UNIVERSIDADE FEDERAL DE SÃO PAULO
CAMPUS SÃO JOSÉ DOS CAMPOS
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA (DCT)

Exercício de Teoria dos Grafos

UC: **Teoria dos Grafos**

Aluno: **Thauany Moedano**

RA: **92486**

Professor: **Dr. Reginaldo Massanobu Kuroshu.**

Entrega: **16/08/2016**

Resumo

Resolução de exercícios (3) de Teoria dos Grafos

Exercício 1. Seja G um grafo conexo G tal que m(G) = n(G) - 1. Prove que G é uma árvore.

Basta mostrar que uma árvore com n $\geq$ 1 vértices possui n-1 arestas. Podemos mostrar por indução:

Caso base: n = 1 A única árvore com 1 vértice é um vértice isolado. Número de arestas m = 1-1 = 0

Hipótese Indutiva: Supondo que o torema vale para n vértices de uma árvore G, provar que vale para n-1 vértices.

Passo Indutivo: Selecione um vértice v que seja folha. A remoção deste vértice cria um novo grafo G'. G' é conexo e acíclico. Por hipótese, G' tem n-2 arestas. Mas como v é folha, sua remoção implica na retirada de apenas uma aresta. Pela fórmula: (n-2)+1 = n-1 arestas.

Exercício 2. Forneça um algoritmo com complexidade de tempo O(V) que determine se um dado grafo não direcionado G = (V,E) contém um ciclo.

```
int main() {
   int N, A, B, M; //N numero de nos, A,B arestas a serem inseridas, M numero
      de arestas
   int array_D[N]; //vetor que guarda o grau dos vertices
   int ehCiclo = 0;


```

```
    for(i = 0; i < M; i++) {
        cin >> A;
        cin >> B;
        array_D[A]++;
        array_D[B]++;

    }

    for(i = 0; i < N; i++) {
        if(array_D[i] >= 2) {
            ehCiclo = 1;
            break;
        }
    }
    if(ehCiclo)
        cout << "Contem ciclo" << endl;
    else
        cout << "Nao contem ciclo" << endl;

}
```

Exercício 3. Seja G um grafo k-regular bipartido com k>=2. Mostre que G não tem aresta de corte.

Um grafo bipartido é um grafo conexo pois pode ser dividido em um conjunto X e Y com uma aresta em uma ponta em X e outra em Y. Cada vértice têm necessariamente vizinhos em um conjunto oposto. Se retirarmos uma aresta de um vértice qualquer, teremos um vértice com k=1 e ainda com uma aresta que tem ponta no conjunto X e outra em um conjunto Y, portanto conexo. Como isso ocorre para todos os vértices do grafo, G não tem aresta de corte.

Exercício 4. Seja G um grafo conexo e T uma árvore de busca em profundidade de G, onde todas as arestas sao orientadas do pai para o filho e as arestas de volta (backedge) são orientadas do descendente para um ancestral:

a.) Mostre que

$f*(v) = minf(v), min(backedge(v,w)f(w), min(ufilho(v))f*(u)$

f*(v) é uma função que significa o menor tempo de entrada do menor ancestral alcançado por uma backedge. Isso significa que:

- O valor de f*(v) pode ser o próprio f(v) no caso que ele seja o ancestral alcançado por uma backedge;
- Se mais de um ancestral é alcançado por backedge, o ancestral de menor tempo de entrada será considerado;
- Se o filho de v também alcança o mesmo ancestral, f*(u) = f*(v);

Portanto o menor valor das três opções é considerado.

b.) Forneça uma adaptação do algoritmo DFS que calcule os valores f*(v). Qual a complexidade do tempo de execução?

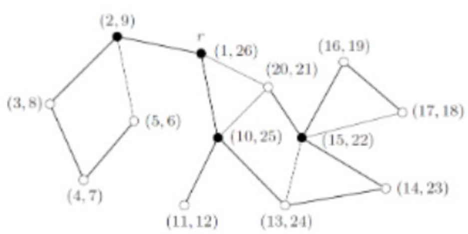
```
DFS(G,r) {
   i = 0;
   mark(r);
   f[r] = i;
   f*[r] = i;
   while(S != NULL) {
      u = last(S);
      i = i+1;
      se u tem um vizinho nao marcado v {
         mark(v);
         push(S,v);
         p[v] = u;
         f[v] = i;
         f*[v] = i;
      }
      senao {
         para cada neighbour = vizinhos de u que nao sao o ancestral direto;
            f*[v] = min(f*[v], f*[neighbour];
         pop(S);
         l[u] = i;
         father = last(S);
         f*[father] = min(f*[u], f*[father]);

      }
   }
```

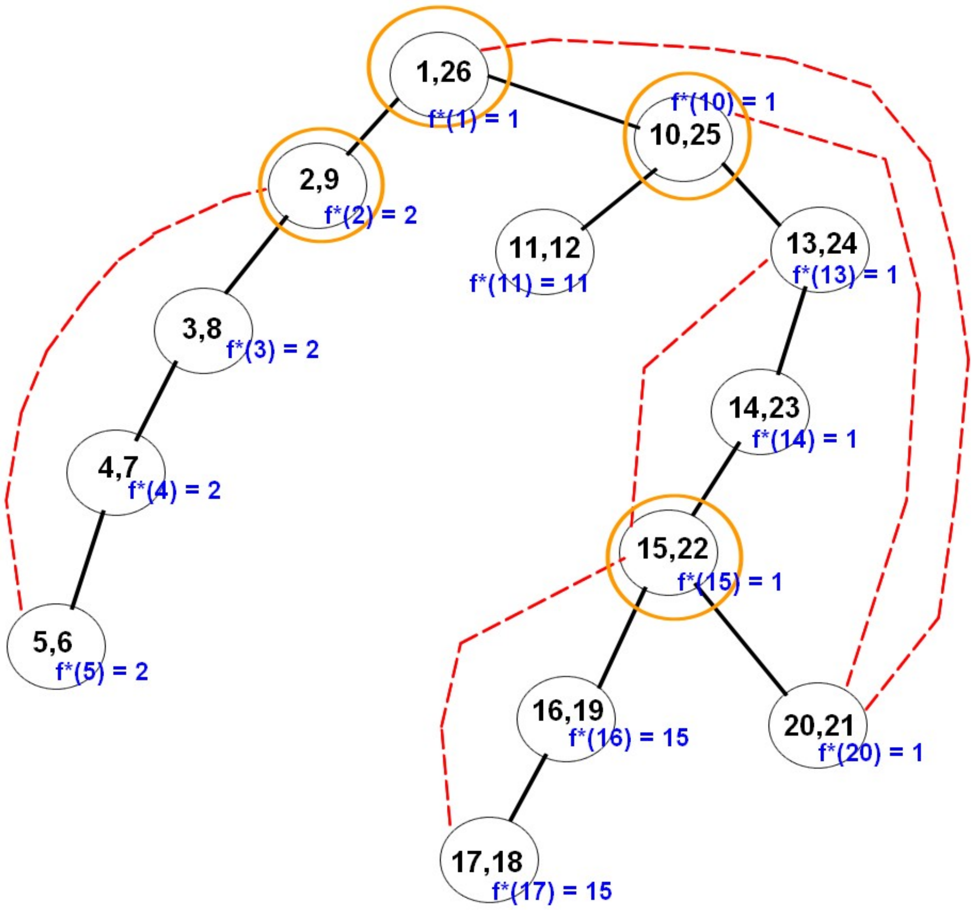
c.) Mostre como encontrar arestas de corte.

Pelo teorema, uma aresta *e* é uma aresta de corte se e somente se pertence a nenhum ciclo de G. Portanto, para encontrar arestas de corte basta procurar quais arestas em um grafo G formam ciclos e ir removendo estas arestas até sobrarem apenas as arestas de corte.

d.) Calcule os valores de f*(v) para cada vértice v do grafo abaixo. Em cada vértice, estão indicados os valores de (f(v), l(v))



Os valores de f*(v) estão indicados abaixo:



Exercício 5. Seja (G,w) um grafo conexo com pesos com arestas de pesos distintos

a.) Prove ou mostre um contra-exemplo: a aresta com peso máximo não faz parte de nenhuma árvore geradora mínima.

Em uma árvore geradora mínima somente arestas seguras são adicionadas a solução. Para procurar uma aresta segura deve-se procurar uma aresta leve: uma aresta que possui o menor peso entre as arestas de um corte. Se a aresta com peso máximo fosse uma aresta leve, entraríamos em contradição pois as outras arestas deveriam ser de peso menor que a aresta de peso máximo.

b.) Prove ou mostre um contra-exemplo: a aresta com segundo menor peso faz parte de qualquer árvore geradora mínima. Para gerar a árvore geradora mínima, é necessário adicionar arestas seguras ao conjunto A. Suponha um conjunto S de vértices que contém a aresta de menor peso de todo o grafo. Ao calcular o d(S), a aresta u,v de segundo menor peso será considerada como aresta segura e será inserida no conjunto A.

c.) Mostre que G tem uma única árvore ótima; Uma árvore geradora é mínima se e somente se cada aresta e de T tem custo mínimo no corte de G T-e. Se esse custo é mínimo não existe outra aresta com custo menor ou senão, esta seria a aresta da árvore geradora. Portanto, os custos obtidos a cada corte são únicos resultando em um único valor para o peso da árvore geradora. A árvore geradora mínima pode ter diversas formas mas o custo sempre é o mesmo.