INSTITUTO SUPERIOR TÉCNICO

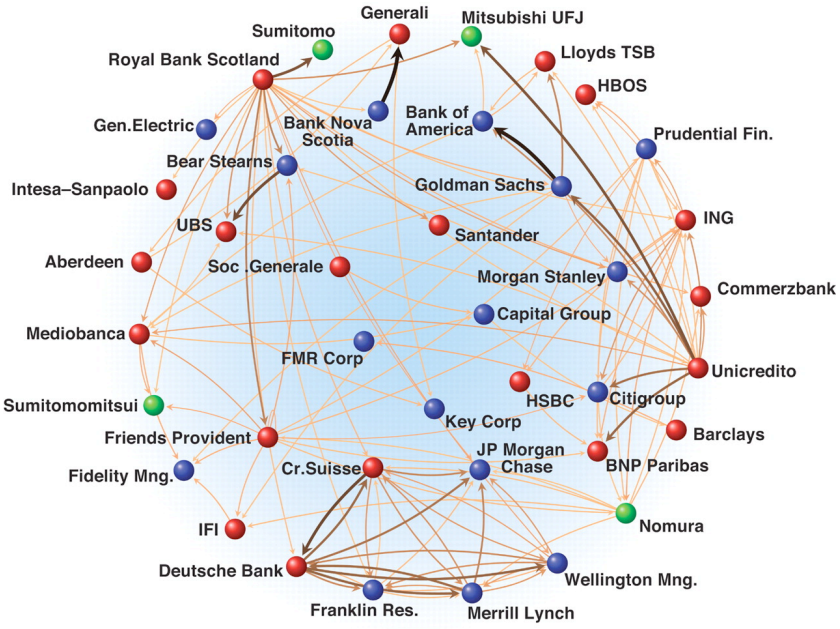
Introdução aos Algoritmos e Estruturas de Dados

2014/2015 - 2º semestre

Enunciado do 1º Projecto

Data de entrega: 24 de Março de 2015 (23h59)

1. Introdução



A recente crise financeira demonstrou a necessidade de compreendermos a forma como os sistemas financeiros estão interligados. Uma das formas de o fazer é mapear o sistema financeiro numa rede, onde cada nó (ou vértice) representa um banco e uma ligação (orientada) entre dois nós representa um empréstimo. Este tipo de representação permite analisar a robustez de uma rede de interacções financeira face, por exemplo, a um colapso de um dos seus intervenientes.

Neste projecto pretendemos desenvolver um programa em C que registe e manipule a informação referente ao aparecimento e desaparecimento de ligações entre bancos, providenciado uma base para o desenvolvimento de ferramentas de análise mais complexas.

2. Especificação do programa

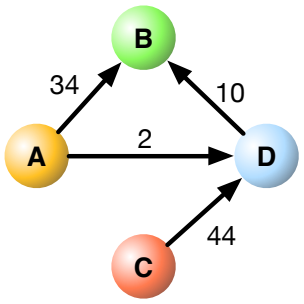
No cadastro de cada banco deve figurar o número de registo, o nome do banco, e a sua classificação actual. A classificação de um banco poderá representar o seu *rating* actual, podendo, neste contexto, tomar dois valores: `1` ("*bom*") ou `0` ("*mau*"). Ao longo do tempo o número de bancos *bons* em estudo pode variar, por registo de novas instituições, bem como pela desclassificação e promoção de outras, assumindo-se que o limite máximo de registos é de 10^3 bancos.

Sempre que um empréstimo é efectuado, é adicionada uma ligação entre os dois bancos envolvidos. O utilizador pode pedir ao programa listagens dos bancos registados, entre outras informações.

3. Sugestão de estrutura de dados

Há várias estruturas de dados apropriadas para representar uma rede orientada, tal como é pedido neste projecto. A estrutura mais simples talvez seja a matriz de adjacências[1], cuja a ideia passamos a exemplificar. Dada uma rede de N nós (ou vértices), uma rede poderá ser representada por uma matriz A, de tamanho $N \times N$, onde cada elemento A_{ij} representa o peso de uma ligação entre o banco i e o banco j. Neste caso, A_{ij} poderá representar o valor total da dívida de j para com i.

Considere o seguinte exemplo, constituído por 4 bancos (ou nós): A, B, C e D. O banco A emprestou 34 unidades monetárias (UM) ao banco B e 2 UM ao banco D, o banco C emprestou 44 UM ao banco D e, finalmente, o banco D emprestou 10 UM ao B.



	A	B	C	D
A	0	34	0	2
B	0	0	0	0
C	0	0	0	44
D	0	10	0	0

Podemos visualizar a rede resultante do lado esquerdo na figura em cima. Se recorrermos a uma representação da mesma rede através de uma matriz de adjacência obtemos a tabela do painel da direita.

4. Dados de Entrada

O programa deverá ler os dados de entrada a partir do standard input, na forma de um conjunto de linhas iniciadas por um carácter, que se passa a designar por *comando*, seguido de um número variável de informações; o comando e cada uma das informações são separados por um espaço e assim, dentro de cada informação, não poderão existir espaços.

Os comandos disponíveis são descritos de seguida e correspondem às letras `a, k, r, e, l, K, x.` Cada comando indica uma determinada acção que se passa a caracterizar em termos de objectivo, número de informações por linha (n.i.l.), sintaxe e output:

`a`: adicionar um novo banco (n.i.l. = 3)

```
a nome classificação ref
```

- `nome`: máximo **41 letras**;
- `classificação`: assumido *bom* (`=1`) ou *mau* (`=0`);

[1] http://en.wikipedia.org/wiki/Adjacency_matrix

- `ref`: referência/código do banco (e.g., `2189000`), dado por um número inteiro positivo. Poderá assumir que o utilizador nunca insere dois bancos com a mesma referência.

Observações: não implica qualquer output.

`k`: classificar um banco com o rating *mau* (n.i.l. = 1)

```
k ref
```

- `ref`: referência do banco a atribuir a classificação "0" (*mau*).

Observações: não implica qualquer output.

`r`: promover um banco ao rating *bom* (n.i.l. = 1)

```
r ref
```

- `ref`: referência do banco a atribuir a classificação "1" (*bom*).

Observações: não implica qualquer output.

`e`: adicionar empréstimo (n.i.l. = 3)

```
e ref1 ref2 valor
```

- `ref1`: referência do banco que fornece o crédito;
- `ref2`: referência do banco que recebe o empréstimo;
- `valor`: valor (inteiro positivo) emprestado em UM.

Observações: a) não implica qualquer output; b) situações onde `ref1=ref2` não serão contempladas no input; c) empréstimos recíprocos (A empresta a B e B empresta A) são tomados como independentes; as amortizações são introduzidas com o comando seguinte.

`p`: adicionar amortização de empréstimo (n.i.l. = 3)

```
p ref1 ref2 valor
```

- `ref1`: identificação do banco pagador;
- `ref2`: identificação do banco que recebe o valor;
- `valor`: valor (inteiro positivo, menor ou igual ao valor em dívida) pago em UM.

Observações: não implica qualquer output; situações onde `ref1=ref2` não serão contempladas no input.

`l`: emitir listagem (n.i.l. =1)

```
l tipo
```

- `tipo`: número inteiro identificador da listagem pretendida, a definir na Secção 5.

Observações: ver Secção 5 para detalhes do output.

`K`: despromover banco "*bom*" em maior dificuldades. (n.i.l. =0)

```
K
```

Procura e desclassifica o banco *"bom"* com maior exposição à divida de bancos *"maus"*, ou seja, o banco de classificação igual a "1" com maior valor emprestado a bancos "*maus*" (i.e., com classificação igual a "0"). Se existir mais do que um banco nestas condições, o comando `K` deverá despromover aquele que tiver sido introduzido mais recentemente no sistema.

Observações: ver Secção 5 para detalhes do output. Caso não haja nenhum banco *bom* com um valor superior a 0 emprestado a bancos *maus*, o o sistema ficará inalterado. A

execução sucessiva deste comando permite analisar a robustez da rede e eventuais efeitos de "cascata" que poderão ocorrer.

`x`: sair do programa (n.i.l. = 0)
`x`
Observações: ver Secção 5 para detalhes do output.

5. Dados de Saída

O programa deverá escrever no standard output as respostas a certos comandos apresentados no standard input. As respostas são igualmente linhas de texto, com informações separadas por um espaço, não podendo haver espaços em cada uma das informações.

As respostas aos pedidos no input são as seguintes

- Como resposta ao **comando `x`**, o programa deverá escrever o número de total de bancos registados, seguido do número total de bancos *bons*. Os dois valores deverão ser impressos na mesma linha, separado por um espaço.

- Como resposta ao **comando `l 0`**, o programa deverá listar todos os bancos ordenados pela ordem de introdução no sistema. Cada banco deverá ser impresso numa linha distinta na forma:
`ref nome classificação`
onde,
- `ref`: referência do banco;
- `nome`: nome do banco;
- `classificação`: classificação do banco (*mau*=0; *bom*=1).

- Como resposta ao **comando `l 1`**, o programa deverá listar todos os bancos ordenados pela ordem de introdução no sistema. Cada banco deverá ser impresso numa linha distinta na forma:

`ref nome classificação inP outP outV outVM inV inVM`

onde, para além dos elementos descritos no ponto anterior, temos
`inP`: número total de bancos parceiros a quem o banco tem uma dívida;
`outP`: número total de bancos parceiros a quem o banco tem dinheiro emprestado;
`outV`: valor total emprestado pelo banco em questão a outros bancos;
`outVM`: valor total emprestado pelo banco em questão a bancos *maus*;
`inV`: valor total emprestado ao banco em questão por outros bancos;
`inVM`: valor total emprestado ao banco em questão por bancos *maus*;

- Como resposta ao **comando `l 2`**, o programa deverá escrever a distribuição/histograma *d(k)* do número de bancos actualmente com *exactamente k* parceiros comerciais, sobe a forma:

```
0 d(0)
1 d(1)
2 d(2)
```
etc.

Entende-se por parceiro comercial um qualquer banco a quem se tem uma divida (não liquidada), ou qualquer banco a quem se emprestou uma quantia (não liquidada). Usando a nomenclatura do ponto anterior, o número de parceiros de um banco é sempre menor ou igual a `inP + outP`.[2]

Por exemplo, o output poderá tomar a forma

```
0 2
1 7
2 3
3 1
6 1
```

correspondendo a primeira coluna a *k* e a segunda ao número de indivíduos com *k* parceiros, i.e., *d(k)*. No exemplo acima, a segunda linha indica que existem 7 indivíduos com apenas 1 parceiro. Quando *d(k)=0*, a respectiva linha deverá ser omitida. No exemplo acima, por não exisitirem bancos com 4 parceiros comerciais, não existe nenhuma linha que comece com 4.

- Como resposta ao **comando `K`**, o programa deverá mostrar os dados do banco desclassificado (após a desclassificação) — tal como no comando `l 1` (ver detalhes em cima) — precedido do carácter '`*`'. Esta informação deverá ser seguida de uma nova linha com a informação sobre o número total de bancos registados, seguido do número total de bancos *bons* (tal como no comando `x`) após a desclassificação. Exemplo:

```
*ref nome classificação inP outP outV outVM inV inVM
Ntotal Nbons
```

Caso não haja nenhum banco em condições de ser desclassificado (ver em cima), o programa deverá escrever apenas a segunda linha:

```
Ntotal Nbons
```

6. Exemplos (Input/Output)

Exemplo 1

Dados de Entrada:

```
a A-bank 1 100340
a B-bank 1 3011100
a C-bank 1 410040
a D-bank 1 550340
k 3011100
l 0
x
```

Dados de Saída:

```
100340 A-bank 1
3011100 B-bank 0
410040 C-bank 1
550340 D-bank 1
4 3
```

[2]Note que um banco pode pertencer simultaneamente ao conjunto `inP` e `outP`, dado que um banco A pode fazer um empréstimo ao banco B, e B fazer um empréstimo a A.

Exemplo 2

Dados de Entrada:

```
a A-bank 1 100340
a B-bank 1 3011100
a C-bank 1 410040
a D-bank 1 550340
e 100340 3011100 34
e 100340 550340 2
e 410040 550340 44
e 550340 3011100 10
l 1
k 3011100
l 1
x
```

Dados de Saída:

```
100340 A-bank 1 0 2 36 0 0 0
3011100 B-bank 1 2 0 0 0 44 0
410040 C-bank 1 0 1 44 0 0 0
550340 D-bank 1 2 1 10 0 46 0
100340 A-bank 1 0 2 36 34 0 0
3011100 B-bank 0 2 0 0 0 44 0
410040 C-bank 1 0 1 44 0 0 0
550340 D-bank 1 2 1 10 10 46 0
4 3
```

Exemplo 3

Dados de Entrada:

```
a A-bank 1 100340
a B-bank 1 3011100
a C-bank 1 410040
a D-bank 1 550340
e 100340 3011100 34
e 100340 550340 2
e 410040 550340 44
e 550340 3011100 10
k 3011100
e 100340 3011100 20
l 1
K
K
K
K
l 1
x
```

Dados de Saída:

```
100340 A-bank 1 0 2 56 54 0 0
3011100 B-bank 0 2 0 0 0 64 0
410040 C-bank 1 0 1 44 0 0 0
550340 D-bank 1 2 1 10 10 46 0
*100340 A-bank 0 0 2 56 54 0 0
4 2
*550340 D-bank 0 2 1 10 10 46 2
4 1
*410040 C-bank 0 0 1 44 44 0 0
4 0
4 0
100340 A-bank 0 0 2 56 56 0 0
3011100 B-bank 0 2 0 0 0 64 64
410040 C-bank 0 0 1 44 44 0 0
550340 D-bank 0 2 1 10 10 46 46
4 0
```

Exemplo 4

Dados de Entrada:

```
a ICBC 1 1
a HSBC_Holdings 1 2
a China_Const_Bank_Corporation 1 3
a BNP_Paribas 1 4
a Mitsubishi_UFJ_Fin_Group 1 5
a JPMorgan_Chase_&_Co 1 6
```

Dados de Saída:

```
0 3
1 5
2 1
5 1
10 10
```

```
a Agricultural_Bank_of_China 1 7
a Bank_of_China 1 8
a Crédit_Agricole_Group 1 9
a Barclays_PLC 1 10
e 1 2 10
e 1 3 10
e 1 4 10
e 1 5 10
e 1 6 10
e 4 9 2
e 4 9 2
e 1 3 10
e 1 3 10
e 1 4 10
e 4 1 2
l 2
x
```

7. Compilação do Programa

O compilador a utilizar é o `gcc` com as seguintes opções de compilação: `-ansi -Wall -pedantic`. Para compilar o programa deve executar o seguinte comando:

```
$ gcc -ansi -Wall -pedantic -o proj1 *.c
```

o qual deve ter como resultado a geração do ficheiro executável `proj1`, caso não haja erros de compilação. A execução deste comando não deverá escrever qualquer resultado no terminal. Caso a execução deste comando escreva algum resultado no terminal, considera-se que o programa não compilou com sucesso. Por exemplo, durante a compilação do programa, o compilador não deve escrever mensagens de aviso (warnings).

8. Execução do Programa

O programa deve ser executado da forma seguinte:

```
$ ./proj1 < test01.in > test01.myout
```

Posteriormente poderá comparar o seu output com o output previsto usando o comando **`diff`**

```
$ diff test01.out test01.myout
```

9. Entrega do Projecto

A entrega do projecto deverá respeitar o procedimento seguinte:

- Na página da disciplina aceda ao sistema para entrega de projectos. O sistema será activado uma semana antes da data limite de entrega. Instruções acerca da forma de acesso ao sistema serão oportunamente fornecidas.

- Efectue o upload de um ficheiro arquivo com extensão `.zip` que inclua todos os ficheiros fonte que constituem o programa. Se utilizar ficheiros cabeçalho (**`.h`**) não se esqueça de os juntar também ao pacote.

- Para criar um ficheiro arquivo com a extensão .zip deve executar o seguinte comando na directoria onde se encontram os ficheiros com extensão .c e .h (se for o caso), criados durante o desenvolvimento do projecto:
  ```
  $ zip proj1.zip *.c *.h
  ```

  Se só tiver um único ficheiro (e.g., proj1.c), bastará escrever:
  ```
  $ zip proj1.zip proj1.c
  ```

- Como resultado do processo de upload será informado se a resolução entregue apresenta a resposta esperada num conjunto de casos de teste.

- O sistema não permite submissões com menos de 10 minutos de intervalo para o mesmo grupo. Exemplos de casos de teste serão oportunamente fornecidos.

- Data limite de entrega do projecto: 24 de Março de 2015 (23h59m). Até à data limite poderá efectuar o número de submissões que desejar, sendo utilizada para efeitos de avaliação a última submissão efectuada. Deverá portanto verificar cuidadosamente que a última submissão corresponde à versão do projecto que pretende que seja avaliada. Não existirão excepções a esta regra.

10. Avaliação do Projecto

a. Componentes da Avaliação

Na avaliação do projecto serão consideradas as seguintes componentes:

1. A primeira componente avalia o desempenho da funcionalidade do programa realizado. Esta componente é avaliada entre 0 e 16 valores.
2. A segunda componente avalia a qualidade do código entregue, nomeadamente os seguintes aspectos: comentários, identação, estruturação, modularidade, abstracção, entre outros. Esta componente poderá variar entre -4 valores e +4 valores relativamente à classificação calculada no item anterior e será atribuída na discussão final do projecto.
3. Durante a discussão final do projecto será averiguada a participação de cada elemento do grupo na realização do projecto, bem como a sua compreensão do trabalho realizado. A respectiva classificação será ponderada em conformidade, isto é, elementos do mesmo grupo podem ter classificações diferentes. Elementos do

grupo que se verifique não terem participado na realização do respectivo projecto terão a classificação de 0 (zero) valores.

b. Atribuição Automática da Classificação

- A classificação da primeira componente da avaliação do projecto é obtida através da execução *automática* de um conjunto de testes num computador com o sistema operativo GNU/Linux. Torna-se portanto essencial que o código compile correctamente e que respeite o formato de entrada e saída dos dados descrito anteriormente. Projectos que não obedeçam ao formato indicado no enunciado serão penalizados na avaliação automática, podendo, no limite, ter 0 (zero) valores se falharem todos os testes. Os testes considerados para efeitos de avaliação poderão incluir (ou não) os disponibilizados na página da disciplina, além de um conjunto de testes adicionais. A execução de cada programa em cada teste é limitada na quantidade de memória que pode utilizar, até um máximo de 64 Mb, e no tempo total disponível para execução, sendo o tempo limite distinto para cada teste.
- Note-se que o facto de um projecto passar com sucesso o conjunto de testes disponibilizado na página da disciplina não implica que esse projecto esteja totalmente correcto. Apenas indica que passou alguns testes com sucesso, mas este conjunto de testes não é exaustivo. É da responsabilidade dos alunos garantir que o código produzido está correcto.
- Em caso algum será disponibilizado qualquer tipo de informação sobre os casos de teste utilizados pelo sistema de avaliação automática. A totalidade de ficheiros de teste usados na avaliação do projecto serão disponibilizados na página da disciplina após a data de entrega.