Trabalho Final de LPG 2018.2

Vinícius Takeo Friedrich Kuwaki

Bibliotecas utilizadas, variáveis globais e protótipos das funções

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int semente;
```

```c
int **CriaMatrizAdjacencia (int num_membros);

void DesalocaMatriz(int num_membros, int **matriz_adjacencia);

void ImprimeMatriz(int num_membros, int **matriz_adjacencia);

void MarcaComoAmigo (int pessoa1, int pessoa2, int **matriz_adjacencia);

float GeraNumeroAleatorio();

void GeraRedeSocialAleatoria( float p, int num_membros, int **matriz_adjacencia);

int numAmigosEmComum (int v, int u, int num_membros, int **matriz_adjacencia);

void LimpaTela();

float coeficienteAglomeracao (int v, int num_membros, int **matriz_adjacencia);
```

Funções do código e seus papéis: main e suas variáveis

```c
int main(){

int i,j;

int u,v;

int num_membros;
int **matriz_adjacencia;

printf("Entre com o numero de membros da rede social:\n");
scanf("%d",&num_membros);
LimpaTela();

printf("Entre com a semente que sera usada na geracao de numeros aleatorios:\n");
scanf("%d",&semente);
srand(semente);
LimpaTela();

matriz_adjacencia = CriaMatrizAdjacencia(num_membros);

GeraRedeSocialAleatoria(GeraNumeroAleatorio(), num_membros, matriz_adjacencia);

ImprimeMatriz(num_membros,matriz_adjacencia);
```

Chamando a função CriaMatrizAdjacência

```
matriz_adjacencia = CriaMatrizAdjacencia(num_membros);
```

```
int **CriaMatrizAdjacencia (int num_membros){

int **matriz;
int i,j;

matriz = (int**)calloc(sizeof(int*),num_membros);
for(i=0;i<num_membros;i++)
    matriz[i] = (int*)calloc(sizeof(int),num_membros);

for(i=0;i<num_membros;i++)
    for(j=0;j<num_membros;j++)
        matriz[i][j] = 0;

return matriz;
}
```

Gerando a rede social aleatória

```
GeraRedeSocialAleatoria(GeraNumeroAleatorio(), num_membros, matriz_adjacencia);
```

```
void GeraRedeSocialAleatoria( float p, int num_membros, int **matriz_adjacencia){

    int i,j;
    float r;

    for(i=0;i<num_membros-1;i++){
        for(j=i+1;j<num_membros;j++){
            r = GeraNumeroAleatorio();
            if( r < p )
                MarcaComoAmigo(i,j, matriz_adjacencia);
        }
    }
}
```

```
void MarcaComoAmigo (int pessoa1, int pessoa2, int **matriz_adjacencia){

    matriz_adjacencia[pessoa1][pessoa2] = 1;
    matriz_adjacencia[pessoa2][pessoa1] = 1;

}
```

```
float GeraNumeroAleatorio(){

int numInt;
float x;

numInt = (rand()%100);
x = (float)numInt / 100.00;

return (x);
}
```

Imprimindo a matriz

```
ImprimeMatriz(num_membros,matriz_adjacencia);
```

```
void ImprimeMatriz(int num_membros, int **matriz_adjacencia){

    int i,j;

    printf(" \t");
    for(i=0;i<num_membros;i++){
        printf("%d\t",i);
    }
    printf("\n\n");

    for(i=0;i<num_membros;i++){
        printf("%d\t",i);
        for(j=0;j<num_membros;j++){
            printf("%d\t",matriz_adjacencia[i][j]);
        }
        printf("\n\n");
    }

}
```

```
        0       1       2       3       4       5       6

0       0       1       0       0       1       1       1

1       1       0       0       0       1       0       0

2       0       0       0       0       1       1       1

3       0       0       0       0       1       0       1

4       1       1       1       1       0       0       1

5       1       0       1       0       0       0       0

6       1       0       1       1       1       0       0
```

Imprimindo o número de amigos em comum

```
printf("\n");
for(i=0;i<num_membros;i++){
    for(j=i+1;j<num_membros;j++){
        printf("Numero de amigos em comum entre %d e %d eh: %d\n",i,j,numAmigosEmComum(i,j,num_membros,matriz_adjacencia));
    }
}
```

```
int numAmigosEmComum (int v, int u, int num_membros, int **matriz_adjacencia){

    int contadorAmigos = 0;
    int i;

    for( i=0;i<num_membros;i++ ){
        if( matriz_adjacencia[u][i] == 1 && matriz_adjacencia[v][i] == 1 ){
            contadorAmigos++;
        }
    }

    return contadorAmigos;

}
```

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

```
Numero de amigos em comum entre 0 e 1 eh: 1
Numero de amigos em comum entre 0 e 2 eh: 3
Numero de amigos em comum entre 0 e 3 eh: 2
Numero de amigos em comum entre 0 e 4 eh: 2
Numero de amigos em comum entre 0 e 5 eh: 0
Numero de amigos em comum entre 0 e 6 eh: 1
Numero de amigos em comum entre 1 e 2 eh: 1
Numero de amigos em comum entre 1 e 3 eh: 1
Numero de amigos em comum entre 1 e 4 eh: 1
Numero de amigos em comum entre 1 e 5 eh: 1
Numero de amigos em comum entre 1 e 6 eh: 2
Numero de amigos em comum entre 2 e 3 eh: 2
Numero de amigos em comum entre 2 e 4 eh: 1
Numero de amigos em comum entre 2 e 5 eh: 0
Numero de amigos em comum entre 2 e 6 eh: 1
Numero de amigos em comum entre 3 e 4 eh: 1
Numero de amigos em comum entre 3 e 5 eh: 0
Numero de amigos em comum entre 3 e 6 eh: 1
Numero de amigos em comum entre 4 e 5 eh: 2
Numero de amigos em comum entre 4 e 6 eh: 3
Numero de amigos em comum entre 5 e 6 eh: 2
```

Imprimindo o coeficiente de aglomeração

```
printf("\n");
    for(i=0;i<num_membros;i++)
        printf("A probabilidade de dois amigos de %d tambem serem amigos entre si eh de: %.2f\n",i,
               coeficienteAglomeracao(i,num_membros,matriz_adjacencia)*100);
```

```
float coeficienteAglomeracao (int v, int num_membros, int **matriz_adjacencia){

int amigosV = 0;
int cont=0;
float maxAmizades, contFloat;
int i,j,k;
int *vetorAuxiliar;
```

```
for(i=0;i<num_membros;i++){
    if(i == v)
        i++;
    if(matriz_adjacencia[v][i] == 1){
        amigosV++;
    }
}
```

amigosV = 4

```
4       1       1       1       1       0       0       1
```

```
if(amigosV == 1 || amigosV == 0)
    return 0;

vetorAuxiliar = (int*)calloc(sizeof(int),amigosV);
```

```
for(i=0;i<num_membros;i++){
    if(i == v)
        i++;
    if(matriz_adjacencia[v][i] == 1){
        vetorAuxiliar[cont] = i;
        cont++;
    }
}
```

matriz_adjacencia

NA LINHA 4

1	1	1	1	0	1
i=0	i=1	i=2	i=3	i=4	i=5

vetorAuxiliar

0	1	2	3	5
cont=0	cont=1	cont=2	cont=3	cont=4

```
cont = 0;
for( i=0; i<amigosV; i++){
    for( j=i+1; j<amigosV; j++){
        if(matriz_adjacencia[vetorAuxiliar[i]][vetorAuxiliar[j]] == 1){
            cont++;
        }
    }
}
```

matriz_adjacencia [0][1] == 1? SIM cont = 1

matriz_adjacencia [0][2] == 1? NÃO cont = 1

matriz_adjacencia [0][3] == 1? NÃO cont = 1

matriz_adjacencia [0][6] == 1? SIM cont = 2

matriz_adjacencia [1][2] == 1? NÃO cont = 2

matriz_adjacencia [1][3] == 1? NÃO cont = 2

matriz_adjacencia [1][6] == 1? NÃO cont = 2

matriz_adjacencia [2][3] == 1? NÃO cont = 2

matriz_adjacencia [2][6] == 1? SIM cont = 3

matriz_adjacencia [3][6] == 1? SIM cont = 4

cont = 4

Calculando o coeficiente para a

```
maxAmizades = amigosV*(amigosV - 1)/2.0;
contFloat = (float)cont;

free(vetorAuxiliar);

return contFloat / maxAmizades;
```

cont = 4

amigosV = 5

maxAmizades = amigosV * (amigosV - 1) / 2

maxAmizades = 5 * (5 - 1) / 2

maxAmizades = 5 * 4 / 2

maxAmizades = 10

return (0.4)

```
A probabilidade de dois amigos de 0 tambem serem amigos entre si eh de: 33.33
A probabilidade de dois amigos de 1 tambem serem amigos entre si eh de: 100.00
A probabilidade de dois amigos de 2 tambem serem amigos entre si eh de: 33.33
A probabilidade de dois amigos de 3 tambem serem amigos entre si eh de: 100.00
A probabilidade de dois amigos de 4 tambem serem amigos entre si eh de: 40.00
A probabilidade de dois amigos de 5 tambem serem amigos entre si eh de: 0.00
A probabilidade de dois amigos de 6 tambem serem amigos entre si eh de: 50.00
```

Desalocando a matriz de adjacência

```
DesalocaMatriz(num_membros,matriz_adjacencia);

return 0;
}
```

```
void DesalocaMatriz(int num_membros, int **matriz_adjacencia){

int i;

for(i=0;i<num_membros;i++)
    free(matriz_adjacencia[i]);

free(matriz_adjacencia);

}
```