Busca em Profundidade para Grafos Dirigidos

```
void dfsR(Graph G, Edge e)
  { link t; int v, w = e.w; Edge x;
    show("tree", e);  st[e.w]=e.v;
    pre[w] = cnt++;
    for (t = G->adj[w]; t != NULL; t = t->next)
      if (pre[t->v] == -1) dfsR(G, EDGE(w, t->v));
      else
        { v = t->v; x = EDGE(w, v);
          if (post[v] == -1) show("back", x);
          else if (pre[v] > pre[w]) show("down", x);
          else show("cross", x);
        }
    post[w] = cntP++;
  }
```

Busca em Profundidade para Grafos Dirigidos, Exemplo

```
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
```

Busca em Profundidade para Grafos Dirigidos, Exemplo

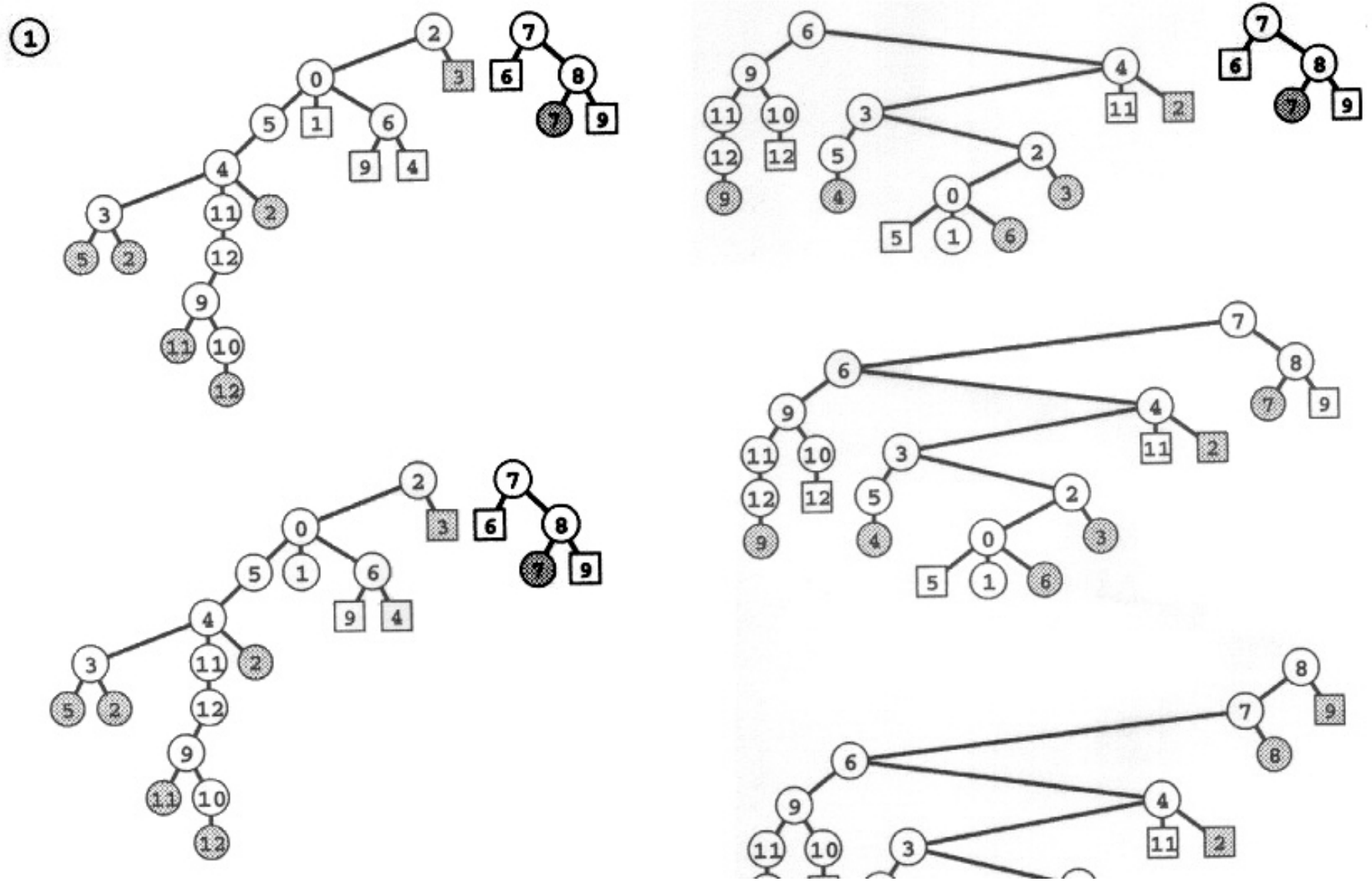
```
0-0 tree
  0-5 tree
    5-4 tree
      4-3 tree
        3-5 back
        3-2 tree
          2-0 back
          2-3 back
      4-11 tree
        11-12 tree
          12-9 tree
            9-11 back
            9-10 tree
              10-12 back
            4-2 down
      0-1 tree
      0-6 tree
 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

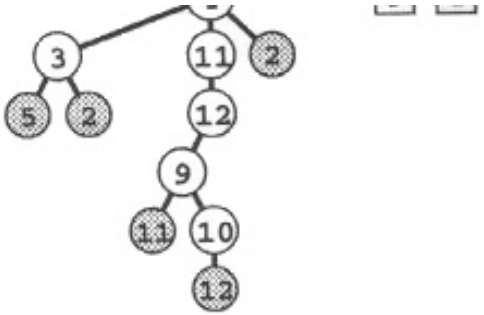
Busca em Profundidade para Grafos Dirigidos, Exemplo

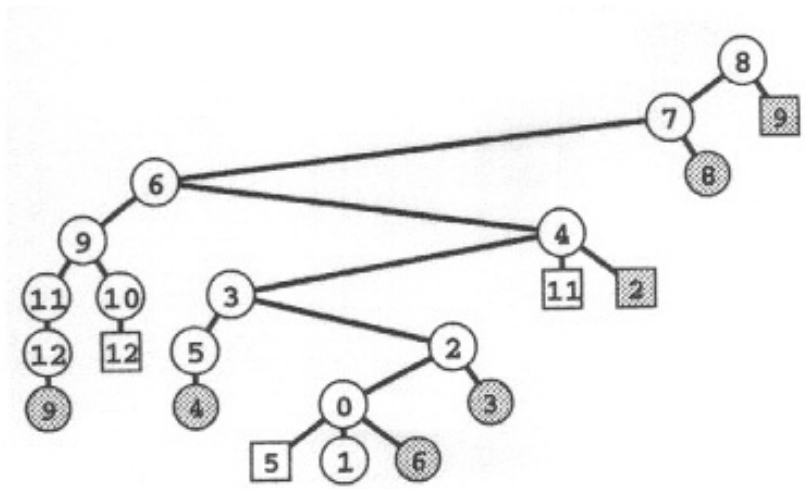
v:	0	1	2	3	4	5	6	7	8	9	10	11	12
pre[]:	0	9	4	3	2	1	10	11	12	7	8	5	6
post[]:	10	8	0	1	6	7	9	12	11	3	2	5	4
st[]:	0	0	3	4	5	0	0	7	7	12	9	4	11

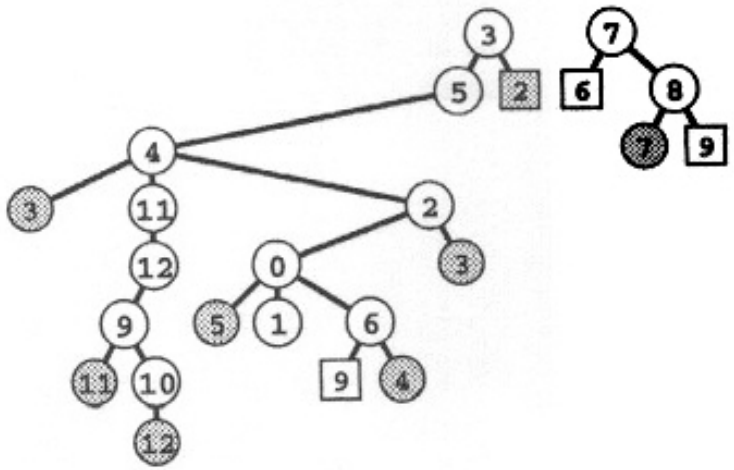
Classificação dos arcos

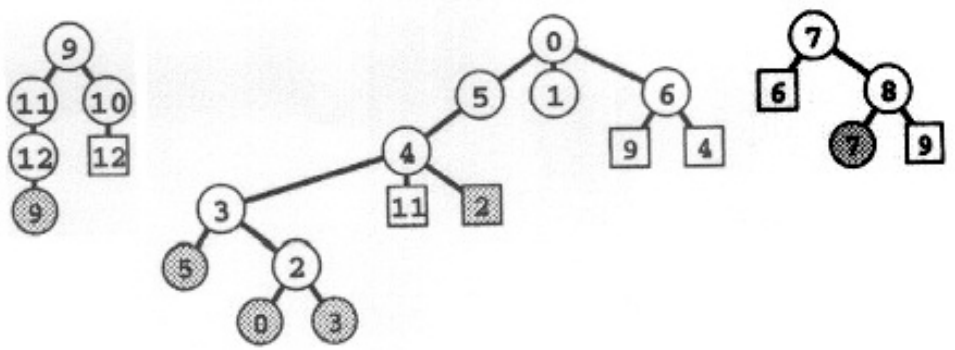
```
e = (v, w)

pre post exemplo tipo
 <    >      4-2   down
 >    <      2-0   back
 >    >      7-6   cross
```

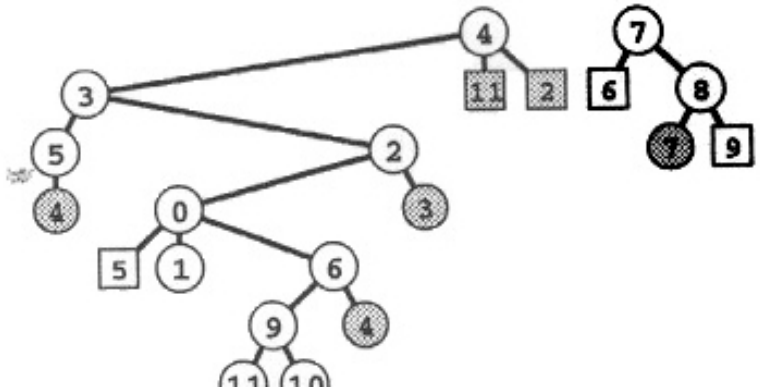

1

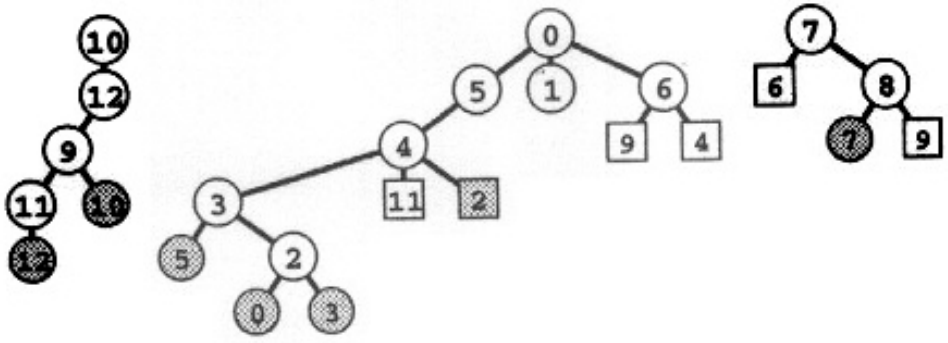


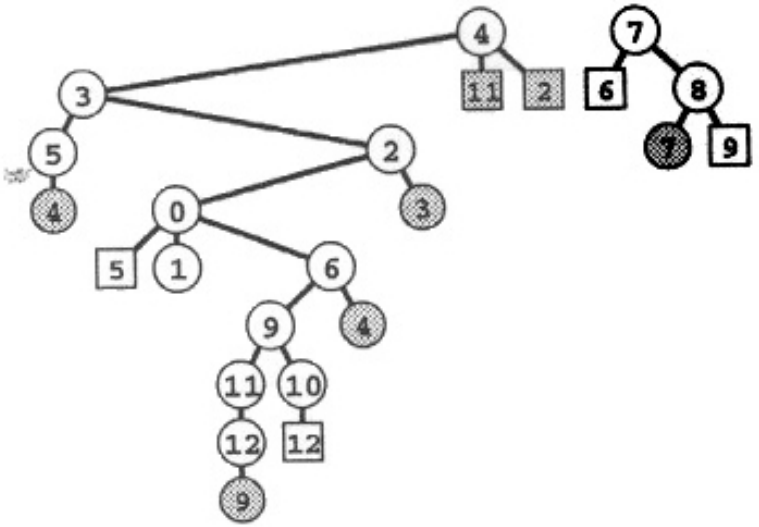


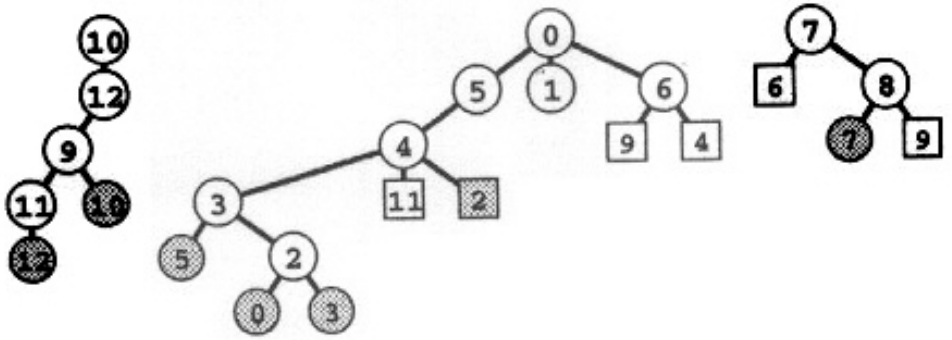


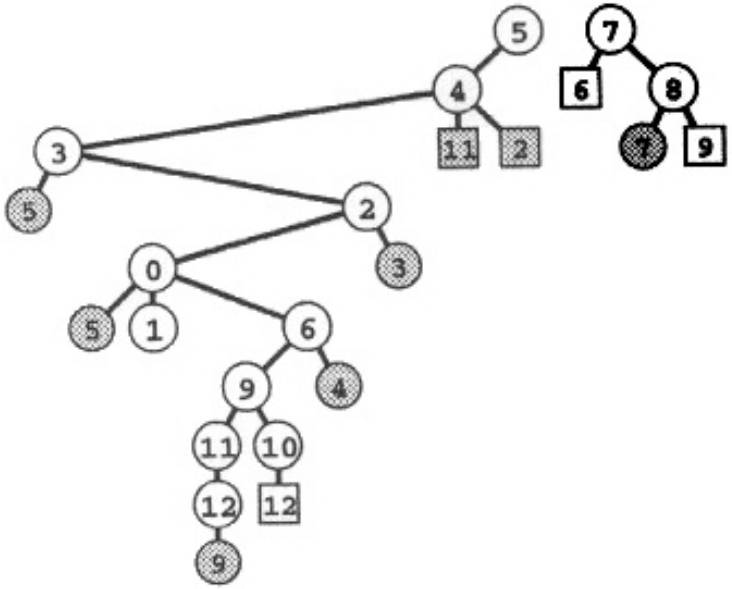


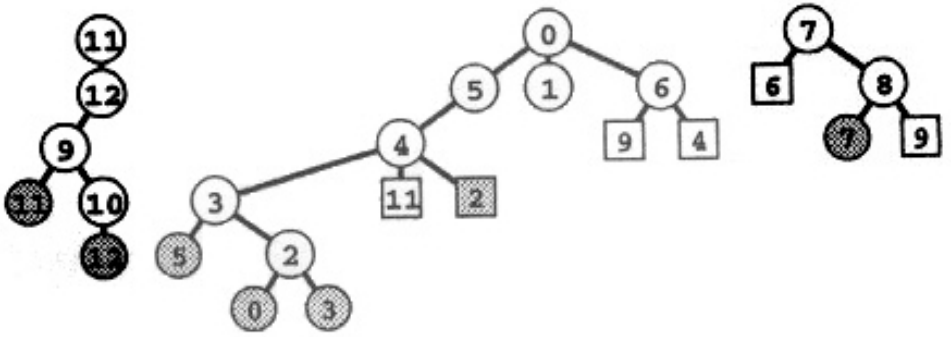


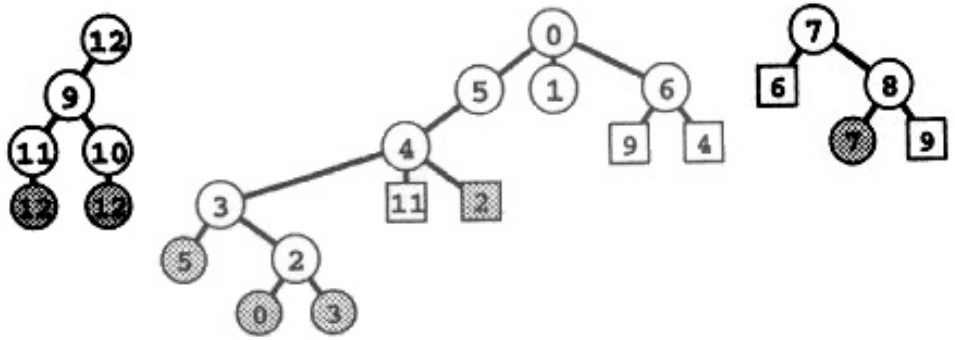












Busca em Profundidade para Grafos Dirigidos, Exemplo

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

Busca em Profundidade para Grafos Dirigidos, Exemplo

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

Busca em Profundidade para Grafos Dirigidos, Exemplo

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

Busca em Profundidade para Grafos Dirigidos, Exemplo

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

Propriedade dos parênteses. . .

Algoritmos baseados em BeP

1. Detecção de circuitos Dirigidos

2. Acessibilidade a partir de um vértice

Detecção de circuitos Dirigidos

Propriedade 19.4 Um grafo dirigido G é um DAG se e só se back arcs não ocorrem.

Acessibilidade a partir de um vértice

Propriedade 19.5 Uma busca em profundidade disparada a partir de um vértice s alcança todos os vértices acessíveis de s em tempo $O(|E_s|)$, onde E_s é o conjunto de arcos do grafo induzido pelos vértice acessíveis a partir de s.