



INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO

Bacharelado em Ciências da Computação

Library Mapper

Thiago Gomes Toledo
Supervisor: Prof. Dr. Marcelo Finger

São Paulo - SP

Segundo semestre de 2011

Agradeço a minha mãe Mara Lucia Gomes Toledo e a meu pai João Batista Toledo por terem me dado a oportunidade de realizar meu sonho; A minha esposa Lucianna Thomaz Almeida Toledo por me dar conforto, ajuda e sábias dicas; A meu filho que me traz alegria só por imaginar seu rosto e a Deus por ter me dado todos e tudo que sempre precisei.

1

¹Para mais informações sobre essa monografia, apresentação, código, etc.. acesse o site: www.linux.ime.usp.br/~renoir/mac499/proposta.html

Sumário

1	Introdução	3
2	Parte Objetiva	3
2.1	Objetivo	3
2.2	Problema	3
2.3	Solução	3
2.4	Conceitos e tecnologias estudadas	4
2.4.1	O que são Qr-Codes?	4
2.4.2	Grid Graphs	5
2.4.3	Breadth-First Search (BFS)	6
2.4.4	Bi-directional BFS	7
2.4.5	MER - Modelo Entidade Relacionamento	9
2.4.6	PgAdmin -	9
2.4.7	POO - Programação Orientada a Objetos	9
2.4.8	Mybatis - Integração MER e POO	9
2.4.9	Servlet - WebService JAVA	9
2.4.10	Apache Tomcat - Servidor WEB	9
2.4.11	Métodos Ágeis - Kanban e Integração Contínua	9
2.4.12	HTML5 - Canvas	9
2.4.13	Desenvolvimento mobile	9
2.5	Library Mapper, o cliente	9
2.5.1	Construindo o seu mapa Web	9
2.5.2	Transformação do JavaScript para Nodes	9
2.5.3	Salvando no Banco de Dados	9
2.6	Library Mapper, o usuário	9
2.6.1	Identificando o usuário	9
2.6.2	A consulta e a ligação com a biblioteca	9
2.6.3	Lista de livros e a busca pela estante	9
2.6.4	Montando o mapa com HTML5 e canvas	9
2.7	Atividades realizadas	9
2.8	Resultados e produtos obtidos	9
2.9	Conclusões	9
3	Parte Subjetiva	9
3.1	Desafios e frustrações	9
3.2	Relações entre Disciplinas	9
3.3	Os próximos passos	9

1 Introdução

Como frequentador da biblioteca do Instituto de Matemática e Estatística da USP desde 2008 e trabalhando na biblioteca do Instituto de Geociências da USP em 2010-2012, pude notar que as reclamações numa biblioteca sobre a dificuldade em se achar um livro lideram o ranking de problemas apontados por visitantes que usualmente têm que apelar para a boa vontade dos funcionários, pois acham muito complicado ou até sem nenhum sentido o modo como as fileiras de livros são identificadas.

Em ambos os casos os alunos ainda eram capazes de encontrar seus livros ou teses dado que a maior delas possuía não mais que 20.000 exemplares. Contudo, numa biblioteca como a da Faculdade de Filosofia, Letras e Ciências Humanas que hoje possui um inventário de mais de 500 mil livros e ainda 600 mil teses¹, a busca se torna completamente inviável sem nenhum auxílio real.

2 Parte Objetiva

Nessa parte do texto, serão apresentados os conceitos teóricos do trabalho, além da parte prática de desenvolvimento.

2.1 Objetivo

Com uma ferramenta que mostre como ir do ponto onde está o visitante até o livro através de um mapa, esse dificilmente se sentiria perdido ou precisaria contar com o auxílio de terceiros. Portanto, o objetivo desse projeto Library Mapper é que o visitante busque, escolha o livro e receba no seu aparelho mobile um mapa com o caminho correto até esse livro.

2.2 Problema

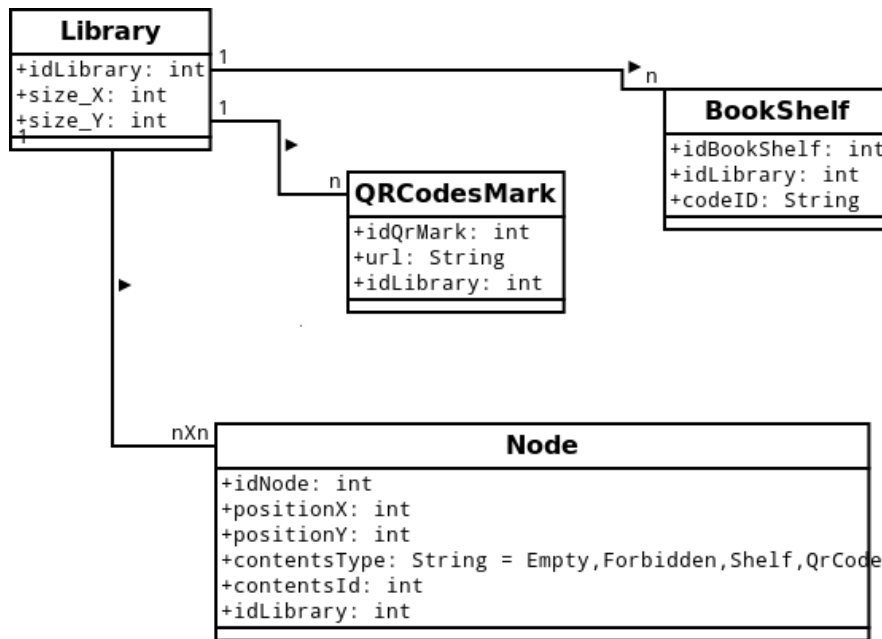
Como dispositivos GPS não mapeam dentro de residências, é inviável utilizar a Geolocalização do visitante para qualquer fim e portanto era necessário criar algum método para identificar onde ele estava localizado, como era esse lugar e como gerar uma busca entre onde ele estava e onde devia chegar.

2.3 Solução

Inicialmente desenvolvida para dispositivos IOS, essa ferramenta utiliza tags com códigos QR para auxiliar o reconhecimento da localização atual do visitante. O visitante ao entrar numa biblioteca procura essa tag e com seu dispositivo IOS, reconhece o código nela contido. Esse código abre um Web Browser no dispositivo e uma página da biblioteca na internet é mostrada na tela. O visitante digita o livro num campo de busca e uma vez que esse existe nessa biblioteca, a tela do web browser é substituída por uma tela com um mapa que mostra o caminho da tag selecionada até a fileira onde está o livro.

Para ser possível a localização do livro, será utilizada a prévia catalogação feita pela biblioteca da posição do mesmo e uma réplica proporcional do mapa real da biblioteca. Essa réplica é desenvolvida através de uma plataforma web que gera mapas 2D também desenvolvida nesse projeto. As informações obtidas nesse mapa, localização de estantes e suas correspondentes identificações, Qr-Codes, obstáculos e limites da biblioteca, serão enviados e armazenados num banco de dados da aplicação.

¹Dados levantados em janeiro 2012 com a diretora da biblioteca, Maria Aparecida Laet



O mapa será gerado num grafo tipo grid e cada nó está dividido em 4 tipos principais que são QR-Code, Forbidden, Free e Bookshelf numa classe Node e será baseado nesses tipos que a busca pelo livro dado será feita.

Como todas as arestas deste grafo tem peso único um, o algoritmo de busca usado é uma Bi-direcional BFS (Breadth-first search) com uma busca saindo da origem (QR-Code selecionado pelo visitante) e outra do destino (estante onde está o livro).

Essa busca é feita de uma maneira paralela onde cada thread é um dos pólos do caminho Bi-direcional.

2.4 Conceitos e tecnologias estudadas

2.4.1 O que são Qr-Codes?

Quick Response Code foi inventado por DENSO WAVE, uma empresa Japonesa de captura automática de dados, FA e campos de solução de negócios, em 1994 e foi um dos primeiros códigos de barra 2D a ser usado em aplicações onde era necessário uma combinação de camera de aparelhos celulares e códigos de barra 2D. [2]Esses códigos trocam barras e espaços por pontos e espaços arranjados num vetor ou uma matrix, aumentando consideravelmente a quantidade de dados armazenados num espaço igualmente usado para um código de barra 1D. Apenas em 2002 cameras de celulares foram comercialmente vendidas com dispositivos reconhecedores de Qr-codes e desde então uma vasta variedade de aplicativos começou a surgir no Japão e vem tomando o mundo.



.Em 2006, como parte do Windows Live service, a Microsoft lançou o Windows Live Barcode , onde Qr-codes eram usados como ponte para troca de informações entre aparelhos mobile e outras medias como PCs, Tv de plasma e revistas.

As Qr-Codes geradas para esse projeto passarão para o usuário o endereço de uma página web de busca dos livros da biblioteca em que ele se encontra.Junto nessa URL tem um id do Qr-Code, que mostra para o programa onde no grafo do mapa da biblioteca se encontra o visitante, para dessa maneira poder criar o caminho que o mesmo deve traçar.

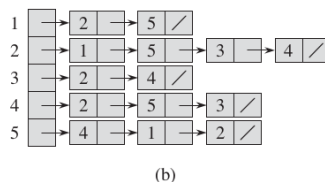
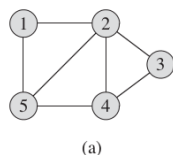
2.4.2 Grid Graphs

Um grafo G é uma par (V, E) , onde V é um conjunto finito e E é uma relação binária de V . O conjunto V é chamado de o *conjunto de vértices* de G e seus elementos são chamados de *vértices*. O conjunto E é chamado de *conjunto de arestas* de G e seus elementos são chamados de *arestas*. Existem dois tipos de grafos - grafo direcionados e grafos não-direcionados. [1]

Uma aresta num *grafo não-direcionado* é um conjunto $\{u, v\}$ onde $u, v \in V$ e $u \neq v$ que por convenção é definida como (u, v) ou (v, u) e é possível ir de u para v e de v para u .

Já num *grafo direcionado* (ou *dígrafo*) uma aresta é identificada por setas e $(u, v) \neq (v, u)$.

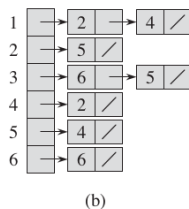
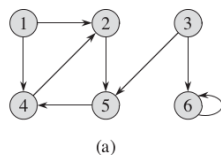
Grafo não-direcionado



(c)

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Grafo direcionado

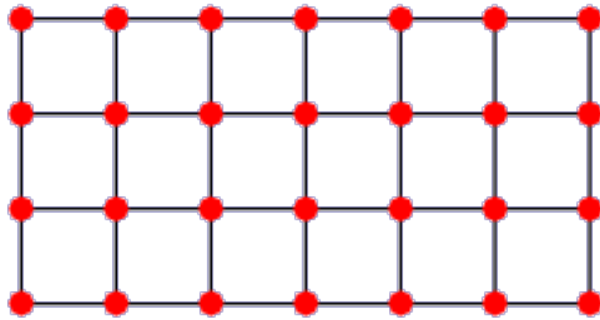


(c)

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Em ambos os casos uma aresta (u,v) pode possuir um *peso* que indica o custo por passar do vértice u para o v .

Um Grid Graph (grafo do tipo Grid) é um grafo não-direcionado; uma grade de nós (vértices) $n \times n$ sendo que cada nó tem exatamente 4 arestas com exceção dos nós das extremidades.



Para o LibraryMapper será usado um grafo não-direcionado do tipo Grid alocado numa matriz, sendo que cada vértice do grafo será um nó da classe Node. Essa Possui os seguintes atributos:

```
1 package library.domain;
2
3 public class Node {
4
5     private Integer idNode;
6     private Integer positionX;
7     private Integer positionY;
8     private Integer contentId;
9     private Integer idLibrary;
10    private String contentType;
11    private Node parentFromBeginNode;
12    private Node parentFromEndNode;
13    private int mapValue;
14
15    /*... continua ...*/
```

2.4.3 Breadth-First Search (BFS)

Dado um grafo $G=(V,E)$ com um vértice s distinto, Breadth-first search explora sistematicamente as arestas de G para descobrir todo vértice que pode ser alcançado por s . Ele computa a distância (menor número de arestas) do vértice s até todos os vértices que ele possa alcançar. Para manter o controle do progresso da busca, BFS procura cores em cada vértice - branco, cinza e preto. Todos os vértices começam com cor branca, eventualmente passam para cinza e por fim ficam pretos. [1]

Uma vez que um vértice é descoberto ele se torna não-branco. Um vértice só se tornará preto quando todos os vértices que são adjacentes a ele tenham sido descobertos, enquanto isso ele será cinza. Isso representa a fronteira entre vértices descobertos e não-descobertos.

Como no pior caso, a busca tem que ser feita em todos os m nós de um grafo e como a BFS passa não volta nos caminhos que já traçou, a complexidade desse algoritmo é $O(m)$. Além das cores, nesse projeto essa BFS distingue entre nós (vértices) do tipo Free, Bookshelf, Qr-Code e Forbidden e será aplicada de uma maneira Bi-direcional.

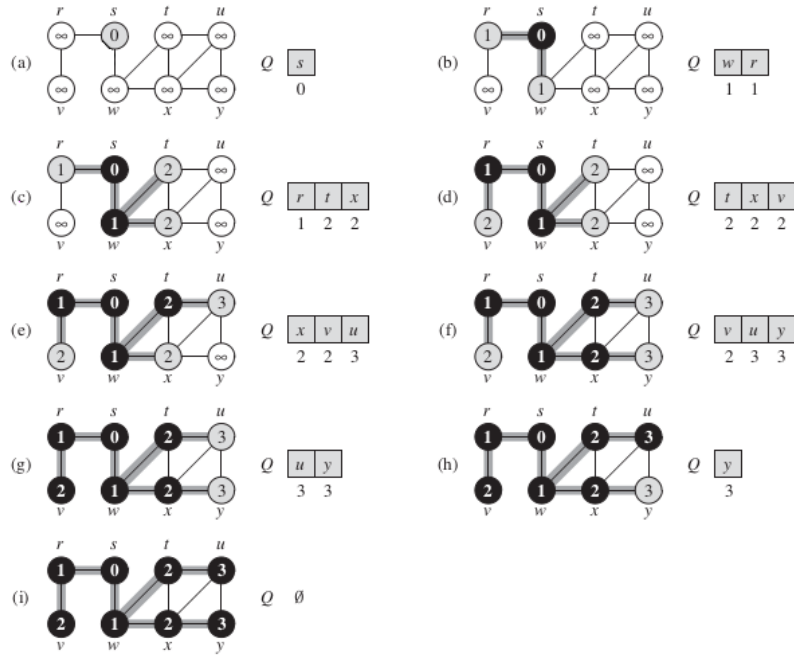
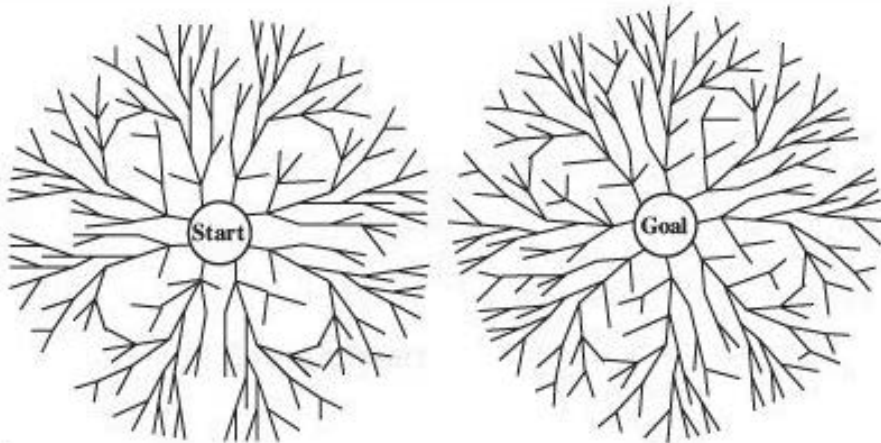


Figura 1: Close-up of a gull

2.4.4 Bi-directional BFS

A idéia por trás de uma busca Bi-direcional é fazer duas buscas rodarem simultaneamente - Uma "para frente", saindo do estado inicial (a origem) da busca inteira e outra "para trás", saindo do estado final (a chegada) da busca inteira esperando que ambas se encontrem no meio. [3].



Nesse projeto a busca Bi-direcional será realizada através de métodos concorrentes de programação. Cada pólo iniciará sua busca paralelamente liberando assim duas threads. Caso uma thread chegue em um nó que já foi visitado ela define aquele nó como o "meio do caminho" e a partir dele gera a rota inteira do mapa. Mas caso o nó que chegue esteja sendo visitado, ela para e abandona o que estiver fazendo para depois ser reciclada num outro novo processo.

Pela profundidade da busca ter caído pela metade, ou seja, no máximo uma busca tem que ir até a metade do grafo, a complexidade do algoritmo é $O(m/2)$.

2.4.5 MER - Modelo Entidade Relacionamento

2.4.6 PgAdmin -

2.4.7 POO - Programação Orientada a Objetos

2.4.8 Mybatis - Integração MER e POO

2.4.9 Servlet - WebService JAVA

2.4.10 Apache Tomcat - Servidor WEB

2.4.11 Métodos Ágeis - Kanban e Integração Contínua

2.4.12 HTML5 - Canvas

2.4.13 Desenvolvimento mobile

2.5 Library Mapper, o cliente

2.5.1 Construindo o seu mapa Web

2.5.2 Transformação do JavaScript para Nodes

2.5.3 Salvando no Banco de Dados

2.6 Library Mapper, o usuário

2.6.1 Identificando o usuário

2.6.2 A consulta e a ligação com a biblioteca

2.6.3 Lista de livros e a busca pela estante

2.6.4 Montando o mapa com HTML5 e canvas

2.7 Atividades realizadas

2.8 Resultados e produtos obtidos

2.9 Conclusões

3 Parte Subjetiva

3.1 Desafios e frustrações

3.2 Relações entre Disciplinas

3.3 Os próximos passos

Referências

- [1] H. Thomas Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press, 1 edition, 2009.
- [2] Hiroko Kato, Keng T. Tan, and Douglas Chai. *Barcodes for Mobile Devices*. Cambridge, 1 edition, 2010.
- [3] Stuart Russel and Peter Norvig. *Artificial Intelligence, A Modern Approach*. Prentice Hall, 3 edition, 2009.