INSTITUTO MAUÁ DE TECNOLOGIA

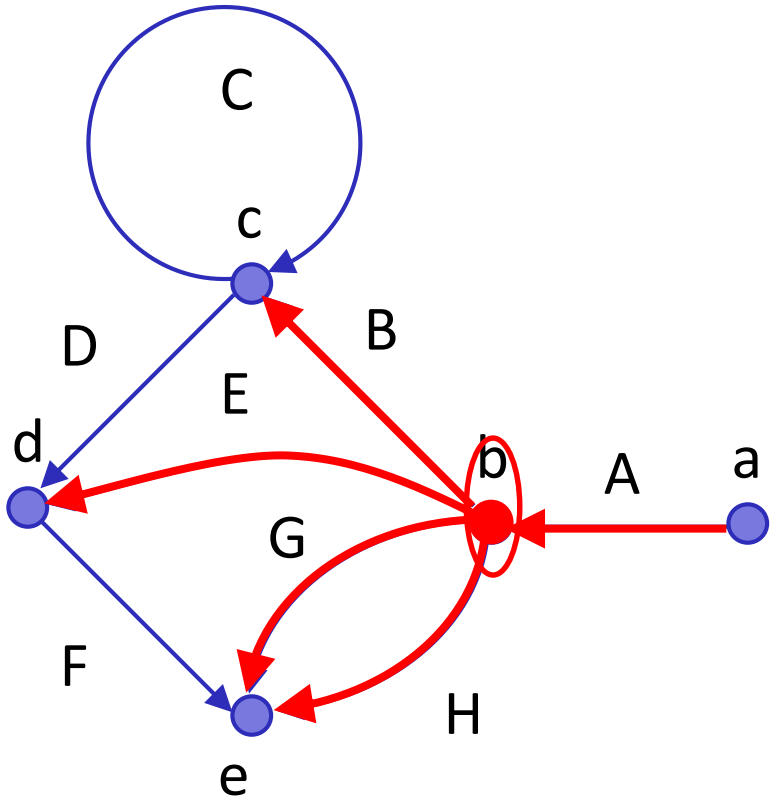
MAUÁ

16

Grafos

Biblioteca e Algoritmos

Grau de Entrada e de Saída



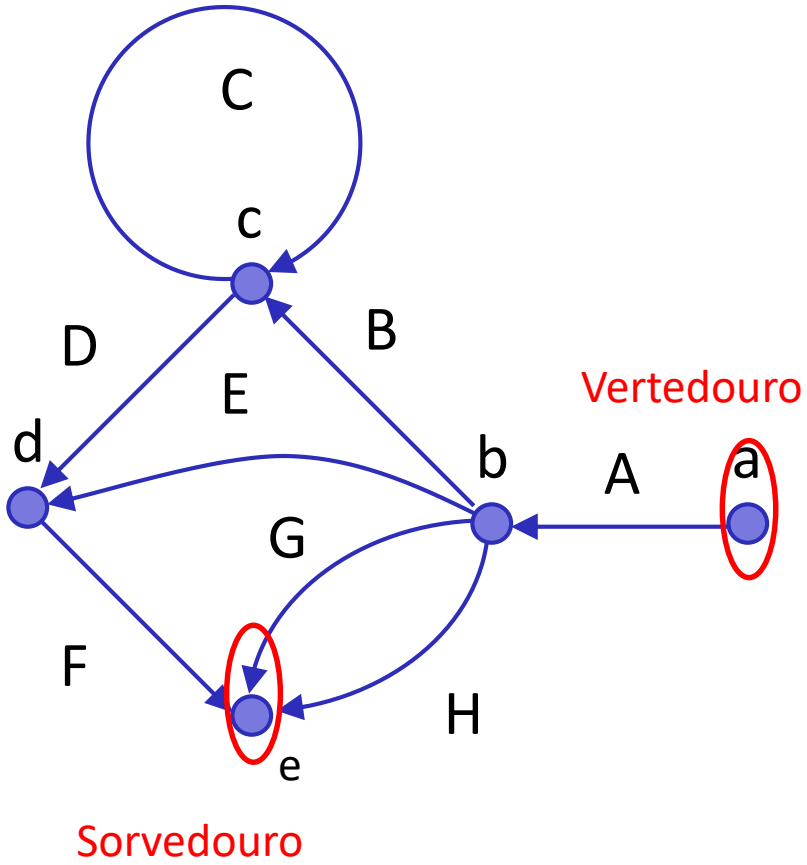
Matriz de adjacências

	a	b	c	d	e
a	0	1	0	0	0
b	0	0	1	1	2
c	0	0	1	1	0
d	0	0	0	0	1
e	0	0	0	0	0

Grau de saída = 4

Grau de entrada = 1

Vertedouro e Sorvedouro



Matriz de adjacências

	a	b	c	d	e
a	0	1	0	0	0
b	0	0	1	1	2
c	0	0	1	1	0
d	0	0	0	0	1
e	0	0	0	0	0

Grau de entrada = 0

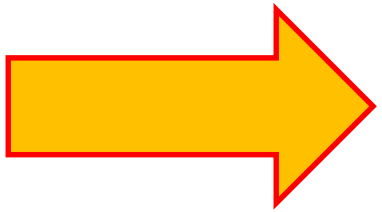
Grau de saída = 0

Permite verificar quais vértices podem ser alcançáveis;

Dígrafo

Matriz de adjacências

g	a	b	c	d	e
a	0	1	0	0	0
b	0	0	1	1	2
c	0	0	1	1	0
d	0	0	0	0	1
e	0	0	0	0	0



Matriz lógica de adjacências

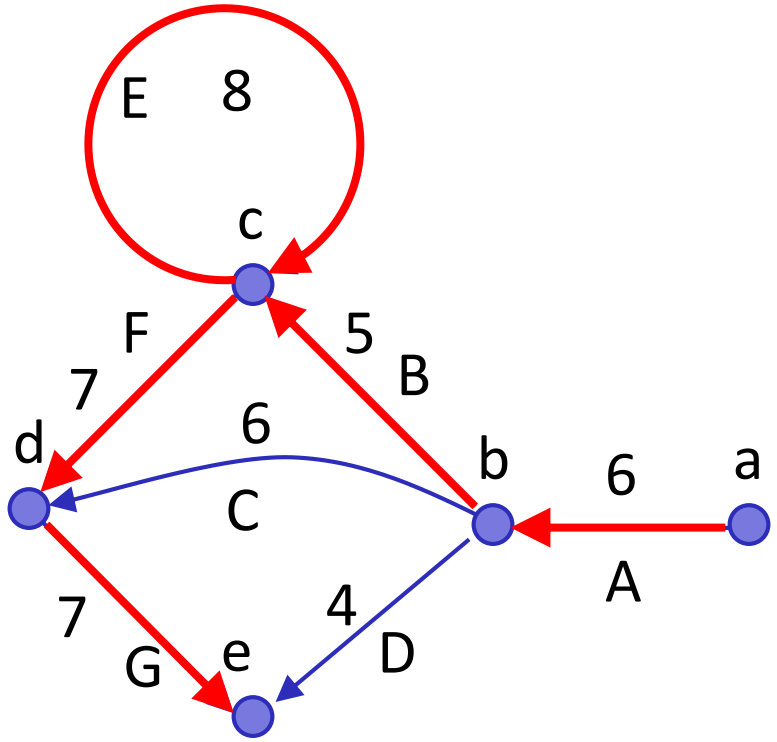
W	a	b	c	d	e
a	F	T	F	F	F
b	F	F	T	T	T
c	F	F	T	T	F
d	F	F	F	F	T
e	F	F	F	F	F

Algoritmo Warshall

```
Procedimento Warshall (Grafo g, int W[MAX_VERT][MAX_VERT])
/*
Entrada:
  g: Grafo com matriz de adjacências ou de pesos do dígrafo
Saída:
  W: a matriz "lógica" de vértices Alcançáveis
*/

{



para i percorrendo todos os vértices
  para j percorrendo todos os vértices
    W[i][j] ← PesoDaAresta(g, i, j) != 0;

para k percorrendo todos os vértices
 para i percorrendo todos os vértices
  para j percorrendo todos os vértices
   W[i][j] ← W[i][j] || (W[i][k] && W[k][j]);

}
```

W	a	b	c	d	e
a	F	T	T	T	T
b	F	F	T	T	T
c	F	F	T	T	T
d	F	F	F	F	T
e	F	F	F	F	F

Passeios



aAbBcEcFdGe = 33

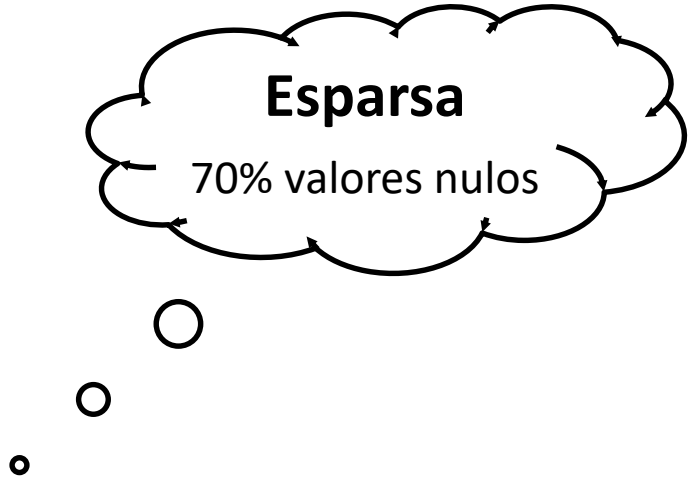
Matriz de pesos

	a	b	c	d	e
a	0	6	0	0	0
b	0	0	5	6	4
c	0	0	8	7	0
d	0	0	0	0	7
e	0	0	0	0	0

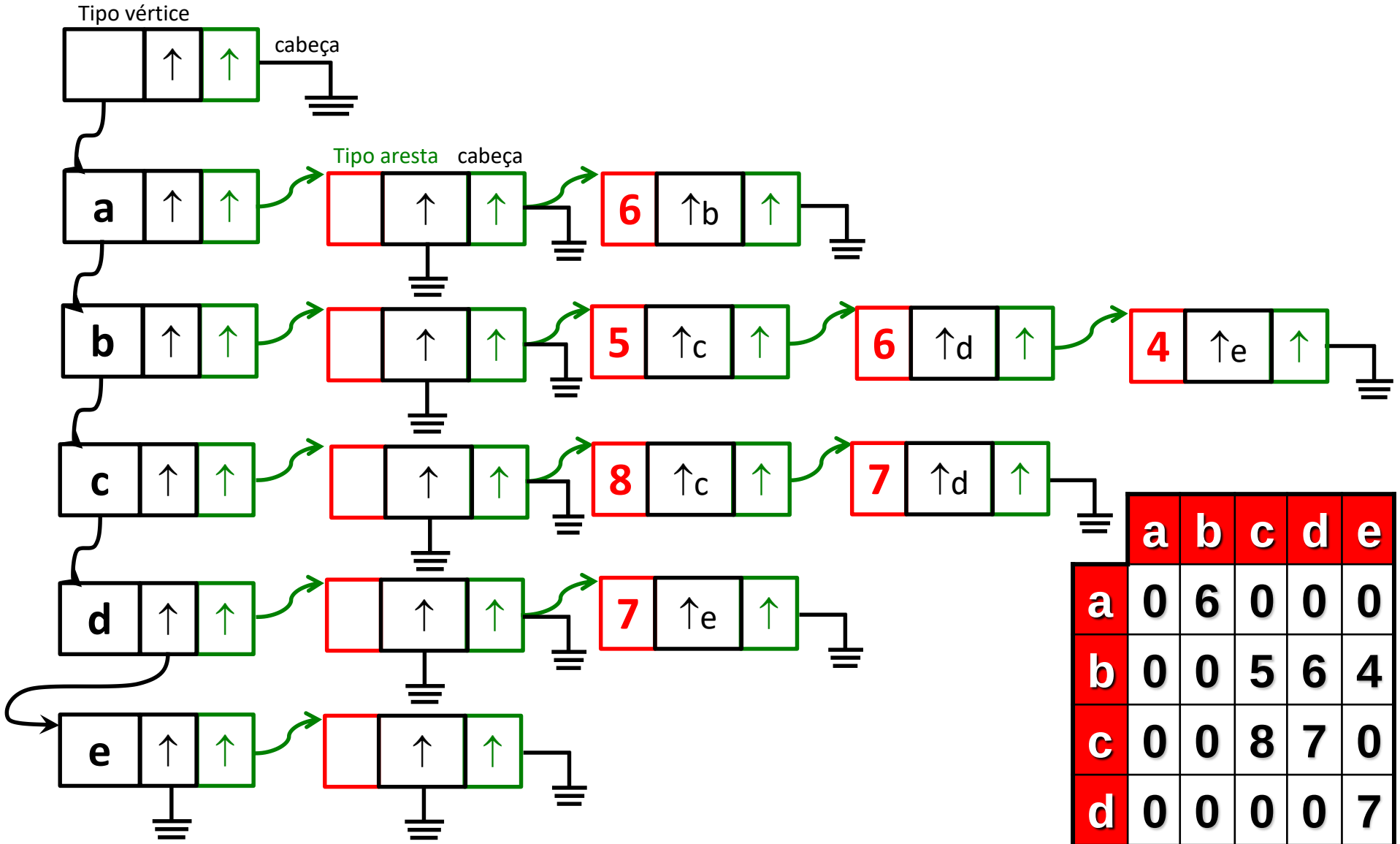
Evitar grande consumo de memória;

Memória: $O(n^2)$.

	a	b	c	d	e
a	0	6	0	0	0
b	0	0	5	6	4
c	0	0	8	7	0
d	0	0	0	0	7
e	0	0	0	0	0



Lista ligada

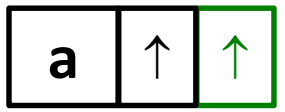


	a	b	c	d	e
a	0	6	0	0	0
b	0	0	5	6	4
c	0	0	8	7	0
d	0	0	0	0	7
e	0	0	0	0	0

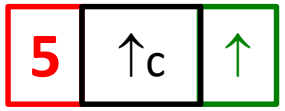
Lista ligada

```
typedef struct SVertice;
typedef struct SAresta;
```

```
typedef struct SVertice{
    int id;
    struct SVertice *proxVert;
    struct SAresta *proxAresta;
} TVertice;
```




```
typedef struct SAresta{
    float peso;
    struct SVertice *destino;
    struct SAresta *proxAresta;
} TAresta;
```



grafo.h

```
#ifndef GRAFO_H_INCLUDED
#define GRAFO_H_INCLUDED

#define MAX_VERT 100
#define MAX_VERT_LISTA 100


typedef struct
{
    float Pesos[MAX_VERT][MAX_VERT];
    int nVertices;
    int digrafo;
} Grafo;


typedef struct
{
    int vertices[MAX_VERT_LISTA];
    int nVertices;
} ListaDeVertices;
```

grafo.h

```
void CriarGrafo (Grafo *g, int n, int dig);
void InserirAresta (Grafo *g, int de, int para, float peso);
void RemoverAresta (Grafo *g, int de, int para);
float PesoDaAresta (Grafo g, int de, int para);

int GrauDeEntrada (Grafo g, int v);
int GrauDeSaida (Grafo g, int v);

int Vertedouro (Grafo g, int v);
int Sorvedouro (Grafo g, int v);

int Adjacente (Grafo g, int de, int para);

void Warshall (Grafo g, int w[MAX_VERT][MAX_VERT]);
int Alcanca (Grafo g, int de, int para);

float PesoDoPasseio (Grafo g, ListaDeVertices p);

#endif // GRAFO_H_INCLUDED
```

Passagem por valor:

g.

Passagem por referência:

(*g). ou **g->**