

Faculdade de Engenharia da Universidade do Porto



RELATÓRIO

Ana Isabel Neves Alves de Sousa, 201108026, ei11068@fe.up.pt

Gabriel Cardoso Candal, 201108021, ei11066@fe.up.pt

Tiago Almeida Fernandes, 201106911, ei11054@fe.up.pt

Mestrado Integrado em Engenharia Informática e Computação

Conceção e Análise de Algoritmos

26 de Abril de 2013

Índice

1. INTRODUÇÃO	3
2. DESCRIÇÃO DO TEMA.....	4
3. SOLUÇÃO IMPLEMENTADA	5
3. ALGORITMOS IMPLEMENTADOS.....	8
4. CASOS DE UTILIZAÇÃO	10
5. DIAGRAMA UML.....	12
6. DIFICULDADES SENTIDAS	14
7. ESFORÇO DE CADA ELEMENTO	15

1. INTRODUÇÃO

No âmbito da unidade curricular de Conceção e Análise de Algoritmos foi-nos proposto o desenvolvimento de uma ferramenta que permitisse o planeamento do sistema interno de transporte de pessoas em *Shimizu Mega-City Pyramid*.

2. DESCRIÇÃO DO TEMA

O sistema interno de transporte de pessoas é baseado em passareiras rolantes rápidas, escadas rolantes e carruagens automatizadas.

A ferramenta criada possibilita o dimensionamento e a simulação da rede de transportes, permitindo calcular o fluxo de pessoas que podem passar em cada trecho da rede, sendo que cada trecho possui uma capacidade que está dependente do tipo de transporte instalado. Ao longo da rede existem pontos de agregação bem como pontos de dispersão que fazem variar o fluxo de pessoas.

Como dados de entrada a ferramenta recebe redes de diferentes dimensões, bem como os transportes utilizados em cada trecho e retorna o fluxo máximo de pessoas em cada trecho da rede.

3. SOLUÇÃO IMPLEMENTADA

O programa começa por mostrar ao utilizador o seguinte menu:

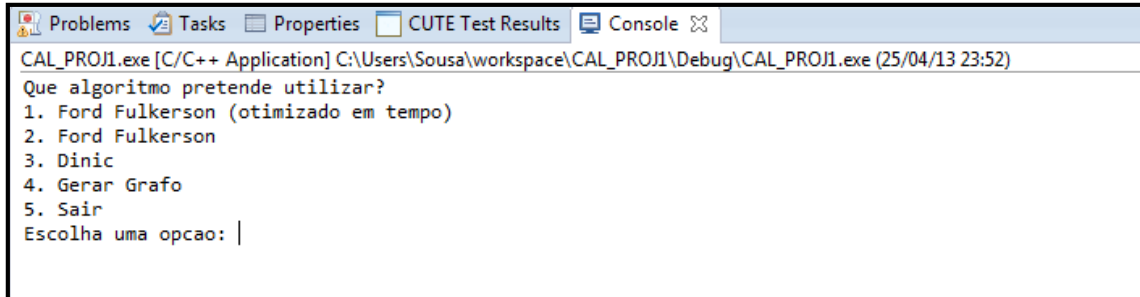


Figura 1: menu inicial

O ficheiro de leitura, e de posterior escrita, possui a informação sobre a rede sobre a qual irá ser efetuada o cálculo do fluxo deverá ter o seguinte formato:

- Primeira linha corresponde ao número de vértices, seguido do número de arestas;
- Restantes linhas correspondem às arestas existentes, sendo indicado, para cada uma, a sua fonte, o seu destino e a sua capacidade.

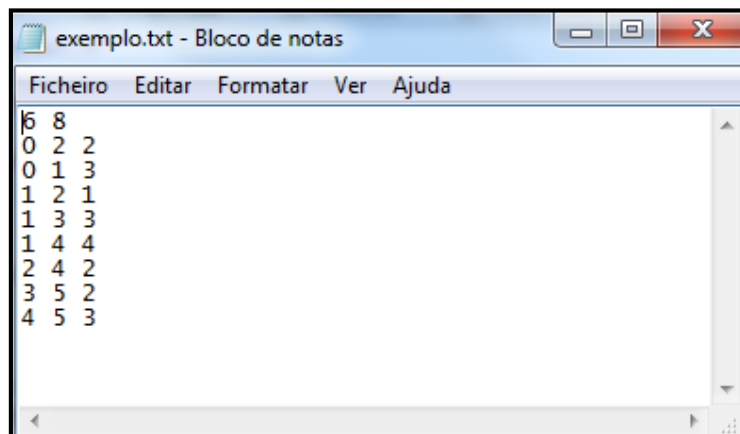


Figura 2: exemplo do ficheiro de leitura

Sempre que for solicitada a visualização do grafo correspondente, o utilizador irá visualizar algo semelhante ao seguinte exemplo, em que os vértices vermelhos são potenciais *bottlenecks* (a divisão da capacidade que sai do vértice pela que entra é menor que 0.7), as arestas a vermelho correspondem a arestas que transportam a capacidade máxima e a grossura de cada aresta depende do fluxo que passa.

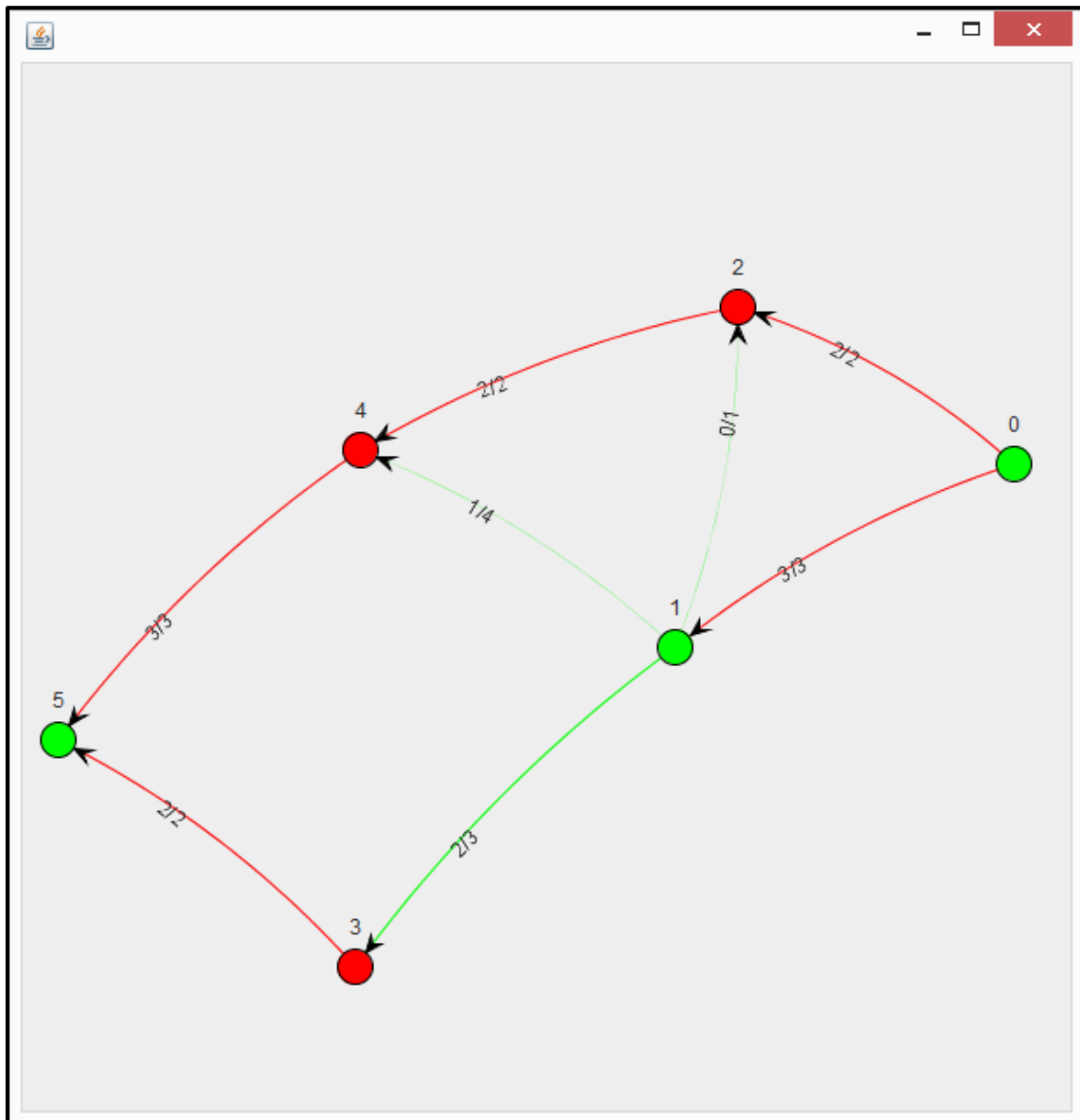


Figura 3: exemplo de grafo de fluxo

Na primeira opção do menu, além do grafo referido anteriormente, é também apresentado o grafo residual final, como se mostra no exemplo seguinte.

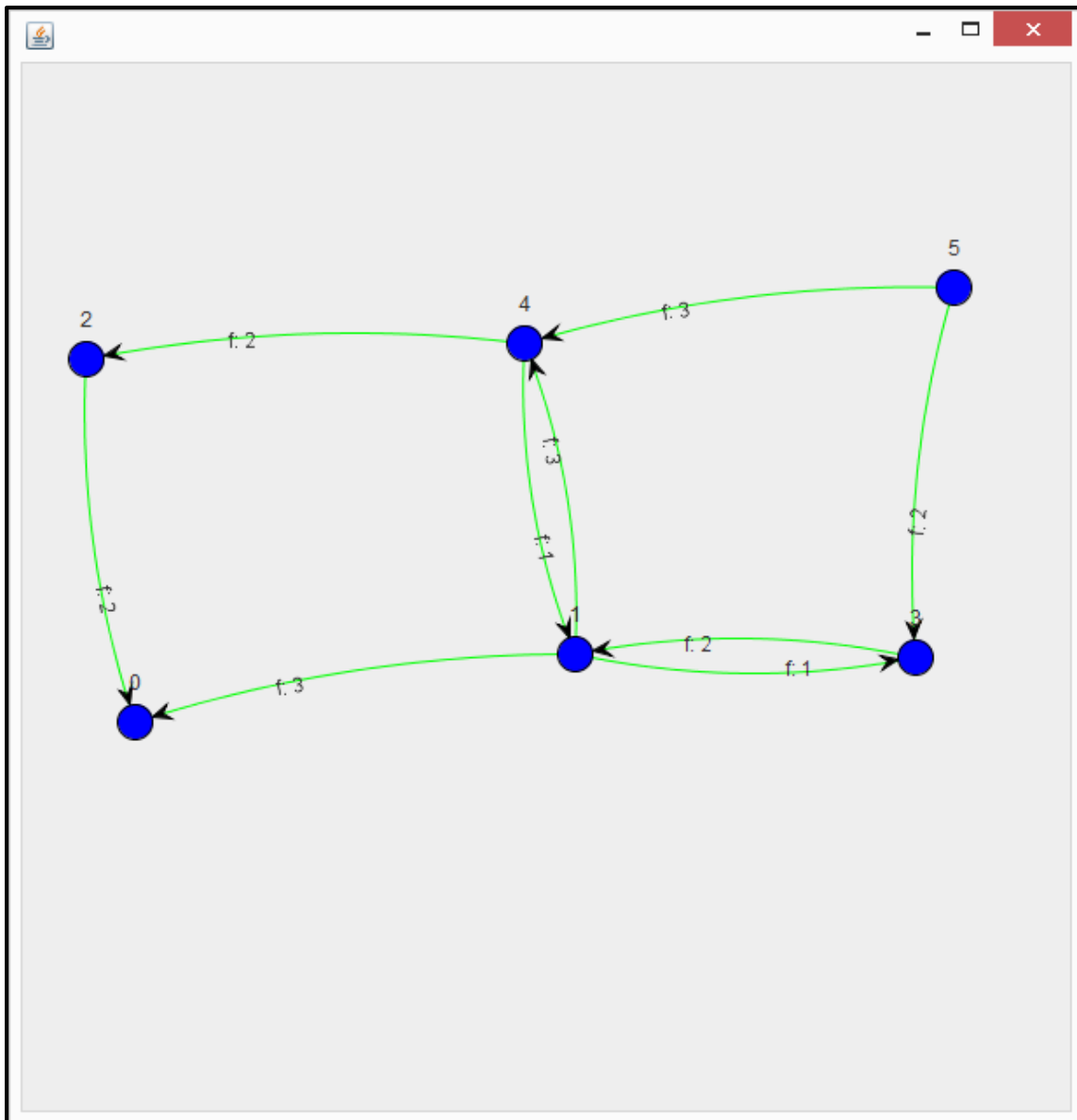


Figura 4: exemplo de grafo residual

3. ALGORITMOS IMPLEMENTADOS

Foram utilizados dois algoritmos para a resolução do problema de fluxo máximo: *Ford-Fulkerson* e *Dinic*.

O algoritmo de *Ford-Fulkerson* (variante de *Edmonds-Karp*) baseia-se no aumento de um valor de fluxo em cada etapa de cálculo. Inicialmente o fluxo de cada aresta é igual a zero. Em cada etapa calcula-se um caminho de aumento da fonte ao destino do grafo sendo que ao fluxo total se adiciona o fluxo máximo desse caminho. O algoritmo termina quando não é possível encontrar mais nenhum caminho de aumento.

O algoritmo de *Dinic* tem um funcionamento semelhante ao de *Ford-Fulkerson*, com a adição do conceito de nível de nó (que permite escolher de forma mais eficiente o caminho de aumento, que será um caminho de comprimento mínimo), que representa a distância de um nó à fonte.

Em relação à complexidade, o tempo de execução do algoritmo de *Ford-Fulkerson* é igual a $|E| * |V| * U$ (em que E corresponde ao número de arestas; V ao número de vértices e U à capacidade máxima) e o do algoritmo de *Dinic* é igual a $|E| * |V|^2$.

Após a análise dos tempos obtidos com a implementação que fizemos sobre a estrutura usada nas aulas teórico-práticas, achámos que seria possível obter melhores tempos facilitando a acessibilidade aos valores das arestas, então optámos por fazer uma implementação que usasse uma matriz de adjacências, que apesar de resultar em espaço desperdiçado, possibilita obter resultados bastante melhores.

Os gráficos a seguir representados foram feitos com base em grafos cuja capacidade máxima por aresta é 30, daí que, apesar de na generalidade dos casos ser espectável que o algoritmo de *Dinic* tenha melhor desempenho, tal não se verificou nos testes que efetuámos (como seria de esperar, dado que $|V| > U$).

Por outro lado, o uso da estrutura de dados que usámos não foi o melhor e necessita de realizar trabalho demorado para efetuar operações sobre arestas, e como

o algoritmo de *Dinic* o faz mais vezes (tem que esconder arestas, além da normal adição de fluxo) sai prejudicado; este fator seria atenuado se a função que devolve arestas o fizesse por referência, poupando assim tantas chamadas que exigem procura vértice-a-vértice.

Arestas	Nós	Tempo Ford Fulkerson	Tempo Dinic	Tempo Ford-Fulkerson (otimizado tempo)
25	13	0,001	0,002	0
1225	50	0.13	0,268	0,001
4950	100	1,677	3,476	0,012
11175	150	7,358	15,346	0,034
19900	200	22,506	46,671	0,077

Tabela 1: tempos de execução para cada algoritmo em função do número de arestas e nós

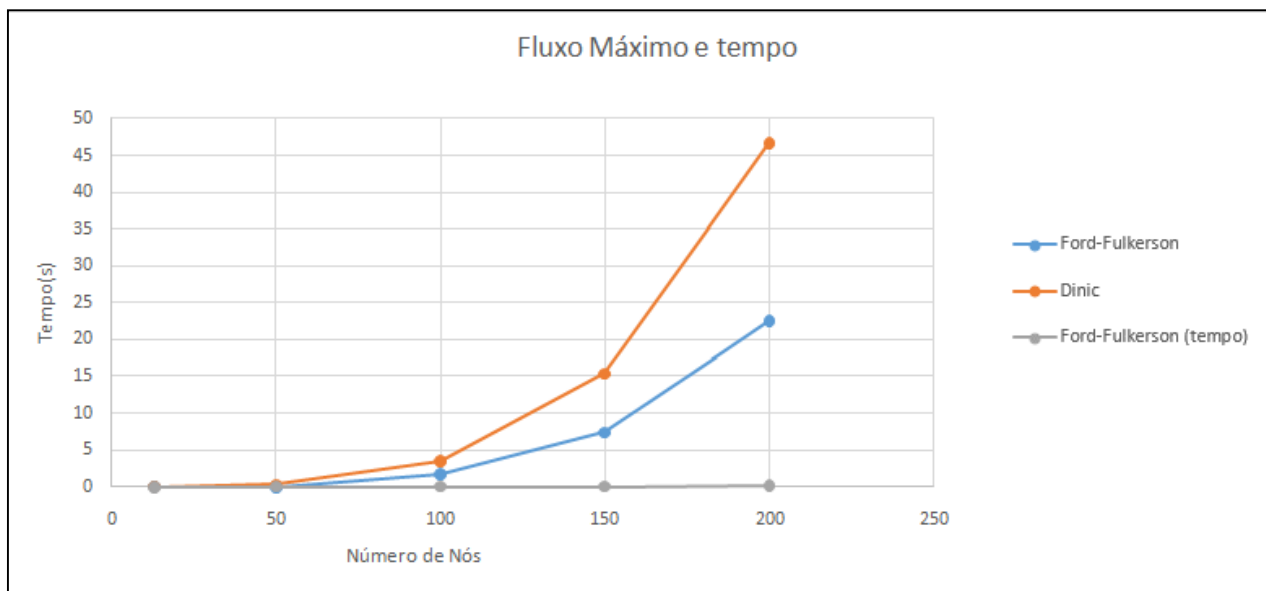
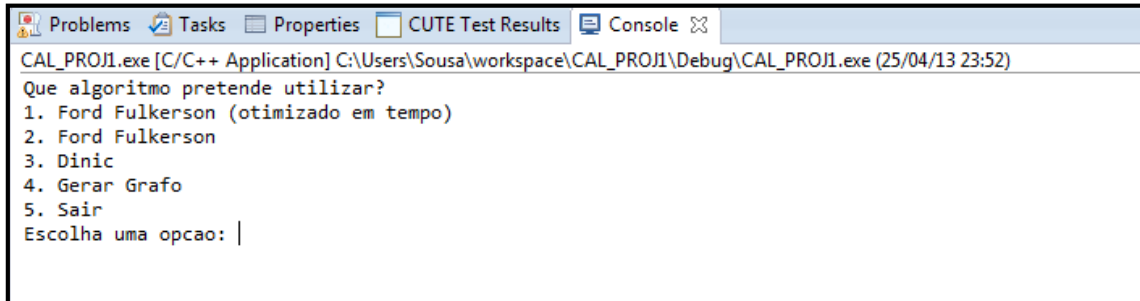


Gráfico 1: gráfico correspondente à tabela 1

4. CASOS DE UTILIZAÇÃO

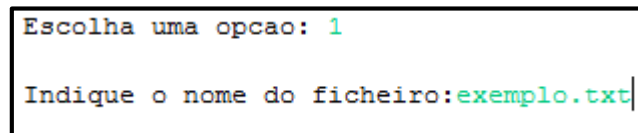
Inicialmente é apresentado ao utilizador o seguinte menu:



```
Problems Tasks Properties CUTE Test Results Console
CAL_PROJ1.exe [C/C++ Application] C:\Users\Sousa\workspace\CAL_PROJ1\Debug\CAL_PROJ1.exe (25/04/13 23:52)
Que algoritmo pretende utilizar?
1. Ford Fulkerson (otimizado em tempo)
2. Ford Fulkerson
3. Dinic
4. Gerar Grafo
5. Sair
Escolha uma opcao: |
```

Figura 1

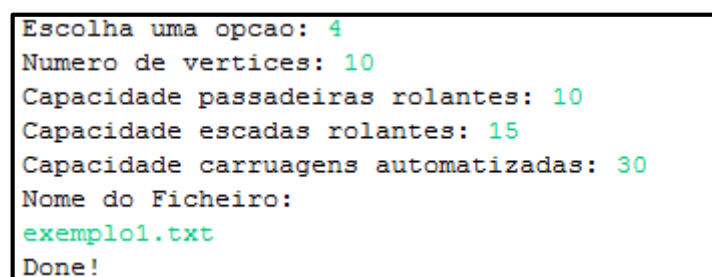
Qualquer uma das opções de 1 a 3 irá levar ao seguinte menu, em que é pedido o nome do ficheiro do ficheiro de texto que possui a informação sobre a rede.



```
Escolha uma opcao: 1
Indique o nome do ficheiro: exemplo.txt|
```

Figura 5: menu de escolha de ficheiro de texto

Se o utilizador escolher a opção 4 no menu inicial, um grafo aleatório será gerado segundo os limites que este introduzirá e será guardado no ficheiro que indicar, sendo que em seguida, será novamente apresentado o menu inicial.



```
Escolha uma opcao: 4
Numero de vertices: 10
Capacidade passadeiras rolantes: 10
Capacidade escadas rolantes: 15
Capacidade carruagens automatizadas: 30
Nome do Ficheiro:
exemplo1.txt
Done!
```

Ilustração 6: exemplo de geração de grafo aleatório

A opção 5 faz com que o programa termine.

Após a escolha do algoritmo a utilizar, irá surgir o seguinte menu:

```
1. Alterar a capacidade de uma aresta
2. Visualizar o grafo
3. Sair
Escolha uma opcao:
```

Figura 7: menu de opções para cada algoritmo

Se o utilizador escolher a opção 1 poderá alterar a capacidade de uma aresta existente.

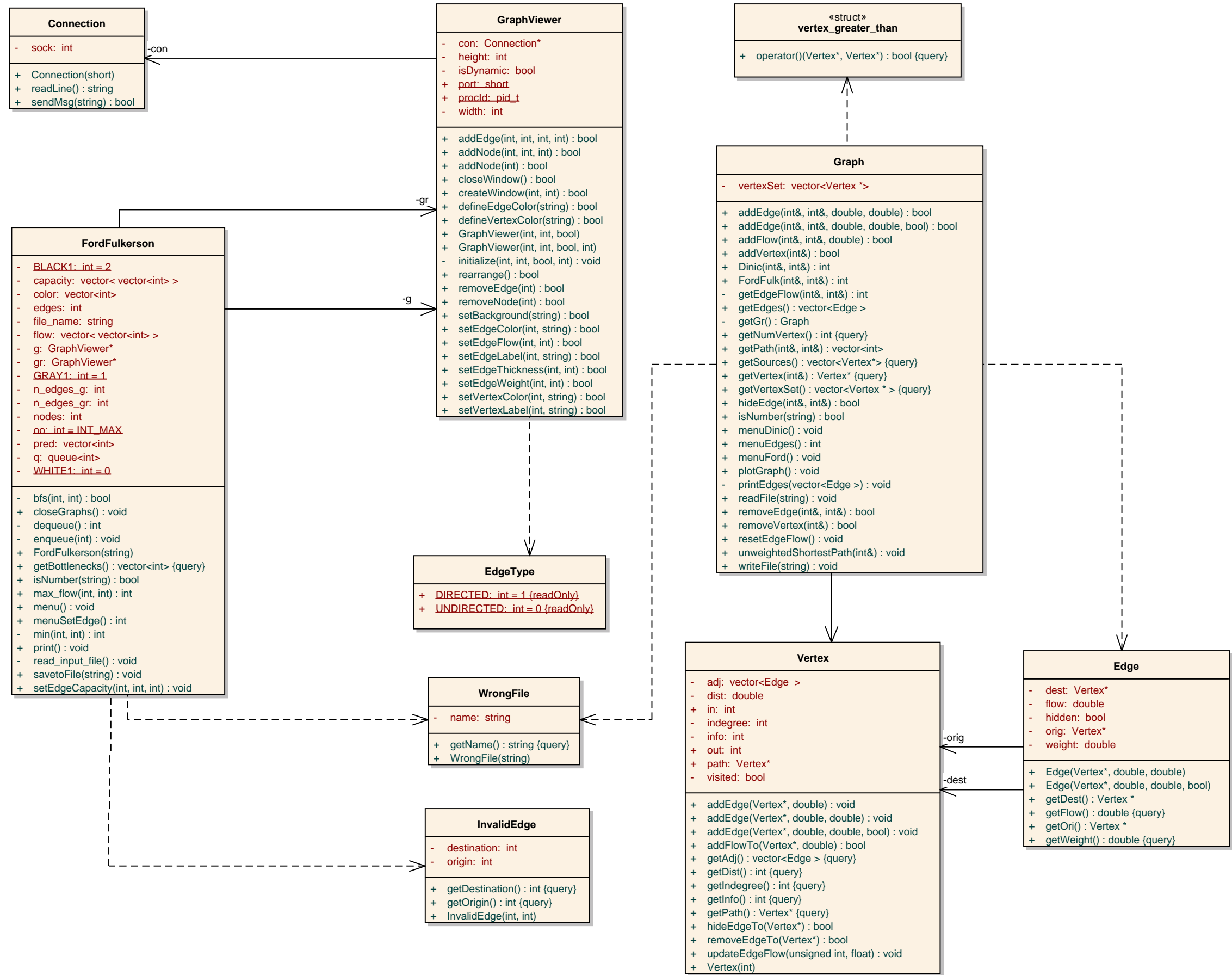
```
Escolha uma opcao: 1

Indique a fonte da aresta: 0
Indique o destino da aresta:1
Indique a nova capacidade da aresta:10
```

Figura 8: exemplo de alteração da capacidade de uma aresta

Se escolher a opção 2 irá visualizar o grafo correspondente (devidamente explicado na secção número 2) e se escolher a opção 3 volta ao menu inicial.

5. DIAGRAMA UML



6. DIFICULDADES SENTIDAS

As principais dificuldades sentidas pelo grupo foram:

- ✚ Perceber qual a forma mais apropriada de abordar o problema proposto;
- ✚ Encontrar a maneira mais correta para implementar os algoritmos, o que foi ultrapassado com a leitura dos slides bem como de documentação.

7. ESFORÇO DE CADA ELEMENTO

Consideramos que o trabalho foi igualmente distribuído pelos três membros do grupo, tendo todos ajudado para a concretização das tarefas planeadas, sendo que atribuímos a cada um responsabilidade de $1/3$ do trabalho.