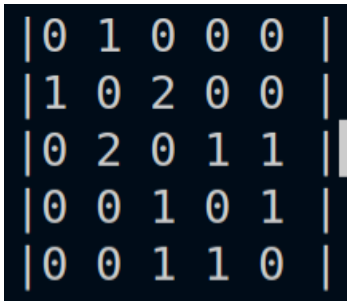


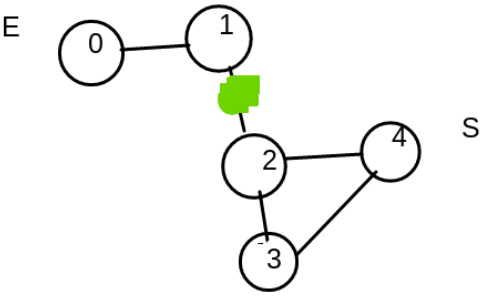
Ripley poderia ir para 4, onde está a saída, porém o Alien a impede, assim se move para 3.





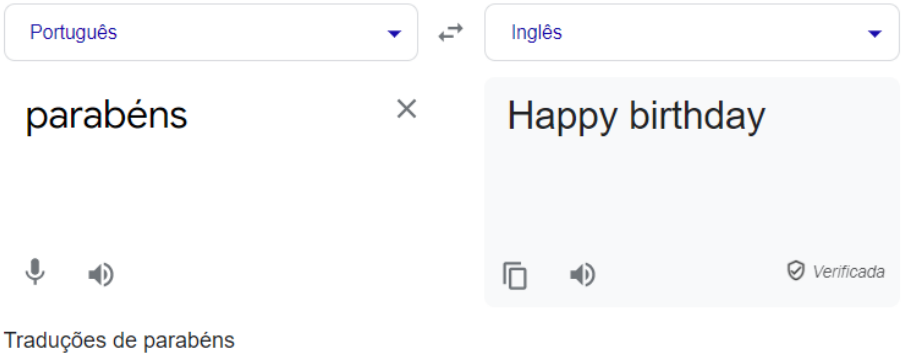
Novamente o alien está em seu caminho, mesmo que já tenha passado em 2, Ripley deve voltar, procurando uma outra forma de chegar em 4.





Finalmente, Ripley está em 2 e pode se mover para 4, achando a saída. Mesmo que o Alien esteja entre 1 e 2, isso não é problema, Ripley já visitou esse vértice e ela já encontrou a saída, que é sua prioridade.

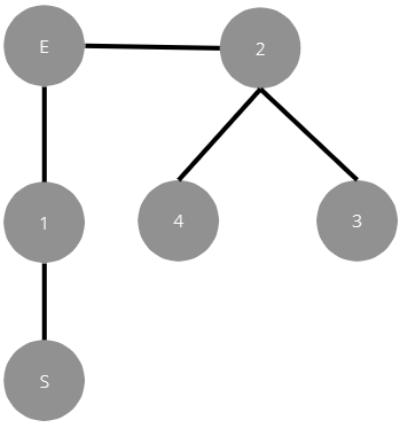
O programa irá exibir a mensagem “Happy birthday” (que significa parabéns em inglês):



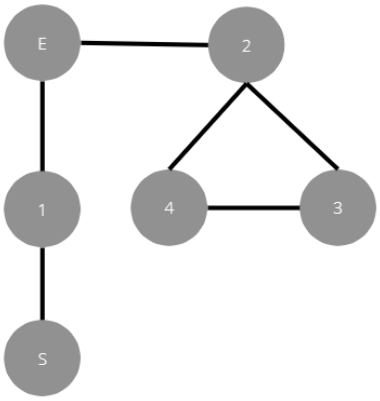
Complexidade do problema: Por Smolinski

1. Quantas arestas compõem o pior caso

Primeiramente é necessário entender que quanto menos arestas tiver, pior fica o problema (E = Entrada e S = Saída)
Segue o exemplo:



Nesse caso a Ripley moveria: E->2->4->2->3->2->E->1->S, totalizando 9 estados.



Já neste, com uma aresta a mais seus movimentos seriam: E->2->4->3->2->E->1->S, 8 estados, menos que o anterior.
Quanto mais arestas com a mesma quantidade de vértices, menor a quantidade de fins de linha no grafo.

2. Cadeia principal do pior caso

Ripley, por conta de seu comportamento, sempre vai seguir o vértice maior, já que a saída é sempre o último, além disso, a saída nunca está ligada diretamente na entrada. Dito isso, um bloco que será visto comumente no pior caso é:

Essa sequência será comum pois, a partir da entrada, o último vértice que Ripley vai visitar é o 1 e este estar ligado na saída coloca ela obrigatoriamente como a última sala visitada. Todos os piores casos vão conter essa estrutura base e o resto de vértices ligados à entrada ou arranjados em cadeias adjacentes a ela.

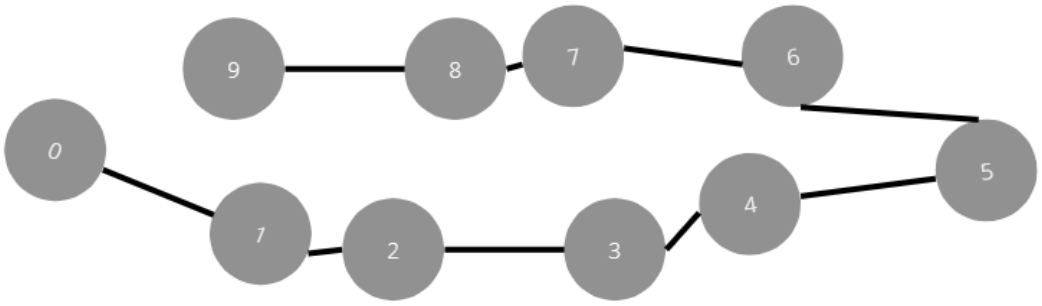
Desconsiderando o Alien, temos em um labirinto tamanho 3 o pior caso com 3 estados, E -> 1 -> S.

3. Cadeias adjacentes

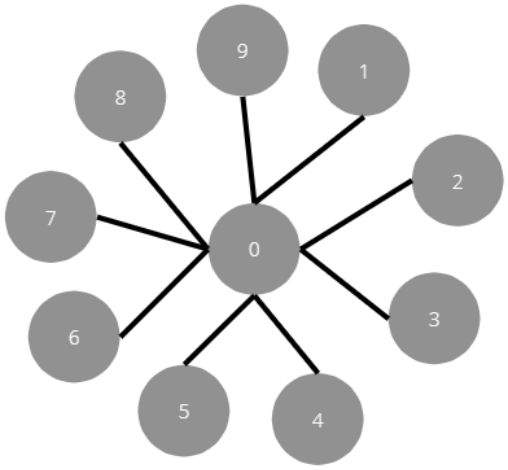
Vamos analisar uma sequência de casos

Em um grafo com 1 vértice, temos no pior caso 1 estado, já que é o único possível.
Em um grafo com 2 vértices, temos no pior caso 3 estados, 0->1->0.

Em um grafo com 10 vértices (com o mínimo de arestas possível) temos no pior caso 19 estados. Por exemplo: 0->1->2->3->4->5->6->7->8->9->8->7->6->5->4->3->2->1->0



Ou: 0->9->0->8->0->7->0->6->0->5->0->4->0->3->0->2->0->1->0



Não coincidentemente, a partir do 1, cada novo vértice adicionou uma aresta (pouco importando a posição), totalizando a soma dos vértices e arestas o total do pior caso. Essa fórmula é comum em Busca em Profundidade O(|V| + |E|), onde V são os vértices e E as arestas. Juntando tudo visto até o momento chegamos a uma fórmula:

O(N + N-1)

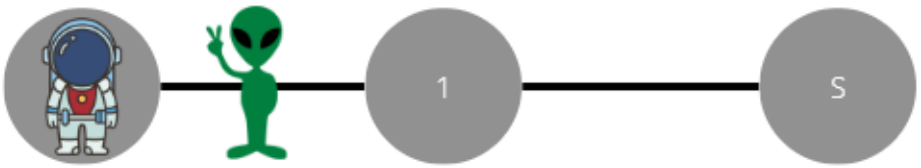
Nesta fórmula, N é o tamanho do labirinto (Número de vértices), a quantidade de arestas é substituída por 'N-1' pois é a menor quantidade possível. Essa fórmula ainda não está completa, pois já que temos a saída no fim de E->1->S, é possível notar que os passos S->1 e 1->E não serão feitos, já que Ripley já saiu da nave. Ou seja:

O(N + N-3) = O(2N - 3)

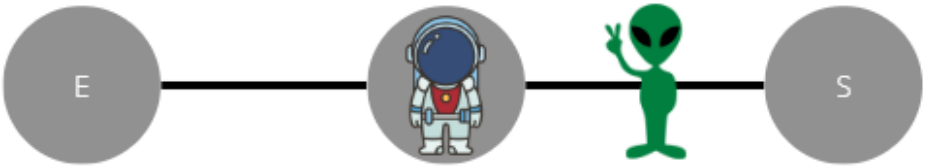
Esta é a fórmula final quando não tratamos do Alien.

4. Alien

A cada passo de Ripley, o Alien surge em alguma aresta aleatória do labirinto. De forma não surpreendente, existem locais em que ele surge que atrapalham mais do que outros, já que sempre que possível, Ripley vai se mover. Exemplo:



Neste caso, Ripley ficaria no vértice E esperando, já que não há para onde se mover, isso acrescentaria 1 estado a mais no pior caso (E->E), por outro lado:



Caso isso acontecesse, Ripley voltaria ao vértice E, já que pode se mover. Esse movimento acrescentaria 2 estados a mais no pior caso (1->E e E->1). Sabendo disso e dado que a estrutura retratada acima estará presente no pior caso, cada pior "trabalhada" do Alien vai causar 2 movimentos extras. Sendo assim, a equação com a inclusão do alien seria:

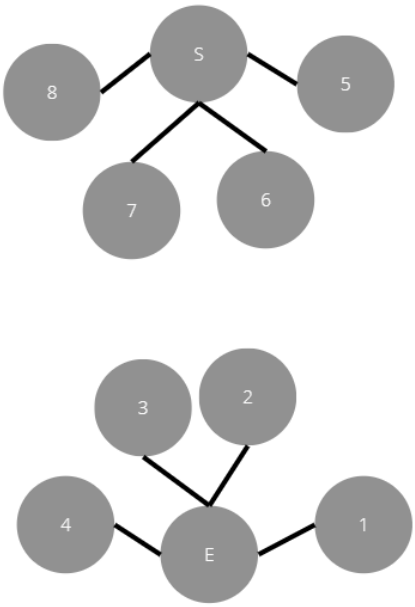
O(2N + 2A - 3)

Importante ressaltar que a variável A são as aparições efetivas do Alien.

5. Caso impossível

Existem principalmente dois casos impossíveis de serem solucionados no programa:

- **Alien infinito;** Como se trata do pior caso, devemos levar em conta a possibilidade (infinitamente pequena) do Alien aparecer de forma infinita na posição '1->S' do exemplo retratado acima. A cada vez que Ripley fosse passar pela posição ele estaria lá, isso colocaria a variável Alien como infinita, ocasionando um não-fim no algoritmo. A chance disso acontecer é nula pois o programa teria que rodar pela eternidade para confirmar que o inimigo sempre apareceria ali, o que não é possível. Por ser nula, ou pelo menos muito pequena, não se procurou soluções para tal.
- **Problema dos Cinco;** Caso fossem geradas 2 grafos separados na aleatoriedade do problema:



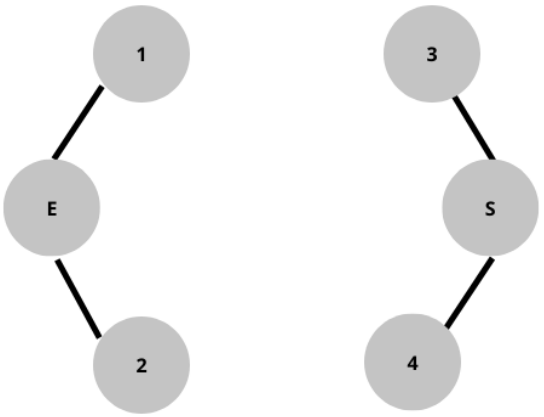
Neste caso, Ripley caminharia pelos 5 vértices adjacentes à entrada mas acabaria não encontrando a saída, fazendo do problema impossível. Para evitar isso, foi implementado que caso todos os vértices que Ripley visitou estivessem pintados de preto o código acabaria, pois não há solução para o problema.

Dificuldades Encontradas: Por Izabela

1. **Criação da nave**

Toda vez em que iniciamos o programa, uma “nave” é gerada de maneira aleatória, por meio da função “aleatorizaLabirinto()”, explicada acima. A nave é criada de forma que cada vértice esteja conectado a pelo menos algum outro vértice, além disso, impede que a saída e a entrada da nave estejam diretamente conectadas.

No entanto, ao criar a matriz de adjacências, existe uma chance mínima de que a nave esteja “quebrada”, assim, apesar de cada vértice possuir pelo menos uma aresta, uma parte da nave não terá conexão com o resto, como vemos na imagem abaixo:

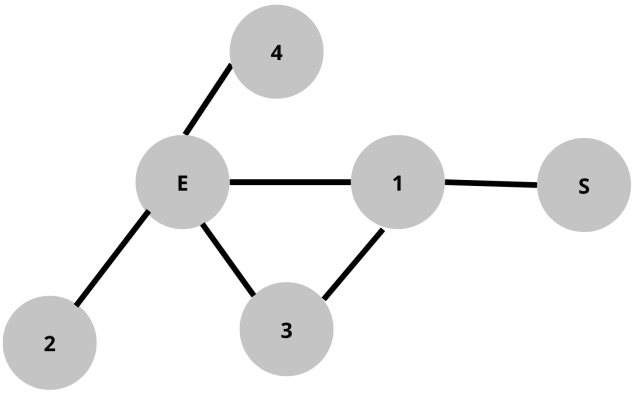


Para contornar esse problema, decidimos que o programa iria exibir a mensagem “Happy Birthday”, caso Ripley tenha visitado todos os vértices que são fisicamente possíveis para ela. Por exemplo, nesse caso, Ripley marcaria os vértices 1 e 2 como “cinza”, e então visitaria o 2, marcando-o como “preto”, voltaria para a entrada e visitaria o vértice 1, que agora também será marcado como “preto”. Sem mais vértices cinza para visitar, a mensagem “Happy birthday” é exibida.

2. Algoritmo de busca

Foi necessário adaptar o algoritmo de busca para que Ripley pudesse realizar um caminho fisicamente possível, de forma que ela não se teleporte pela nave. Após inúmeras tentativas falhas e diversos momentos em que Ripley entrava em um loop infinito procurando pela saída, chegamos a uma conclusão.

Criamos a função RipleyAnda(), que funciona da seguinte forma: Ripley irá marcar todos os vértices adjacentes ao vértice onde ela se encontra como “cinza” e os vértices que ela já visitou como “preto”. Todos os outros permanecem “branco”. Então, adicionamos os vértices cinza em um vetor, que funciona como se fosse uma pilha, embora não seja. Ripley irá visitar, caso possível, o último vértice adicionado na pilha. Por exemplo:



Nesse caso, Ripley iria marcar os vértices 1, 2, 3 e 4 como “cinza” e os colocaria no vetor, nessa ordem. Então ela visita o 4, o marca como “preto”, o retira da “pilha” e, sem outras opções, retorna para a entrada. Ripley visita o vértice 3 e descobre que ele possui uma conexão com o vértice 1, então, ignorando a ordem da “pilha”, Ripley dá prioridade à sala 1 e a remove do vetor. Ela encontra a saída e o programa exibe “Happy birthday”.

3. Matriz X Lista

Uma das maiores dificuldades encontradas foi a tomada de decisão ao escolher entre matriz ou lista de adjacências. Escrevemos dois códigos para criar a nave, um feito utilizando matriz de adjacências e o outro utilizando lista. 2/4 do nosso grupo apoiavam o time ”matriz” e os outros 2/4 apoiavam o time “lista”. Após muito tempo pensando, decidimos utilizar uma matriz de adjacências, por ser mais simples do que a lista.

Fim!!

Agradecimentos:

Agradecemos.