Buscas em grafos

Buscas em grafos

1. Busca em largura

Buscas em grafos

1. Busca em largura

2. Busca em profundidade

Busca em profundidade/Mz de adj.

Busca em profundidade/Mz de adj.

```
#define maxV 10000
static int cnt, pre[maxV];

#define dfsR search
void dfsR(Graph G, Edge e)
  { int t, w = e.w;
    pre[w] = cnt++;
    for (t = 0; t < G->V; t++)
      if (G->adj[w][t] != 0)
        if (pre[t] == -1)
          dfsR(G, EDGE(w, t));
  }
```

A chamada inicial

A chamada inicial

```
void GRAPHsearch(Graph G)
  { int v;
    cnt = 0;
    for (v = 0; v < G->V; v++) pre[v] = -1;
    for (v = 0; v < G->V; v++)
      if (pre[v] == -1)
        search(G, EDGE(v, v));
  }
```

Programa 18.3.driver

Programa 18.3.driver

```
#include <stdio.h>
#include <stdlib.h>
#include "GRAPH.h"
main(int argc, char *argv[])
  { int V = atoi(argv[1]), E = atoi(argv[2]);
    Graph G = GRAPHscan_noST(V, E);
    if (V < 20) GRAPHshow(G);
    else {
      printf("%d vertices, %d edges, ", V, E);
      printf("Too big!\n");
    }
    GRAPHsearch(G);
    GRAPHshow_pre(G);
  }
```

Exemplo

Exemplo

```
8 vertices, 10 edges
 0:  2  5  7
 1:  7
 2:  0  6
 3:  4  5
 4:  3  5  6  7
 5:  0  3  4
 6:  2  4
 7:  0  1  4
pre[]: 0 7 1 4 3 5 2 6
```

Busca em profundidade/Ls de adj.

Busca em profundidade/Ls de adj.

```
#define maxV 10000
static int cnt, pre[maxV];

#define dfsR search
void dfsR(Graph G, Edge e)
  { link t; int w = e.w;
    pre[w] = cnt++;
    for (t = G->adj[w]; t != NULL; t = t->next)
      if (pre[t->v] == -1)
        dfsR(G, EDGE(w, t->v));
  }
```

A chamada inicial: a mesma!

A chamada inicial: a mesma!

```
void GRAPHsearch(Graph G)
  { int v;
    cnt = 0;
    for (v = 0; v < G->V; v++) pre[v] = -1;
    for (v = 0; v < G->V; v++)
      if (pre[v] == -1)
        search(G, EDGE(v, v));
  }
```

Exemplo

Exemplo

8 vertices, 10 edges
0: 5 7 2
1: 7
2: 6 0
3: 5 4
4: 7 6 5 3
5: 4 3 0
6: 4 2
7: 4 1 0
pre[]: 0 4 6 7 2 1 5 3

v e w no mesmo componente?

v e w no mesmo componente?

```
struct graph { int V; int E; link *adj; int *cc;};

int GRAPHconnect(Graph G, int s, int t)
   { return G->cc[s] == G->cc[t]; }
```

v e w no mesmo componente? (Cont.)

v e w no mesmo componente? (Cont.)

```
void dfsRcc(Graph G, int v, int id)
  { link t;
    G->cc[v] = id;
    for (t = G->adj[v]; t != NULL; t = t->next)
      if (G->cc[t->v] == -1) dfsRcc(G, t->v, id);
  }
int GRAPHcc(Graph G)
  { int v, id = 0;
    G->cc = malloc(G->V * sizeof(int));
    for (v = 0; v < G->V; v++) G->cc[v] = -1;
    for (v = 0; v < G->V; v++)
      if (G->cc[v] == -1) dfsRcc(G, v, id++);
    return id;
  }
```

Exemplo

Exemplo

```
10 vertices, 10 edges
 0:  8  3  9  4
 1:  9
 2:  6  3
 3:  0  8  2
 4:  0
 5:  9  6
 6:  2  5
 7:
 8:  0  3
 9:  5  1  0
2 component(s)
```

Mais exemplos

Mais exemplos

```
yoshi@RANDOM ~/Main/www/2004i/mac328/exx/prog18.4
$ a 100 200
100 vertices, 200 edges, 3 component(s)

yoshi@RANDOM ~/Main/www/2004i/mac328/exx/prog18.4
$ a 100 223
100 vertices, 223 edges, 2 component(s)

yoshi@RANDOM ~/Main/www/2004i/mac328/exx/prog18.4
$ a 100 224
100 vertices, 224 edges, 1 component(s)

yoshi@RANDOM ~/Main/www/2004i/mac328/exx/prog18.4
$
```

Complexidade de tempo da busca em prof.

Complexidade de tempo da busca em prof.

- Matrizes de adjacência: $O(|V(G)|^2)$

Complexidade de tempo da busca em prof.

- Matrizes de adjacência: $O(|V(G)|^2)$
- Listas de adjacência: $O(|V(G)| + |E(G)|)$