UNIFEI - UNIVERSIDADE FEDERAL DE ITAJUBÁ
CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

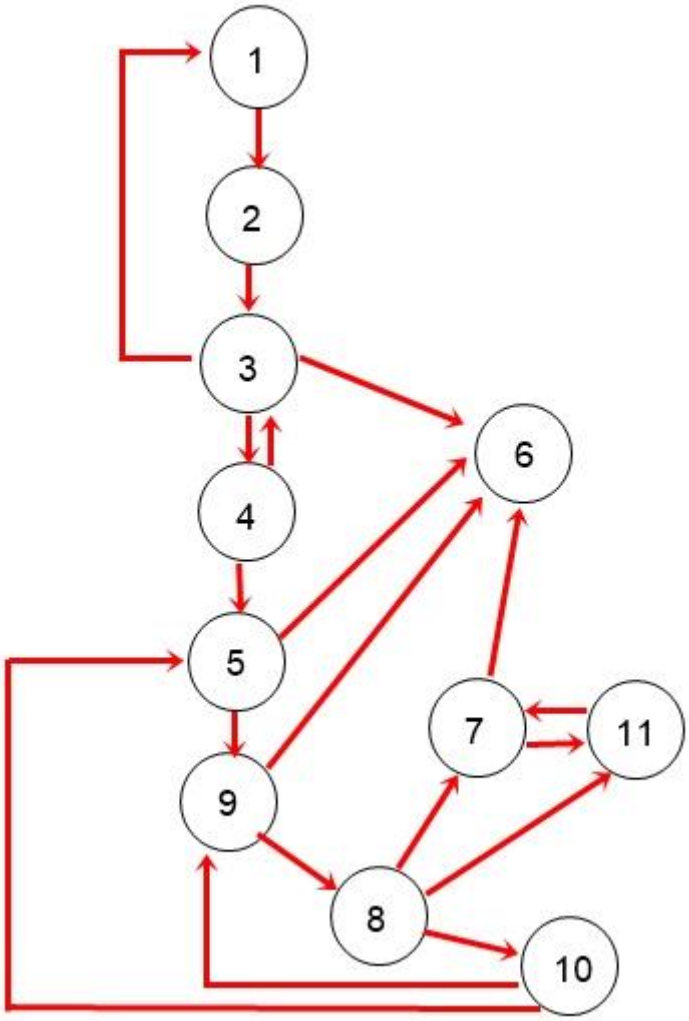
SIN110 - ALGORITMOS E GRAFOS
RESOLUÇÃO DOS EXERCÍCIOS E04 DO DIA 11/09/2015

ITAJUBÁ
2015

Exercícios E04 – 11/09/15

Aluna: Karen Dantas **Número de matrícula:** 31243

1) Dígrafo G_2:

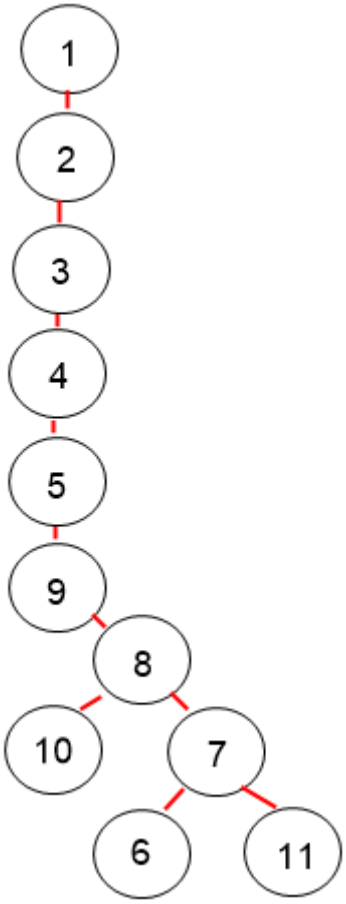


a) Solução:

Busca em profundidade:

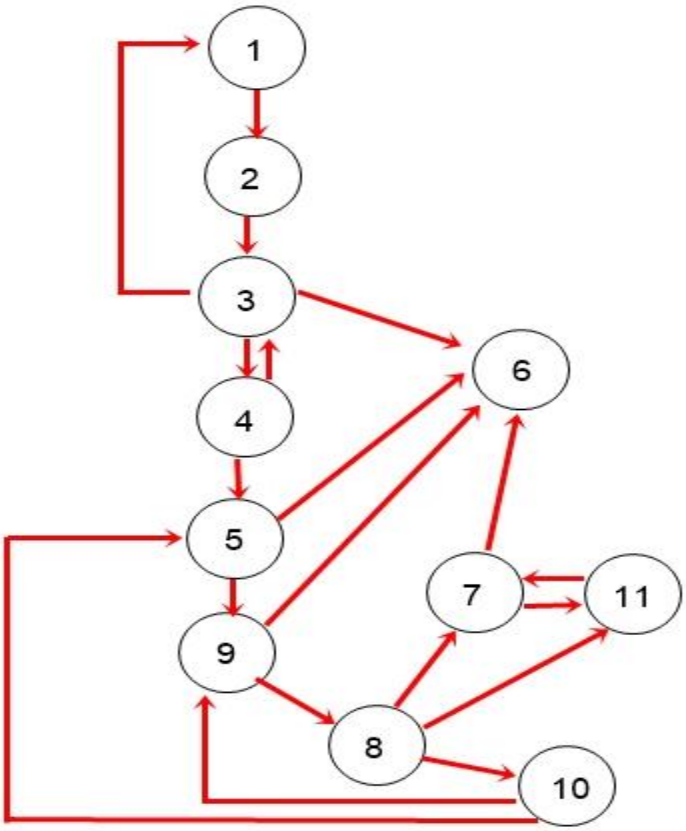
Vértice	Cor(u)	Predecessor(u)	Descoberta (u)	Fim (u)
1	~~b~~ /~~c~~ /~~p~~	-	1	22
2	~~b~~ /~~c~~ /~~p~~	1	2	21
3	~~b~~ /~~c~~ /~~p~~	2	3	20
4	~~b~~ /~~c~~ /~~p~~	3	4	19
5	~~b~~ /~~c~~ /~~p~~	4	5	18
6	~~b~~ /~~c~~ /~~p~~	7	11	12
7	~~b~~ /~~c~~ /~~p~~	8	10	15
8	~~b~~ /~~c~~ /~~p~~	9	7	16
9	~~b~~ /~~c~~ /~~p~~	5	6	17
10	~~b~~ /~~c~~ /~~p~~	8	8	9
11	~~b~~ /~~c~~ /~~p~~	7	13	14

Floresta de busca em profundidade:



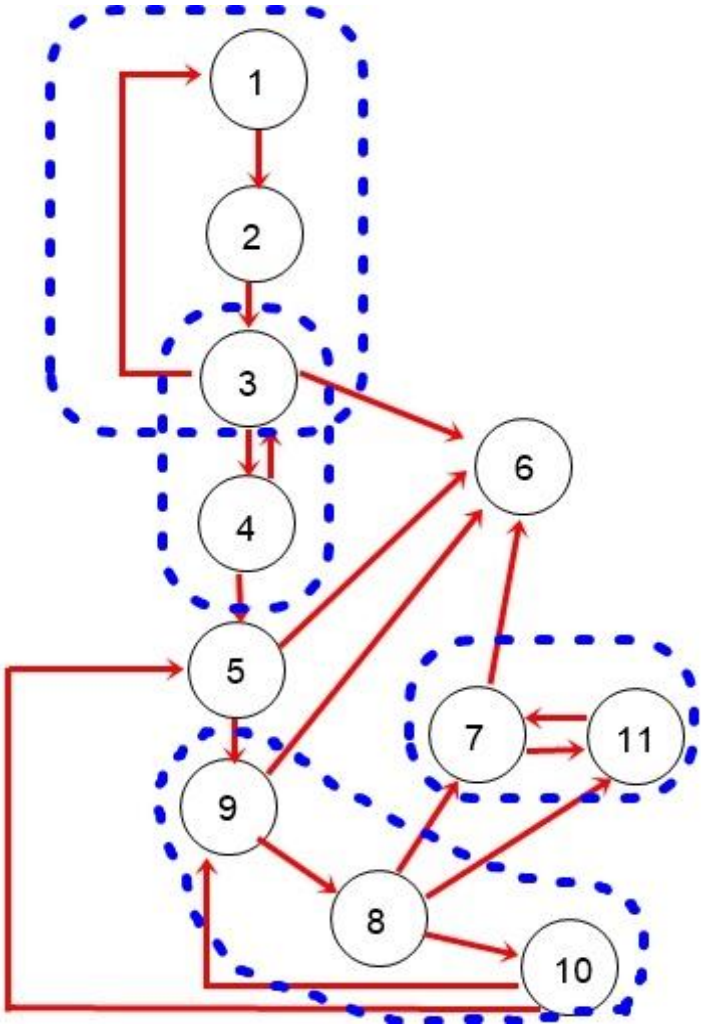
b) <u>Solução:</u>

No dígrafo G_2 há apenas um componente. E ele é composto por todos os vértices do grafo e pode ser representado pelo próprio grafo.



c) Solução:

Sim, existem 4 ciclos nesse dígrafo. São eles: 1-2-3-1; 3-4-3; 9-8-10-9; 7-11-7. Na figura abaixo eles foram circulados em azul.



Função de pesquisa em ciclos que mostra todos os ciclos:

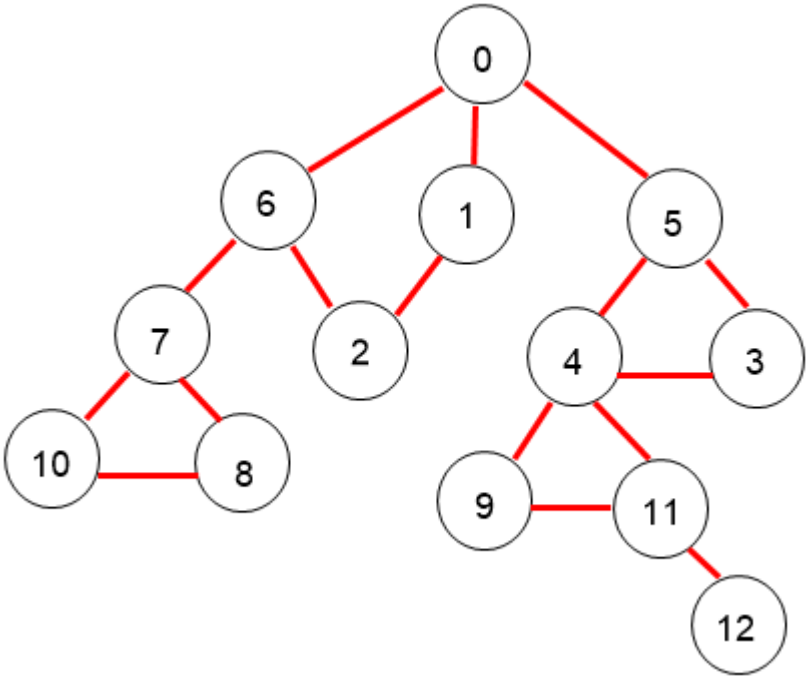
Ciclos-em-digrafos(G)

```
1.      contador<-0
2.       para cada vertice u em G faca
3.              para cada v em Adj(u) faca
4.                      ciclo <- Caminho(G, v, u)
5.               se ciclo = 1
6.                       entao contador <- contador+1
7.       devolve contador
```

2) Solução:

Lista de Adjacência:
0->6->1->5
1->2
2->6
3
4->11->9->3
5->3->4
6->7
7->8->10
8
9->11
10->8
11->12
12

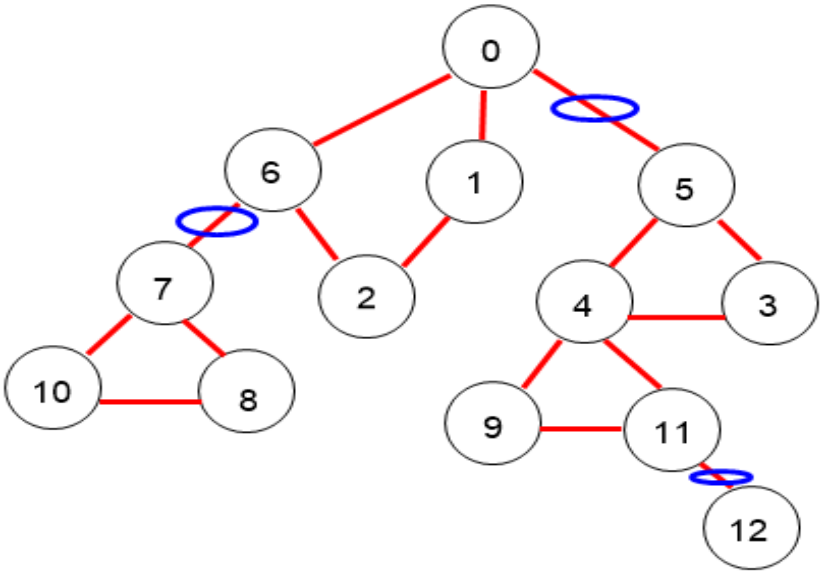
Grafo:



Pontes:

Uma aresta é dita ser um a ponte se sua remoção provoca uma redução na conexidade do grafo. Durante a execução do algoritmo de pesquisa pontes foram imprimidas as seguintes mensagens:
ponte: 0 - 5
ponte: 6 - 7
ponte: 11 - 12

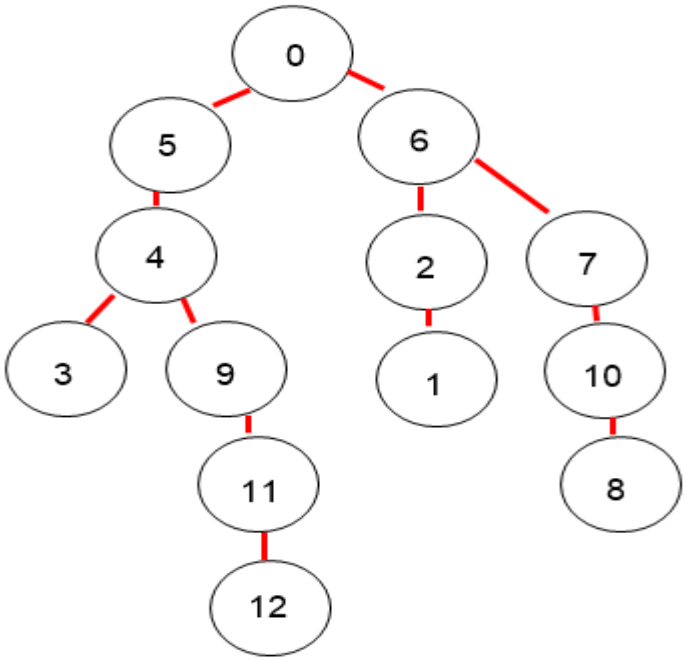
Logo, nesse grafo foram identificadas três pontes, que foram circuladas em azul na figura abaixo:



Busca em profundidade:

Vértice	Cor(u)	Predecessor(u)	Descoberta (u)	Fim (u)
0	~~b~~ /~~c~~ /~~p~~	-	1	26
1	~~b~~ /~~c~~ /~~p~~	2	10	11
2	~~b~~ /~~c~~ /~~p~~	6	9	12
3	~~b~~ /~~c~~ /~~p~~	4	22	23
4	~~b~~ /~~c~~ /~~p~~	5	15	24
5	~~b~~ /~~c~~ /~~p~~	0	14	25
6	~~b~~ /~~c~~ /~~p~~	0	2	13
7	~~b~~ /~~c~~ /~~p~~	6	3	8
8	~~b~~ /~~c~~ /~~p~~	10	5	6
9	~~b~~ /~~c~~ /~~p~~	4	16	21
10	~~b~~ /~~c~~ /~~p~~	7	4	7
11	~~b~~ /~~c~~ /~~p~~	9	17	20
12	~~b~~ /~~c~~ /~~p~~	11	18	19

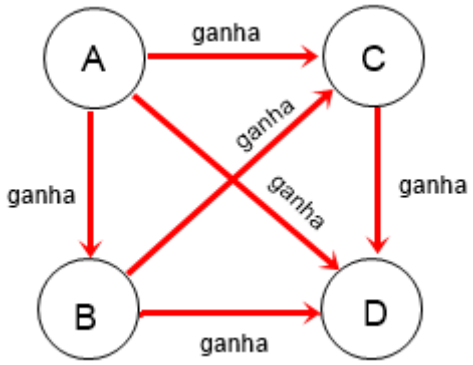
Arborescência:



3) Solução:

Campeonato de futebol: não pode ter empate e pode haver apenas uma partida entre cada par de times.

Para resolver esse problema montei um grafo de um campeonato com 4 times participantes, ele pode ser visto na figura abaixo.



Nele é possível perceber que o campeonato é persistente pois todos jogaram contra todos e não houve empates. Os resultados dos jogos foram:

A ganhou de B, C e D;

B ganhou de C e de D mas perdeu para A;

C ganhou de D mas perdeu para A e B;

D perdeu para A, B e C.

Logo, o time A ganhou o campeonato.

Algoritmo:

```
ValidaCampeonato (Times)
    i<-1
    para cada t em Times faça
        se (JaJogou(adjs(t), t) = 0))
            Resultados(i) <- partidas(t)
            i<-i+1
    devolve encontra_vencedor_campeonato(Resultados)
```

O algoritmo recebe o grafo como parâmetro e, para cada vértice do grafo, verifica se o vértice atual já jogou com todos os seus adjacentes senão jogou chama-se a função partida que verificará com quais adjacentes falta o vértice atual jogar e são realizadas as partidas. São retornados os resultados da partida que são armazenados em um índice do vetor Resultados. No final é retornado o vencedor do campeonato através da função encontra_vencedor_campeonato. Esse algoritmo garante que um time jogará com os outros apenas uma vez e que não haverá empates, ou seja, sempre haverá um vencedor.