Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
SCC0503 - Algoritmos e Estruturas de Dados II

Relatório Exercício 06

Dungeon, triangulação de Delaunay,DFS, etc...

Aluno: Cauê Pereira Cermak 8936864
Professor: Leonardo Tórtoro Pereira

Julho
2022

Conteúdo

1 Introdução **1**

2 Desenvolvimento do Relatório **2**
2.1 Ordem de desenvolvimento 2
2.2 Estudando os Conceitos-chave de POO 2
2.3 Entendendo os Conceitos aplicados ao Java 2
2.4 Modificadores de classes, variáveis ou métodos 3

3 "Partindo"para os códigos **4**
3.1 Representação Visual do Código 5
3.2 Breve descrição das principais partes do código 5

4 Conclusão **8**

1 Introdução

1. Proposta Á proposta desse relatório é mostrar interesse em um mínimo entendimento em POO, uma vez que todos os exercícios anteriores foram desenvolvidos por mim em linguagem C. Para cumprir tal objetivo foi passada a tarefa de entendimento dos algorítmos criados em sala de aula pelo professor Leonardo Tórtoro Pereira[8], nos quais são usados para desenvolver a proposta abaixo.

Em resumo temos que criar um nível de jogo através do algoritmo de geração aleatória de Dungeons e Triangulação de Delaunay; criar outro grafo a partir da árvore geradora mínima e marcar algumas salas de interesse (como entrada, saída, checkpoint; etc...)

2 Desenvolvimento do Relatório

2.1 Ordem de desenvolvimento

1. Entendimento dos conceitos principais de POO[9]
2. Entendimento das sintaxes dos conceitos na linguagem Java
3. Entendimento do código[8]
4. Fazer relatório no Latex[3][1][2][12]

Como estava a muito tempo sem envolvimento com a linguagem de POO foi um pouco difícil entender novamente essa "forma de progamar", havia esquecido praticamente tudo.

Comecei pelos conceitos principais de POO, tais como herança, encapsulamento, abstração, fui entendendo novamente o que era uma classe, um objeto - uma instância de uma classe; métodos - comportamentos-; construtores de um objeto, que é um método especial, pois inicializa seus atributos toda vez que é instanciado (inicializado).

2.2 Estudando os Conceitos-chave de POO

Pelo que entendi Abstração é a capacidade de abstrair uma ideia e criar subdivisões. Encapsulamento, capacidade de encapsular os dados de uma aplicação, que significa evitar que estes sofram acessos indevidos. Herança, filho que herda as características do pai, ou seja, um filho tem todas qualidades de pai e também pode chamar seus métodos.

2.3 Entendendo os Conceitos aplicados ao Java

As variáveis têm determinada visibilidade(public, private, protected).[6] **Private:** A única classe que tem acesso ao atributo é a própria classe que o define, ou seja, se uma classe Pessoa declara um atributo privado chamado nome, somente a classe Pessoa terá acesso a ele. **Default:** Tem acesso a um atributo default (identificado pela ausência de modificadores) todas as classes que estiverem no mesmo pacote que a classe que possui o atributo. **Protected:** Esse é o que pega mais gente, ele é praticamente igual ao default, com a diferença de que se uma classe (mesmo que esteja fora do pacote) estende da classe com o atributo protected, ela terá acesso a ele. Então o acesso é por pacote e por herança. **Public:** Todos têm acesso.

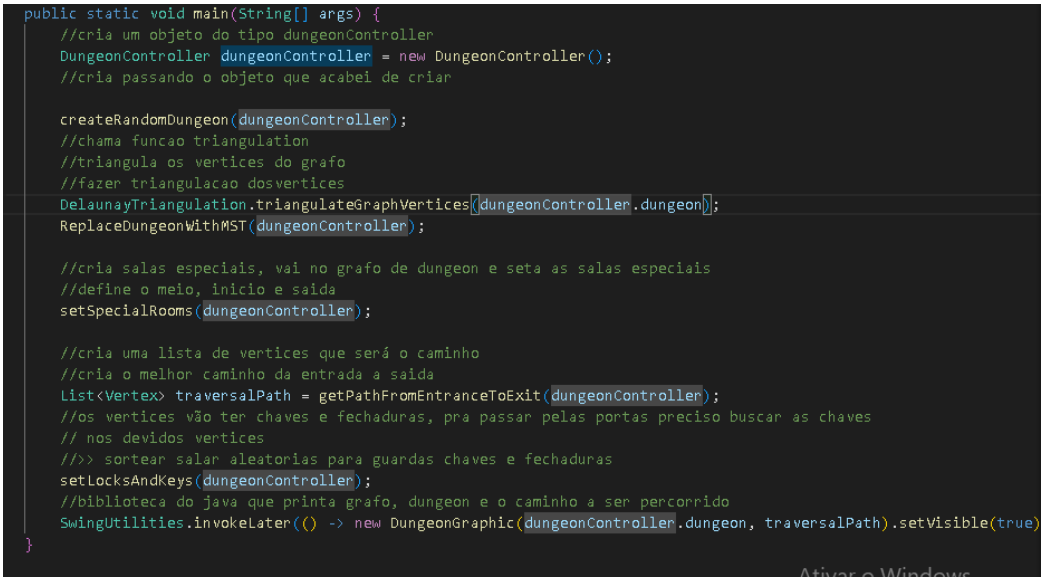
2.4 Modificadores de classes, variáveis ou métodos

abstract: classe que não pode ser instanciada ou método que precisa ser implementado por uma subclasse não abstrata.[10]
class: especifica uma classe.
extends: indica a superclasse que a subclasse está estendendo.
final: impossibilita que uma classe seja estendida, que um método seja sobrescrito ou que uma variável seja reinicializada.
implements: indica as interfaces que uma classe irá implementar interface: especifica uma interface.
native: indica que um método está escrito em uma linguagem dependente de plataforma, como o C.
new: instancia um novo objeto, chamando seu construtor.
static: faz um método ou variável pertencer à classe ao invés de às instâncias.
strictfp: usado em frente a um método ou classe para indicar que os números de ponto flutuante seguirão as regras de ponto flutuante em todas as expressões.
synchronized:indica que um método só pode ser acessado por uma thread de cada vez.
transient: impede a serialização de campos.
volatile:indica que uma variável pode ser alterada durante o uso de threads.

Tendo conhecimento destas coisas já foi possível, partir para o código e começar a entender, relacionando sempre com meus conhecimentos em linguagem C, nos quais auxiliaram bastante.

3 ”Partindo”para os códigos

A classe central desse código para resolução do exercício 6 é Dungeon-Controller, na qual possui o seguinte método main:



```
public static void main(String[] args) {
    //cria um objeto do tipo dungeonController
    DungeonController dungeonController = new DungeonController();
    //cria passando o objeto que acabei de criar

    createRandomDungeon(dungeonController);
    //chama funcao triangulation
    //triangula os vertices do grafo
    //fazer triangulacao dosvertices
    DelaunayTriangulation.triangulateGraphVertices(dungeonController.dungeon);
    ReplaceDungeonWithMST(dungeonController);

    //cria salas especiais, vai no grafo de dungeon e seta as salas especiais
    //define o meio, inicio e saida
    setSpecialRooms(dungeonController);

    //cria uma lista de vertices que será o caminho
    //cria o melhor caminho da entrada a saida
    List<Vertex> traversalPath = getPathFromEntranceToExit(dungeonController);
    //os vertices vão ter chaves e fechaduras, pra passar pelas portas preciso buscar as chaves
    // nos devidos vertices
    //>> sortear salar aleatorias para guardas chaves e fechaduras
    setLocksAndKeys(dungeonController);
    //biblioteca do java que printa grafo, dungeon e o caminho a ser percorrido
    SwingUtilities.invokeLater(() -> new DungeonGraphic(dungeonController.dungeon, traversalPath).setVisible(true));
}
```

Figura 1: Representação visual do código

3.1 Representação Visual do Código

Podemos observar abaixo que este é um código com uma complexidade estrutural alta em relação aos exercícios anteriores; utilizando diversos algorítmos, tais como floydWarshall; DelaunayTriangulation; DungeounGraphic; TraversalStrategy; keyGenerator; entre outros. Foi feita uma representação visual com as funções principais para se ter uma melhor percepção visual do que foi dito.

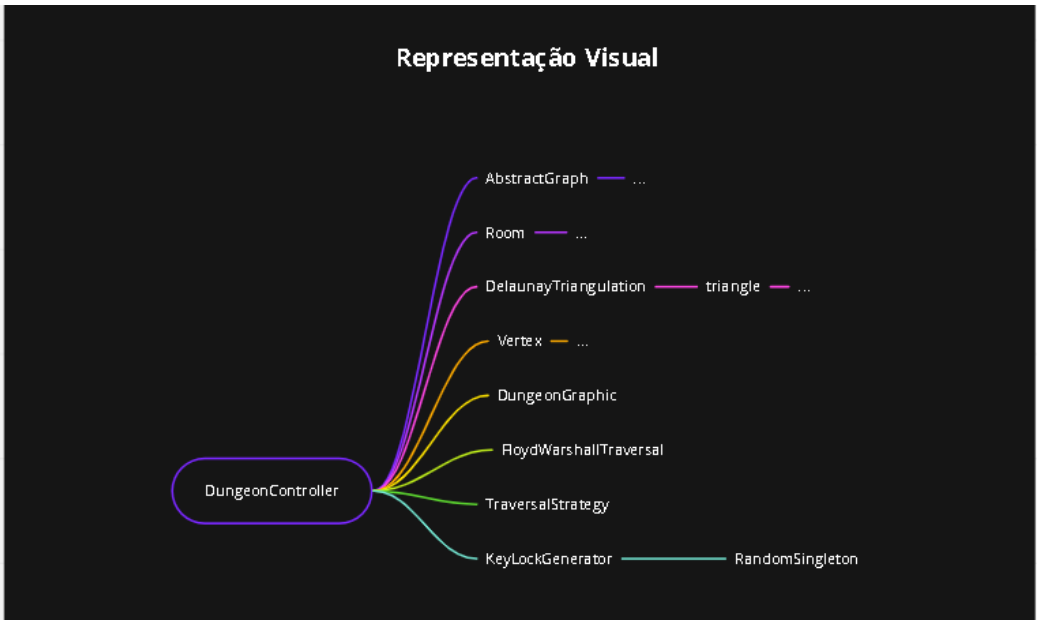


Figura 2: Representação visual do código

3.2 Breve descrição das principais partes do código

Primeiro cria-se um objeto do tipo dungeonController, cria-se os dungeouns a partir de uma "seed"e um o número de "Rooms"inseridas pelo usuário, representadas por um Grafo. Após a inserção destes valores, criam-se retângulo que representaram visualmente os vértices(Rooms).

Após a criação desse Grafo, chama-se "DelaunayTriangulation", como o próprio nome diz, triângula o Grafo, ou seja, faz triangulações, cria-se uma malha de triângulos entre os vértices "aleatórios"do Grafo, nesse algorítimo um triângulo não contém nenhum vértice "dentro"dele.[11]

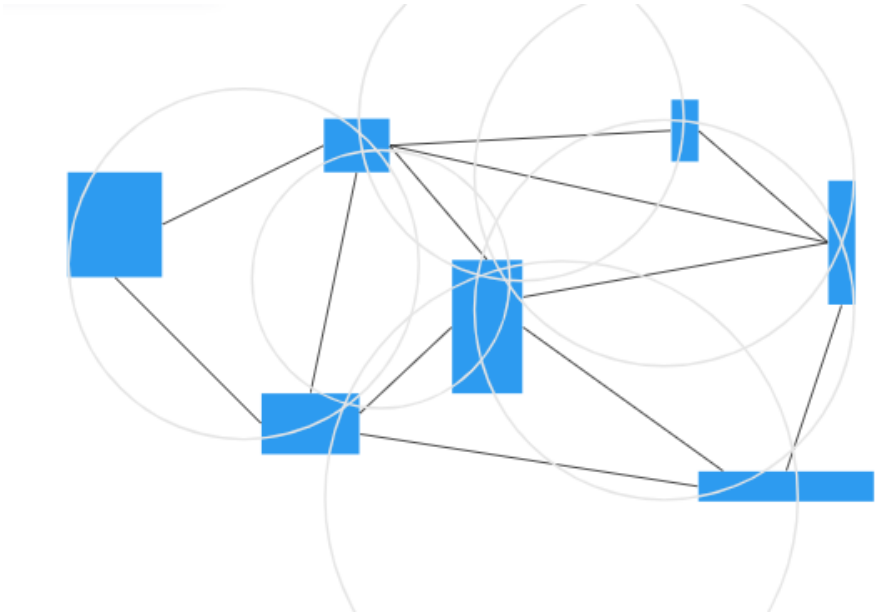


Figura 3: Representação Visual da triangulação de Delaunay

Após a triangulação chama-se o algoritimo MST[4] - arvores geradoras de custo mínimo -, ou seja, encontra-se os menores caminhos entre os vértices, através do algorítmo Prim, com a menor soma do peso das arestas e que contém todos os nós do grafo original. Portanto se o grafo tiver arestas redundantes, elas serão tiradas de modo a obter a soma mínima dos pesos.[5]

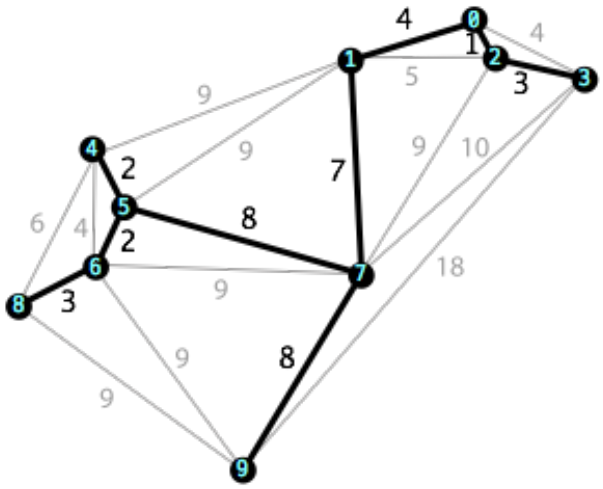


Figura 4: Representação Visual do Algorítmo MST - PRIM

Agora, pega-se o grafo com os menores caminhos e "seta"as salas especiais, o vértice central(aquele com a maior das distâncias entre todos os outros vértices), o menos central(aquele com a maior das distâncias entre todos os outros vértices) e o vértice - sala - final(o vértice mais distante deste "menos central"e marcá-lo como a sala final).

Para isso usa-se o algorítmo de FloydWarshall, desenvolvido em projetos anteriores. Algorítmo este que encontra o menor caminho, de custo mínimo, entre todos os vértices de um grafo com peso nas suas arestas.

Cria-se uma lista de vértices com menor caminho entre a entrada e saída.

Utiliza-se uma busca em largura, também desenvolvida em projetos anteriores, a partir do vértice inicial e, com a mesma semente para o gerador de números aleatórios, é feito o sorteio de salas para guardarem chaves e fechaduras (usar, ou tomar como base, o algoritmo desenvolvido no repositório)[7]

Em seguida é mostrado visualmente o resultado do que foi feito até então.

4 Conclusão

Ao contrário dos outros projetos, este eu não desenvolvi, apenas investiguei e tentei entender o que estava desenvolvido nos códigos do repositório do GitHub do Professor Leonardo, entretanto foi bem interessante entrar em contato com a linguagem de programação Java, que há muitas similaridades com C, mas uma estrutura de código bem diferente. Para desenvolver em POO é nítido que precisa-se de uma estrutura e construção de pensamento para resolver problemas. É bem interessante como é fácil reutilizar métodos e atributos de determinada classe. Confesso que fiquei um pouco perdido no meio de tantos códigos e tive que abstrair determinadas partes pra conseguir entender melhor o todo.

Agradeço pelo semestre, quando comecei a matéria achei que não ia conseguir terminar, muitas vezes me questionei, *"será que eu consigo programar essas coisas complexas?"*. Com o tempo, com os projetos, com sua ajuda, as coisas começaram ficar mais fáceis; me sinto mais confiante em relação a programação.

Obrigado por me instigar a fazer esse relatório e ir atrás de POO, com certeza vou estudar mais por conta.

Grande abraço e obrigado, de coraçao Léo! Convida a gente pra ver sua defesa!

Referências

[1] Colaborativo. Comandos básicos de latex. https://pt.wikibooks.org/wiki/Latex.

[2] Colaborativo. Comandos básicos de latex. https://pt.wikibooks.org/wiki/Latex.

[3] Andrade Doherty. Uma introdução ao latex. 2000.

[4] Paulo Feofiloff. Árvores geradoras de custo mínimo (mst). https://www.ime.usp.br/ pf/algoritmos$_p ara_g rafos/aulas/mst.html$.

[5] Sérgio Mucciaccia. Diferença na aplicação dos algoritmos de dijkstra e prim. https://pt.stackoverflow.com/questions/123522/diferen

[6] Stack Overflow. Diferenca entre modificadores. https://pt.stackoverflow.com/questions/23/qual-

[7] Leonardo Tórtoro Pereira. Credo dos frita sinos: Infinidade - o estágio - descrição do projeto 6. https://edisciplinas.usp.br/mod/assign/view.php?id=4282819.

[8] Leonardo Tórtoro Pereira. Grafos - busca em profundidade e implementação em hash. https://github.com/LeonardoTPereira/Graph2022/tree/v0.1.0-alpha.

[9] Leonardo Tórtoro Pereira. Grafos - busca em profundidade e implementação em hash. https://edisciplinas.usp.br/pluginfile.php/6932047/mod$_r esource/content/1/Resum$

[10] Vanessa Sabino. Palavra reservadas do java. http://www.linhadecodigo.com.br/artigo/83/as-52-palavras-reservadas-do-java.aspx.

[11] SCIco. Delaunay triangulation. https://www.youtube.com/watch?v=GctAunEuHt4.

[12] tex.stackecvhange. Comunidade latex online. https://tex.stackexchange.com/.