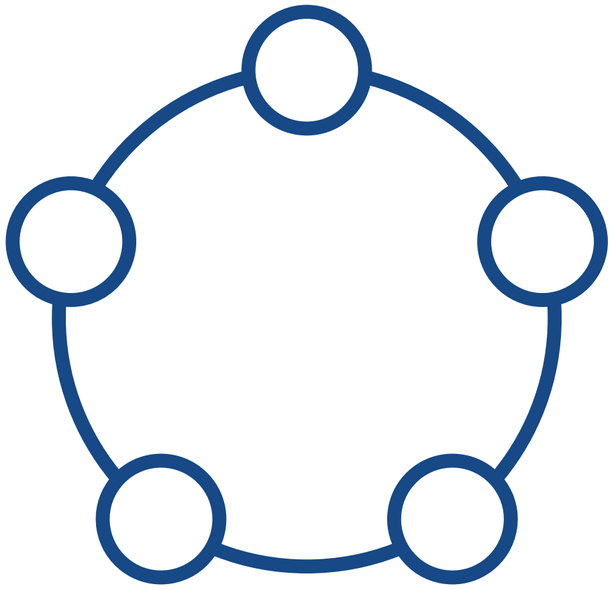


Semana 7

Teoria de Grafos

Ana Luiza Sticca, Gabriel Antônio, Guilherme Cabral e João Pedro Marques

Problemas

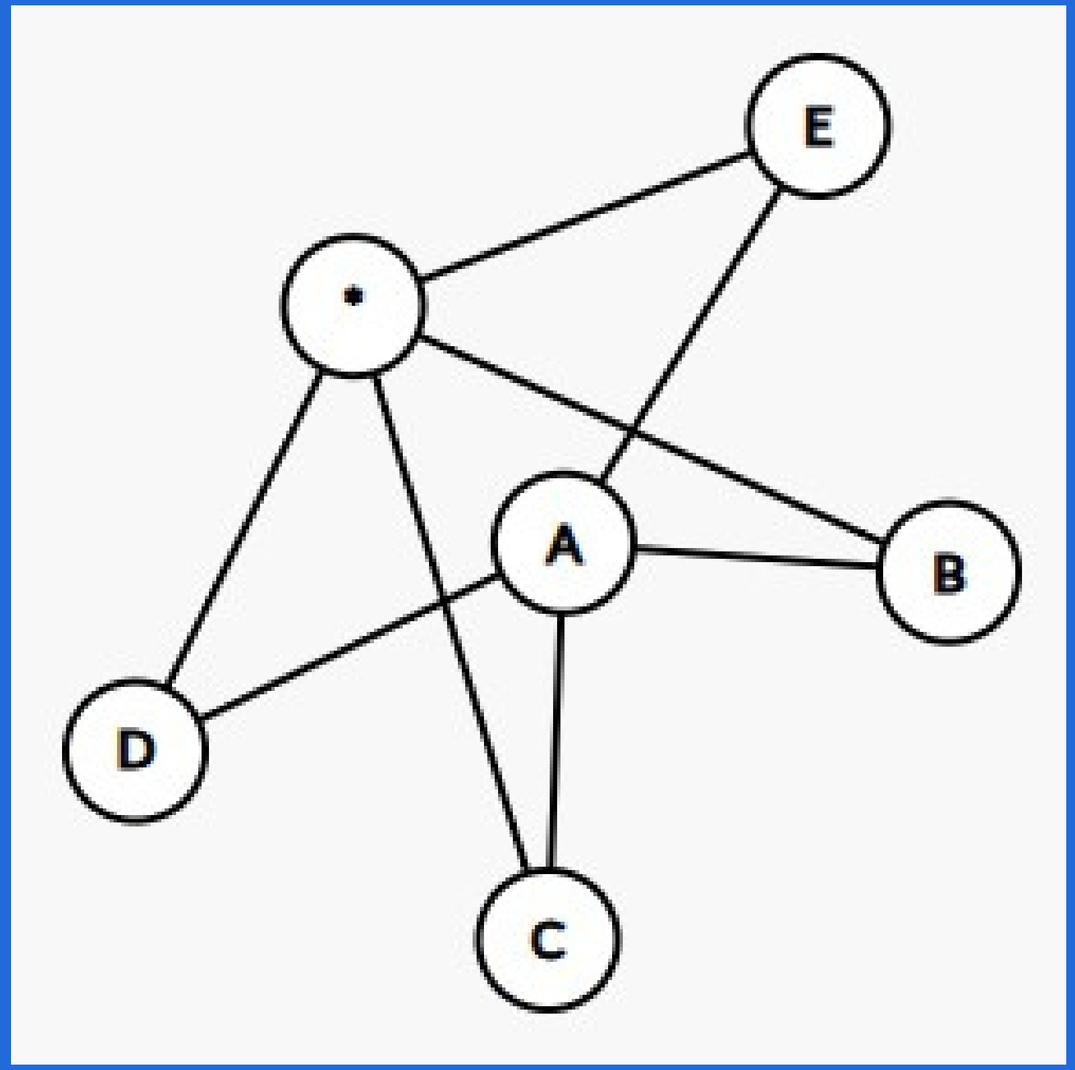
11 Galactic Import

12 Robot

11

Galactic Import

Galactic Import



Galactic Import

Estrutura de dados: Listas de Adjacência

Vértices: Galáxias

Arestas: Exportações entre as galáxias

Galactic Import

Sample Input

```
1
F 0.81 *
5
E 0.01 *A
D 0.01 A*
C 0.01 *A
A 1.00 EDCB
B 0.01 A*
10
S 2.23 Q*
A 9.76 C
K 5.88 MI
E 7.54 GC
M 5.01 OK
G 7.43 IE
I 6.09 KG
C 8.42 EA
O 4.55 QM
Q 3.21 SO
```

Sample Output

```
Import from F
Import from A
Import from A
```

Galactic Import (versão 1)

```
vector< int > grafo[NMAX];

double vet[NMAX];

int dist[NMAX];

double pot095[NMAX];

int getId(char c)
{

        if(c == '*') return 0;

        return (int)(c - 'A') + 1;

}
```

```
char getChar(int u)
{

        if(u == 0) return '*';

        return (char)(u - 1 + 'A');

}
```

```cpp
void BFS(int init)
{

        int u;

        queue< int > fila;

        memset(dist, -1, sizeof dist);

        dist[init] = 0;
        fila.push(init);

        while(fila.empty() == false)
        {

                u = fila.front();
                fila.pop();

                for(auto v : grafo[u])
                {

                        if(dist[v] == -1)
                        {

                                dist[v] = dist[u] + 1;
                                fila.push(v);

                        }

                }

}

return;
```

```
int main()
{

	int n, u, v, idBest, i;

	double best;

	char c;

	string s;

	pot095[0] = 1;
	for(i = 1;i < NMAX;i++)
		pot095[i] = pot095[i - 1] * 0.95;

	while(cin >> n)
	{

		for(i = 0;i < NMAX;i++) grafo[i].clear();

		for(i = 1;i <= n;i++)
		{

			cin >> c;
			u = getId(c);

			cin >> vet[u];

			cin >> s;

			for(auto cur : s)
			{

				v = getId(cur);

				grafo[u].push_back(v);
				if(v != '*') grafo[v].push_back(u);

			}

		}

		BFS(0);

		for(i = 0;i < NMAX;i++)
			vet[i] = vet[i] * pot095[dist[i]];

		best = 0;
		idBest = 0;

		for(i = 0;i < NMAX;i++)
		{

			if(best < vet[i])
			{

				best = vet[i];
				idBest = i;
```

Galactic Import (versão 2)

```java
public void findImport(){
    visited[0] = true;
    double[] tax = new double[visited.length];
    Arrays.fill(tax, 0.0);

    ArrayList<Integer> q = new ArrayList<>();
    int aux;
    q.add(0);

    while(!q.isEmpty()){
        aux = q.remove(0);
        //System.out.println("Visitando adjacentes de " + dic.get(aux));
        for(int w : adj[aux]){
            if(!visited[w]){
                visited[w] = true;
                //System.out.println("Subtraindo " + (tax[aux]*100) + "% d
                //System.out.println("Antes - " + value[w] + " vs Agora: "
                value[w] = value[w] - value[w]*tax[aux];
                tax[w] += tax[aux] + 0.05;
                q.add(w);
```

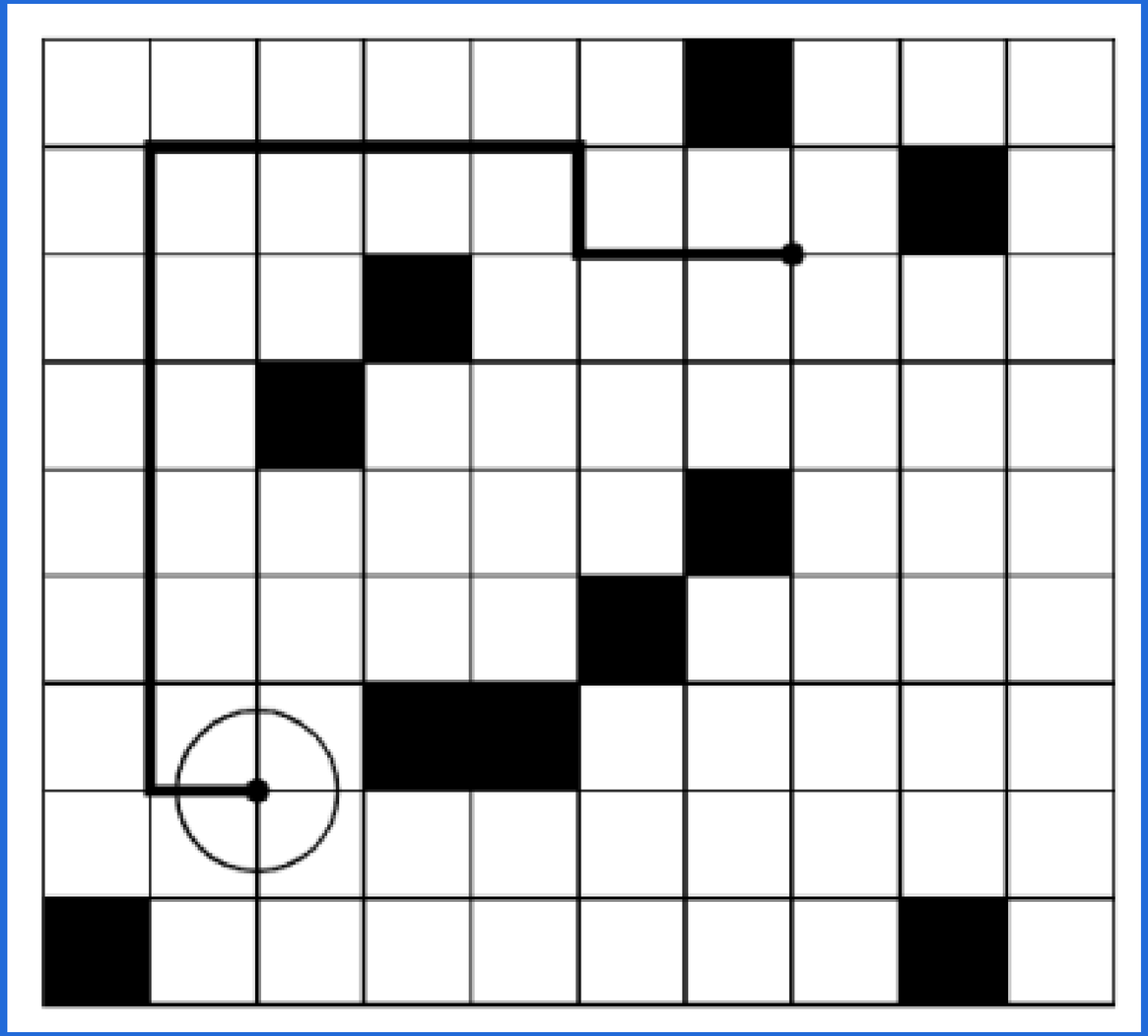
```
public void showValue(){
    int maior = 1;
    for(int w = 2; w < value.length; w++){
        if(visited[w]){
            //System.out.println("Comparando " + dic.get(w) + " - "+
            if(value[w] > value[maior]){
                maior = w;
            }else if(value[w] == value[maior]){
                if(dic.get(w).compareTo(dic.get(maior)) < 0)
                    maior = w;
            }
        }
    }
    System.out.println("Import from " + dic.get(maior));
}
```

Exemplos rodados

```
> sh -c make -s
> ./main
1
F 0.81 *
5
E 0.01 *A
D 0.01 A*
C 0.01 *A
A 1.00 EDCB
B 0.01 A*
10
S 2.23 Q*
A 9.76 C
K 5.88 MI
E 7.54 GC
M 5.01 OK
G 7.43 IE
I 6.09 KG
C 8.42 EA
O 4.55 QM
Q 3.21 SO
Import from F
Import from A
Import from A
```

12 Robot

Robot



Sample Input

```
9 10
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 1 0
7 2 2 7 south
0 0
```

Sample Output

```
12
```

Robot

Estrutura de dados: Listas de Adjacência

Vértices: cruzamento de trilhas na loja

Arestas: movimentos possíveis do robô

Robot (versão 1)

```
typedef struct Info
{

        int lin, col, dir;

}In;

int n, m;

int matInit[NMAX][NMAX];

bool mat[NMAX][NMAX];

int dist[NMAX][NMAX][5];

int dlin[] = {-1, 0, 1,  0};
int dcol[] = { 0, 1, 0, -1};
```

```
bool Valid(int lin, int col)
{

        return (1 <= lin && lin <= n && 1 <= col && col <= m);

}

void BFS(int ilin, int icol, int idir)
{

        int lin, col, dir, nlin, ncol, ndir;

        queue< In > fila;

        for(int i = 1;i <= n;i++)
                for(int j = 1;j <= m;j++)
                        for(int k = 0;k < 4;k++)
                                dist[i][j][k] = INF;

        dist[ilin][icol][idir] = 0;
        fila.push({ilin, icol, idir});
```

```
while(fila.empty() == false)
{

        lin = fila.front().lin;
        col = fila.front().col;
        dir = fila.front().dir;
        fila.pop();

        nlin = lin;
        ncol = col;
        ndir = (dir + 1) % 4;

        if(dist[nlin][ncol][ndir] == INF)
        {

                dist[nlin][ncol][ndir] = dist[lin][col][dir] + 1;
                fila.push({nlin, ncol, ndir});

        }

        nlin = lin;
        ncol = col;
        ndir = (dir + 3) % 4;

        if(dist[nlin][ncol][ndir] == INF)
        {

                dist[nlin][ncol][ndir] = dist[lin][col][dir] + 1;
                fila.push({nlin, ncol, ndir});

        }

        for(int i = 1;i <= 3;i++)
        {

                nlin = lin + i * dlin[dir];
                ncol = col + i * dcol[dir];
                ndir = dir;

                if(Valid(nlin, ncol) == false) break;
                if(mat[nlin][ncol] == false) break;

                if(dist[nlin][ncol][ndir] == INF)
                {

                        dist[nlin][ncol][ndir] = dist[lin][col][dir] + 1;
                        fila.push({nlin, ncol, ndir});
```

```
int main()
{

        int x, a, b, c, d, dir, i, j;

        string s;

        while(cin >> n >> m)
        {

                if(n == 0 && m == 0) break;

                for(i = 1;i <= n;i++)
                {

                        for(j = 1;j <= m;j++)
                        {

                                cin >> matInit[i][j];
```

```
for(i = 1;i <= n;i++)
{

        for(j = 1;j <= m;j++)
        {

                x = 0;
                x += matInit[i][j];
                x += matInit[i][j + 1];
                x += matInit[i + 1][j];
                x += matInit[i + 1][j + 1];

                mat[i][j] = (x == 0);

        }

}
```

```
            n--;
            m--;

            cin >> a >> b >> c >> d >> s;

            if(s == "north")        dir = 0;
            if(s == "east")         dir = 1;
            if(s == "south")        dir = 2;
            if(s == "west")         dir = 3;

            BFS(a, b, dir);

            x = INF;
            x = min(x, dist[c][d][0]);
            x = min(x, dist[c][d][1]);
            x = min(x, dist[c][d][2]);
            x = min(x, dist[c][d][3]);

            if(x == INF) x = -1;

            cout << x << endl;

        }

    return 0;
```

Robot (versão 2)

```java
public class Grafo {
    int V;
    static boolean visited[];
    static int time[];
    ArrayList<Integer> adj[];

    public Grafo(int V) {
        this.V = V;
        adj = new ArrayList[V];
        visited = new boolean[V];
        time = new int[V];
        for (int i =0; i < V; i++){
            adj[i] = new ArrayList<>();
            time[i] = -1;
            visited[i] = false;
    }

    void adicionarAresta(int u, int v) {
        adj[u].add(v);
    }

    List<Integer> adj(int u) { return adj[u];}
```

```java
//achando o tempo minimo entre dois vértices (a e b)
public int bfs (int a, int b) {

    Queue<Integer> q = new LinkedList<Integer >() ;

    visited[a] = true;
    //peso[a] = 0;
    q.add(a);

    while (!q.isEmpty()) {
        int aux = q.remove() ;

        if (aux == b) {
            return time[aux];
        }

        for (int w : adj[aux]) {
            if (!visited[w]) {
                visited[w] = true;
                q.add(w);
                time[w] = time[aux] + 1;
            }
        }
    }
    // Se não houver caminho para o destino
    return -1;
```

Exemplos rodados

```
> sh -c make -s
> ./main
9 10
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 1 0
7 2 2 7 south
0 0
12
>
```