

## Componentes fortemente conexos

## Componentes fortemente conexos

1. Subgrafos induzidos pela relação de equivalência  $x \leftrightarrow y$  (acessibilidade mútua)

## Componentes fortemente conexos

1. Subgrafos induzidos pela relação de equivalência  $x \leftrightarrow y$  (acessibilidade mútua)
2. Algoritmos para identificação de componentes fortemente conexos:

## Componentes fortemente conexos

1. Subgrafos induzidos pela relação de equivalência  $x \leftrightarrow y$  (acessibilidade mútua)
2. Algoritmos para identificação de componentes fortemente conexos:
  - ▷ Pela definição

## Componentes fortemente conexos

1. Subgrafos induzidos pela relação de equivalência  $x \leftrightarrow y$  (acessibilidade mútua)
2. Algoritmos para identificação de componentes fortemente conexos:
  - ▷ Pela definição
  - ▷ Algoritmo de Kosaraju (década de 80)

## Componentes fortemente conexos

1. Subgrafos induzidos pela relação de equivalência  $x \leftrightarrow y$  (acessibilidade mútua)
2. Algoritmos para identificação de componentes fortemente conexos:
  - ▷ Pela definição
  - ▷ Algoritmo de Kosaraju (década de 80) [ $O(n + m)$ ]

## Componentes fortemente conexos

1. Subgrafos induzidos pela relação de equivalência  $x \leftrightarrow y$  (acessibilidade mútua)
2. Algoritmos para identificação de componentes fortemente conexos:
  - ▷ Pela definição
  - ▷ Algoritmo de Kosaraju (década de 80) [ $O(n + m)$ ]
  - ▷ Algoritmo de Tarjan (1972)

## Componentes fortemente conexos

1. Subgrafos induzidos pela relação de equivalência  $x \leftrightarrow y$  (acessibilidade mútua)
2. Algoritmos para identificação de componentes fortemente conexos:
  - ▷ Pela definição
  - ▷ Algoritmo de Kosaraju (década de 80) [ $O(n + m)$ ]
  - ▷ Algoritmo de Tarjan (1972) [ $O(n + m)$ ]

## Componentes fortemente conexos

1. Subgrafos induzidos pela relação de equivalência  $x \leftrightarrow y$  (acessibilidade mútua)
2. Algoritmos para identificação de componentes fortemente conexos:
  - ▷ Pela definição
  - ▷ Algoritmo de Kosaraju (década de 80) [ $O(n + m)$ ]
  - ▷ Algoritmo de Tarjan (1972) [ $O(n + m)$ ]
  - ▷ Algoritmo de Gabow (1999)

## Componentes fortemente conexos

1. Subgrafos induzidos pela relação de equivalência  $x \leftrightarrow y$  (acessibilidade mútua)
2. Algoritmos para identificação de componentes fortemente conexos:
  - ▷ Pela definição
  - ▷ Algoritmo de Kosaraju (década de 80) [ $O(n + m)$ ]
  - ▷ Algoritmo de Tarjan (1972) [ $O(n + m)$ ]
  - ▷ Algoritmo de Gabow (1999) [ $O(n + m)$ ]

## Algoritmo de Kosaraju

## Algoritmo de Kosaraju

1. Invertemos  $G$

## Algoritmo de Kosaraju

1. Invertemos  $G$
2. Executamos uma BeP e determinamos  $\text{post}[]$

## Algoritmo de Kosaraju

1. Invertemos  $G$
2. Executamos uma BeP e determinamos `post []`
3. Executamos uma BeP na ordem dada por `post []` *invertida*

## Programa 19.10, Algoritmo de Kosaraju

## Programa 19.10, Algoritmo de Kosaraju

```
static int post[maxV], postR[maxV];
static int cnt0, cnt1;
void SCdfsR(Graph G, int w)
{ link t;
  G->sc[w] = cnt1;
  for (t = G->adj[w]; t != NULL; t = t->next)
    if (G->sc[t->v] == -1) SCdfsR(G, t->v);
  post[cnt0++] = w;
}
```

## Programa 19.10, Algoritmo de Kosaraju (cont.)

## Programa 19.10, Algoritmo de Kosaraju (cont.)

```
int GRAPHsc(Graph G)
{ int v; Graph R = GRAPHreverse(G);
  cnt0 = 0; cnt1 = 0;
  for (v = 0; v < G->V; v++) R->sc[v] = -1;
  for (v = 0; v < G->V; v++) if (R->sc[v] == -1) SCdfsR(R, v);
  cnt0 = 0; cnt1 = 0;
  for (v = 0; v < G->V; v++) G->sc[v] = -1;
  for (v = 0; v < G->V; v++) postR[v] = post[v];
  for (v = G->V-1; v >=0; v--)
    if (G->sc[postR[v]] == -1)
      { SCdfsR(G, postR[v]); cnt1++; }
  GRAPHdestroy(R);
  return cnt1;
}
```

## Programa 19.10, Algoritmo de Kosaraju (cont.)

## Programa 19.10, Algoritmo de Kosaraju (cont.)

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

Exemplo; grafo da Figura 19.28

## Exemplo; grafo da Figura 19.28

13 vertices, 22 arcs

```
0:  6  1  5  0
1:  1
2:  3  0  2
3:  2  5  3
4:  2 11  3  4
5:  4  5
6:  4  9  6
7:  8  6  7
8:  9  7  8
9: 10 11  9
10: 12 10
11: 12 11
12:  9 12
```

Exemplo, Figura 19.28; Reverso de  $G$

## Exemplo, Figura 19.28; Reverso de $G$

|           |            |
|-----------|------------|
| 0-0 tree  | 1-1 tree   |
| 0-2 tree  | 1-0 cross  |
| 2-4 tree  | 9-9 tree   |
| 4-6 tree  | 9-12 tree  |
| 6-7 tree  | 12-11 tree |
| 7-8 tree  | 11-9 back  |
| 8-7 back  | 11-4 cross |
| 6-0 cross | 12-10 tree |
| 4-5 tree  | 10-9 cross |
| 5-3 tree  | 9-8 cross  |
| 3-4 back  | 9-6 cross  |
| 3-2 cross |            |
| 5-0 cross |            |
| 2-3 down  |            |

Exemplo, Figura 19.28

## Exemplo, Figura 19.28

|         |   |   |   |   |   |   |   |   |   |    |    |    |    |
|---------|---|---|---|---|---|---|---|---|---|----|----|----|----|
| v:      | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  | 10 | 11 | 12 |
| pre[]:  | 0 | 8 | 1 | 7 | 2 | 6 | 3 | 4 | 5 | 9  | 12 | 11 | 10 |
| post[]: | 8 | 7 | 6 | 3 | 5 | 4 | 2 | 0 | 1 | 11 | 10 | 12 | 9  |
| st[]:   | 0 | 1 | 0 | 5 | 2 | 4 | 4 | 6 | 7 | 9  | 12 | 12 | 9  |

## Exemplo, Figura 19.28

## Exemplo, Figura 19.28

|             |            |
|-------------|------------|
| 9-9 tree    | 5-4 cross  |
| 9-10 tree   | 2-0 cross  |
| 10-12 tree  | 4-11 cross |
| 12-9 cross  | 4-3 down   |
| 9-11 tree   | 6-9 cross  |
| 11-12 cross | 0-1 cross  |
| 1-1 tree    | 0-5 down   |
| 0-0 tree    | 7-7 tree   |
| 0-6 tree    | 7-8 tree   |
| 6-4 tree    | 8-9 cross  |
| 4-2 tree    | 8-7 cross  |
| 2-3 tree    | 7-6 cross  |
| 3-2 cross   |            |
| 3-5 tree    |            |

## Algoritmo de Kosaraju

## Algoritmo de Kosaraju

**Propriedade 1 (Propriedade 19.14).** *O método de Kosaraju encontra os componentes fortemente conexos de  $G$  em tempo  $O(n + m)$ .*

## Algoritmo de Kosaraju

**Propriedade 1 (Propriedade 19.14).** *O método de Kosaraju encontra os componentes fortemente conexos de  $G$  em tempo  $O(n + m)$ .*

*Demonstração. (...)*



## Algoritmo de Tarjan, Programa 19.11

## Algoritmo de Tarjan, Programa 19.11

```
void SCdfsR(Graph G, int w)
{ link t; int v, min;
  pre[w] = cnt0++; low[w] = pre[w]; min = low[w];
  s[N++] = w;
  for (t = G->adj[w]; t != NULL; t = t->next) {
    if (pre[t->v] == -1) SCdfsR(G, t->v);
    if (low[t->v] < min) min = low[t->v];
  }
  if (min < low[w]) { low[w] = min; return; }
  do
    { G->sc[(v = s[--N])] = cnt1; low[v] = G->V; }
  while (s[N] != w);
  cnt1++;
}
```

## Algoritmo de Tarjan; Exemplo da Fig. 19.29

## Algoritmo de Tarjan; Exemplo da Fig. 19.29

13 vertices, 22 edges

```
0:  5  1  6  0
1:  1
2:  0  3  2
3:  5  2  3
4:  3 11  2  4
5:  4  5
6:  9  4  6
7:  6  8  7
8:  7  9  8
9: 11 10  9
10: 12 10
11: 12 11
12:  9 12
```

## Algoritmo de Tarjan; Exemplo da Fig. 19.29

## Algoritmo de Tarjan; Exemplo da Fig. 19.29

```
0-0 tree
  0-5 tree
    5-4 tree
      4-3 tree
        3-5 back
        3-2 tree
          2-0 back
          2-3 back
        4-11 tree
          11-12 tree
            12-9 tree
              9-11 back
              9-10 tree
                10-12 back
              4-2 down
            0-1 tree
            0-6 tree
              6-9 back
              6-4 cross
            7-7 tree
              7-6 cross
              7-8 tree
                8-7 cross
                8-9 cross
```

Algoritmo de Tarjan; Exemplo da Fig. 19.29

## Algoritmo de Tarjan; Exemplo da Fig. 19.29

|         |   |   |   |   |   |   |    |    |    |    |   |   |    |
|---------|---|---|---|---|---|---|----|----|----|----|---|---|----|
| pre[] : | 0 | 9 | 4 | 3 | 2 | 1 | 10 | 11 | 12 | 7  | 8 | 5 | 6  |
| low[] : | 0 | 9 | 0 | 0 | 0 | 0 | 0  | 11 | 11 | 5  | 6 | 5 | 5  |
| st[] :  | 0 | 0 | 3 | 4 | 5 | 0 | 0  | 7  | 7  | 12 | 9 | 4 | 11 |
| sc[] :  | 2 | 1 | 2 | 2 | 2 | 2 | 2  | 3  | 3  | 0  | 0 | 0 | 0  |

## Algoritmo de Tarjan

## Algoritmo de Tarjan

**Propriedade 2 (Propriedade 19.15).** *O método de Tarjan encontra os componentes fortemente conexos de  $G$  em tempo  $O(n + m)$ .*

## Algoritmo de Tarjan

**Propriedade 2 (Propriedade 19.15).** *O método de Tarjan encontra os componentes fortemente conexos de  $G$  em tempo  $O(n + m)$ .*

*Demonstração. (...)*



## O Tarjan no SGB

## O Tarjan no SGB

▷ gb\_roget

▷ roget\_components

## Algoritmo de Gabow, Programa 19.12

## Algoritmo de Gabow, Programa 19.12

```
void SCdfsR(Graph G, int w)
{ link t; int v;
  pre[w] = cnt0++;
  s[N++] = w; path[p++] = w;
  for (t = G->adj[w]; t != NULL; t = t->next)
    if (pre[t->v] == -1) SCdfsR(G, t->v);
    else if (G->sc[t->v] == -1)
      while (pre[path[p-1]] > pre[t->v]) p--;
  if (path[p-1] != w) return; else p--;
  do G->sc[s[--N]] = cnt1; while (s[N] != w);
  cnt1++;
}
```

## Algoritmo de Gabow

## Algoritmo de Gabow

**Propriedade 3 (Propriedade 19.16).** *O método de Gabow encontra os componentes fortemente conexos de  $G$  em tempo  $O(n + m)$ .*

## Algoritmo de Gabow

**Propriedade 3 (Propriedade 19.16).** *O método de Gabow encontra os componentes fortemente conexos de  $G$  em tempo  $O(n + m)$ .*

*Demonstração. (...)*

