Ciclos Euleriano e Hamiltoniano

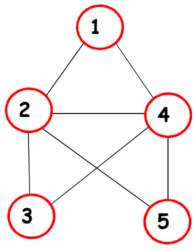
Notas de aula da disciplina IME 04-11311
Algoritmos em Grafos (Teoria dos Grafos)

Paulo Eustáquio Duarte Pinto
(pauloedp at ime.uerj.br)

maio/2018

Grafos - Ciclo e Caminho Euleriano

Ciclo Euleriano- Problema da casinha
Um ciclo Euleriano em um grafo *G* é um ciclo que contém todas as arestas do grafo.



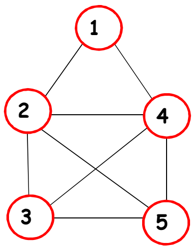
Um ciclo Euleriano no grafo:

3, 2, 4, 5, 2, 1, 4, 3

Este é um problema importante para empresas de entregas, tais como CocaCola e Correios.

Grafos - Ciclo e Caminho Euleriano

Caminho Euleriano- Problema da casinha
Um caminho Euleriano em um grafo *G* é um caminho que contém todas as arestas do grafo, podendo o primeiro vértice ser diferente ao último do caminho.



Um caminho Euleriano no grafo:

3, 2, 4, 5, 2, 1, 4, 3, 5

Este grafo possui caminho Euleriano, mas não possui ciclo Euleriano.

Grafos - Determinação de um ciclo Euleriano

Lema 1: Um grafo *G* simples cujos vértices tenham todos grau maior ou igual a 2 possui um ciclo.

Prova:
A prova é construtiva. Tome um vértice qualquer v e, sucessivamente, vizinhos do último vértice tomado. O novo vértice a entrar no caminho ou já foi tomado antes, o que termina o ciclo, ou possui um vizinho ainda não considerado. Como o número de vértices é finito, o processo de construção sempre acaba.

Grafos - Determinação de um ciclo Euleriano

Teorema 5: Um grafo *G* possui Ciclo Euleriano sse for conexo e todos os vértices tiverem grau par.

Prova:
⇒: Seja $v_1,...v_k,v_1$ um ciclo Euleriano de *G*. Cada ocorrência de um dado vértice v_i no ciclo contribui com 2 unidades para o grau de v_i. Logo, v_i tem grau par. Para verificar que o grafo é conexo, basta tomar qualquer par de vértices (u, v) e exibir um caminho entre esse par. Isso é feito considerando, no ciclo, uma sequência de vértices que começa com u e termina com v, ou vice-versa. Essa sequência sempre existirá, claro. Eliminam-se todos os ciclos dessa sequência (trechos que começam e terminam no mesmo vértice), deixando apenas o vértice inicial do ciclo. O resultado é claramente um caminho de u para v.

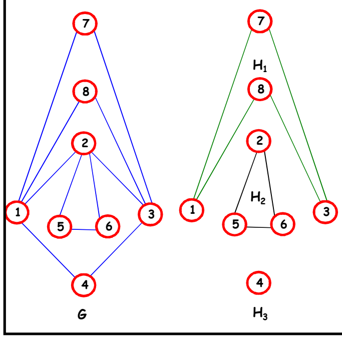
Grafos - Determinação de um ciclo Euleriano

Teorema 5: Um grafo *G* é Euleriano sse for conexo e todos os vértices tiverem grau par.

Prova:
⇐: A prova é por indução em m. Como todo vértice tem grau par ≥ 2, existe um ciclo C_1 em *G* (Lema 1). Retirando esse ciclo de *G*, o grafo restante tem um ou mais componentes onde, cada um deles é vazio ou tem todos os vértices com grau par. Cada componente tem menos arestas que o grafo original. Além disso, cada componente não vazio tem pelo menos um vértice em comum com C_1. Por indução, cada um dos componentes tem um ciclo Euleriano. Então é possível compor um ciclo Euleriano para *G*, tomando o ciclo C_1 e inserindo o ciclo Euleriano de cada componente logo após o primeiro vértice de C_1 que é comum ao componente.

Grafos - Determinação de um ciclo Euleriano - Exemplo

Resultado da retirada do ciclo C_1 (1,2,3,4,1)



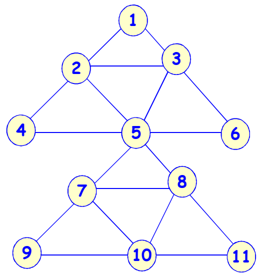
Ciclo Euleriano de H_1:
(1,7,3,8,1)

Ciclo Euleriano de H_2:
(2,6,5,2)

Ciclo Euleriano de G:
(1,7,3,8,1,2,6,5,2,3,4,1)

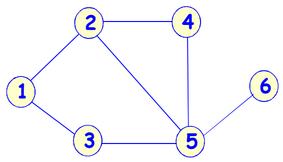
Grafos – Determinação de um ciclo Euleriano

EXS8: mostrar a construção de um ciclo euleriano usando a idéia do Teorema 5.



Grafos - Representação Computacional simplificada

Convenções

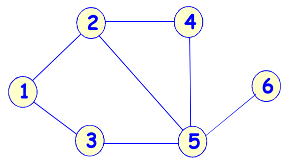


1) Quando não explicitado, supomos que o grafo G(V,E) é grafo simples (sem loops nem multiarestas)
2) Um grafo G(V,E) tem n vértices e m arestas.
3) Os vértices serão numerados de 1 a n.
4) Neste curso trabalharemos com dois tipos de representação:
   matriz de adjacências - representação geral mais simples
   up-trees -quando se quer checar apenas a conectividade

Obs: outros tipos de representação serão estudados no próximo curso.

Grafos - Representação Computacional simplificada

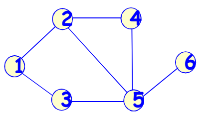
Matriz de adjacências



	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	0	1	1	0
3	1	0	0	0	1	0
4	0	1	0	0	1	0
5	0	1	1	1	0	1
6	0	0	0	0	1	0

Grafos - Representação Computacional simplificada

Matriz de adjacências



	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	0	1	1	0
3	1	0	0	0	1	0
4	0	1	0	0	1	0
5	0	1	1	1	0	1
6	0	0	0	0	1	0

-Matriz n x n, onde cada valor (u, v) igual a 1 indica que u e v são adjacentes, 0 que não são.

-Matriz simétrica com 2m 1's, pois cada aresta é representada 2 vezes.

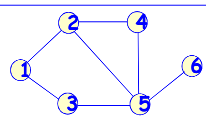
-Diagonal principal zerada.

-Boa para grafos densos.

-Pode ser estendida para representar multigrafos ou grafos simples com pesos nas arestas.

Grafos - Representação Computacional simplificada

Leitura de um grafo simples e armazenamento em uma matriz de adjacências



```
CriaGrafo:
    ler (n, m)
    ZeraMatriz
    para i ← 1 até m incl:
        Ler(u, w)
        E[u,w] ← 1;  E[w,u] ← 1;
```

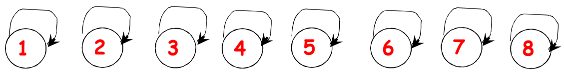
Entrada:
6 7
5 3
2 4
5 6
4 5
2 5
3 1
1 2

Grafos – Determinação de componentes usando up-trees

Conhecendo-se as arestas do grafo, pode-se determinar seus componentes, usando-se os algoritmos de tratamento de conjuntos com up-trees (Union-Find).

a) Criação de conjuntos com up-trees (representadas em vetor):

```
Criação():
#dados: inteiro n;
    para i ← 1 até n inclusive:
        pai[i] ← i
```

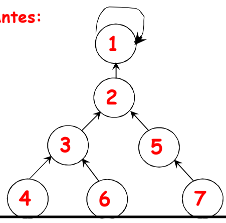


Grafos – Determinação de componentes usando up-trees

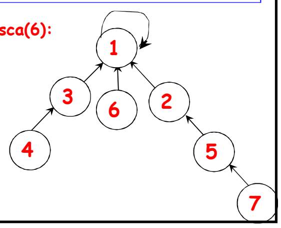
b) Busca da raiz de um elemento com compactação de caminho:

```
Busca(p):
#dados: inteiro p
    se (pai[p] ≠ p):
        pai[p] ← Busca(pai[p])
    retornar pai[p]
```

Antes:



Após Busca(6):

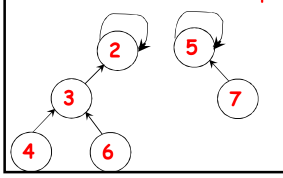


Grafos – Determinação de componentes

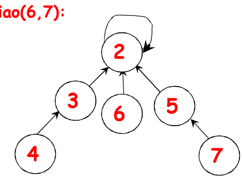
c) União de conjuntos por valor da raiz;

```
Uniao(p,q):
#dados: inteiro p, q;
    Pp ← Busca(p);   Pq ← Busca(q);
    se (Pp < Pq):
        pai[Pq] ← Pp
    senão:
        pai[Pp] ← Pq;
```

Antes:



Após Uniao(6,7):

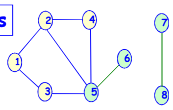


Grafos – Determinação de componentes

Leitura do grafo e determinação de componentes;

```
Componentes():
   ler(n,m);  Criação(n);
   para i ← 1 até m inclusive:
       Ler(u,v);  Uniao(u,v);
   para i ← 1 até  n inclusive:
       p ←  Busca(i)
   c ← 0
   para i ← 1 até n inclusive:
       se (pai[i] = i):
           c ← c+1;  Imprimir ("Componente ",c,":");
           para j ← i até n inclusive:
               se (pai[j]=i):
                   Imprimir j;
```


Grafos – Determinação de componentes



Exemplo (n=8):

Vetor Pai

u	v
5	4
1	3
8	7
5	6
2	4
2	5
5	3
1	2

1	2	3	4	5	6	7	8
1	2	3	4	4	6	7	8
1	2	1	4	4	6	7	8
1	2	1	4	4	6	7	7
1	2	1	4	4	4	7	7
1	2	1	2	4	4	7	7
1	2	1	2	2	4	7	7
1	1	1	2	2	4	7	7
1	1	1	2	2	4	7	7

1	1	1	1	1	1	7	7

Grafos – Determinação de componentes

Observações sobre os algoritmos UNION-FIND

1. É comum que as operações de fusão sejam feitas por tamanho de cada subconjunto. A implementação por valor da raiz é mais simples.

2. A complexidade das operações de n operações de fundir e m operações de buscar é $O(n+m\alpha(m+n,n))$, onde α é o inverso da função de Ackerman. Na prática $\alpha(n,m) \leq 4$.

3. O algoritmo mostrado para listar os componentes não é eficiente. Mas se precisarmos apenas contar os componentes o algoritmo pode ser tornado eficiente.

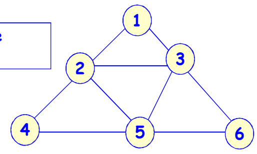
Grafos – Determinação de componentes

EXS9:
a) mostrar as up-trees para a seguinte sequência de uniões de conjuntos disjuntos com n=7 elementos
b) Ao final das uniões, mostrar a situação após as buscas de todos os elementos.

3 5
7 4
1 3
6 4
4 2
7 3
6 7

Grafos – Determinação de um ciclo Euleriano

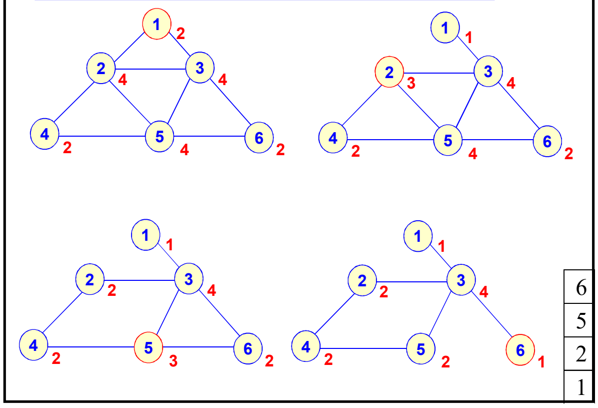
Algoritmo recursivo que destrói o grafo



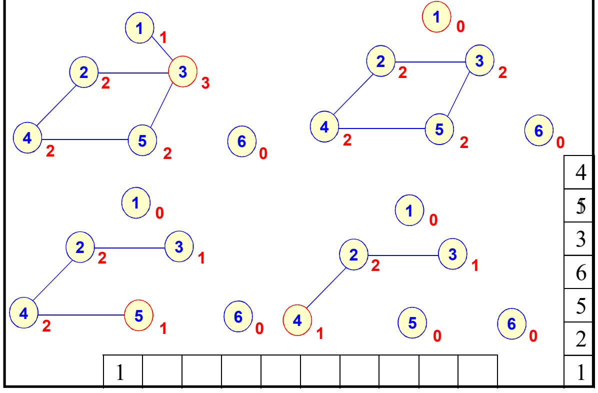
Circuito Euleriano();
#dados grafo G, pilha P, inteiro v;
Esvazia(P); Push (P, v);
enquanto (topo ≠ 0):
v ← P[topo]; w ← ProxV(v); Eliminar (v, w);
Push (P,w);
enquanto ((topo ≠ 0) e (d[P[topo]] = 0)):
w ← Pop(P); Imprimir w;

Complexidade: O(n+m)

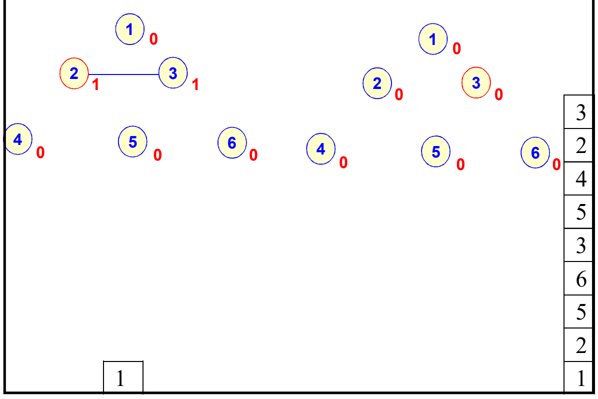
Grafos – Determinação de um ciclo Euleriano



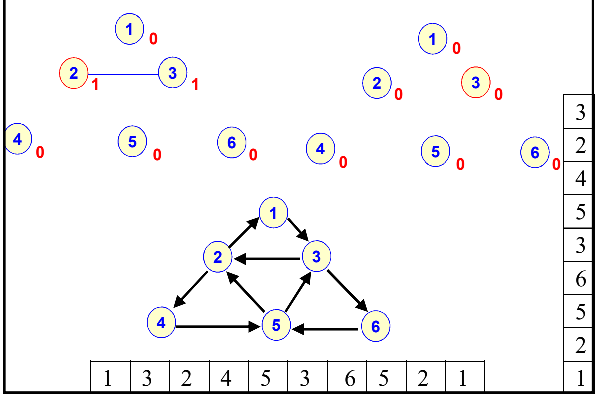
Grafos – Determinação de um ciclo Euleriano



Grafos – Determinação de um ciclo Euleriano

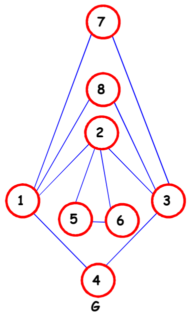


Grafos – Determinação de um ciclo Euleriano



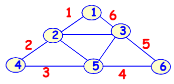
Grafos – Determinação de um ciclo Euleriano

EXS10: Mostrar um ciclo Euleriano para o grafo abaixo.



Grafos – Determinação de um caminho entre vértices

Algoritmo de backtracking.
O vetor M marca os vértices já visitados e o vetor P guarda o caminho.




```
Caminho(v, w);
#dado grafo G
    P[++np] ← v;   M[v] ← 1;
    se (v = w):
         escrever (caminho (P + w))
         retornar 0
    para i ← 1 até n:
         se ((E[v, i] = 1) e (M[i] = 0)):
              se (Caminho(i, w) = 0):
                     retornar 0;
    np--;
    retornar 1
Externamente:
    M[*] ← 0;   np ← 0;    Caminho(v,w);
```

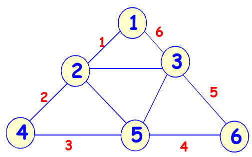

Grafos - Determinação de um caminho entre vértices

Teorema 6: O algoritmo Caminho funciona corretamente, com complexidaade $O(n^2)$ se usarmos matriz de adjacências.

Prova:
Por indução em n podemos ver que o algoritmo funciona corretamente. A recursão é chamada no máximo n vezes. Com a representação de matriz de adjacências, cada chamada executa um loop n vezes. Logo a complexidade será $O(n^2)$.

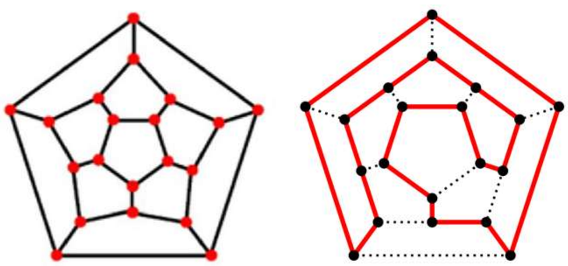
Grafos – Determinação de um ciclo Hamiltoniano

Ciclo Hamiltoniano: ciclo que passa por todos os vértices do grafo. O adjetivo "hamiltoniano" deve-se ao matemático irlandês Sir William Rowan Hamilton (1805-1865), que inventou um jogo que envolve um dodecaedro (sólido regular com 20 vértices, 30 arestas e 12 faces).

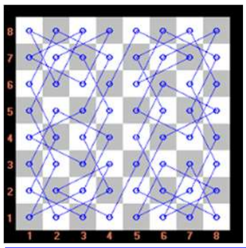


Grafos – Determinação de um ciclo Hamiltoniano

Ciclo Hamiltoniano em um dodecaedro (sólido regular com 20 vértices, 30 arestas e 12 faces).



Grafos – Determinação de um ciclo Hamiltoniano



O percurso do cavalo em um tabuleiro de xadrez é um ciclo hamiltoniano em um grafo cujos vértices são as casas do tabuleiro existindo aresta entre dois vértices se for possível o cavalo pular de uma casa para a outra.

Outras aplicações:
a) circuito ideal para ser feito por empresas de distribuição que têm que visitar todas as cidades de uma região.
b) Problema Banquete

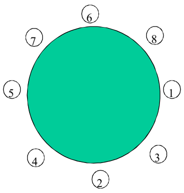
Grafos – Determinação de um ciclo Hamiltoniano

Problema Banquete

Organizar os convidados ao redor de uma mesa de banquete, evitando que inimigos sentem lado a lado, conhecendo-se as inimizados.

Convidados:
- Lula
- Moro
- Pelé
- Maradona
- Trump
- Putin
- ...

Inimizados:
- Lula x Moro
- Pelé x Maradona
- Trump x Putin
- ...



Solução do problema:

Criar um grafo completo dos convidados, eliminar as arestas das inimizades e procurar um ciclo Hamiltoniano no mesmo

Grafos - Determinação de um ciclo Hamiltoniano

Não são conhecidas condições necessárias e suficientes para um grafo ser hamiltoniano.

Não são também conhecidos algoritmos eficientes para se determinar um ciclo hamiltoniano.

Uma condição necessária é a seguinte:

a) Se $G(V,E)$ é hamiltoniano e S um subconjunto próprio de V, então o número de componentes conexos de G-S é $\leq |S|$.

Uma condição suficiente é a seguinte:

b) Se $G(V,E)$ é tal que todo vértice tem grau $\geq n/2$, então G é hamiltoniano.

Grafos - Determinação de um ciclo Hamiltoniano

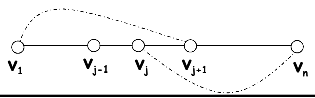
Teorema 7: Se $G(V,E)$ é hamiltoniano e S um subconjunto próprio de V, então o número de componentes conexos de G-S é $\leq |S|$.

Prova: Seja C um ciclo hamiltoniano de G. Então, o número máximo de componentes conexas do grafo C-S é $\leq|S|$, porque C-S é a união de, no máximo, $|S|$ caminhos simples disjuntos. Portanto, G-S e C-S têm o mesmo conjunto de vértices. Além disso, toda aresta de C-S é também de G-S. Logo, o número máximo de componentes conexas deste último é menor ou igual ao do primeiro. Isso prova o teorema.

Grafos - Determinação de um ciclo Hamiltoniano

Teorema 8: Se $G(V,E)$ é tal que $n \geq 3$ e todo vértice tem grau $\geq n/2$, então G é hamiltoniano.

Prova: Suponhamos o teorema falso. Seja G um grafo maximal que obedece as condições e que não seja hamiltoniano e sem a aresta (v_1,v_n) . O acréscimo da aresta (v_1,v_n) torna o grafo hamiltoniano e todo ciclo hamiltoniano em $G+(v_1,v_n)$ tem a aresta (v_1,v_n). Portanto, existe um caminho hamiltoniano $v_1...v_n$ em G. Nesse caminho, existem dois vértices v_j e v_{j+1}, $1 \leq j < n$, tais que (v_1, v_{j+1}) e (v_j,v_n) são arestas de G, pela característica de G. Então existe um ciclo hamiltoniano em G: $v_1v_{j+1}...v_nv_jv_{j-1}...v_1$, o que é absurdo. Logo G é hamiltoniano.



Grafos - Determinação de um ciclo Hamiltoniano

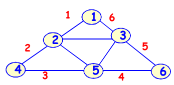
EXS11:

a) Inventar um grafo com 8 vértices e verificar o Teorema 5.

b) Inventar um grafo com 6 vértices e verificar o Teorema 6.

Grafos – Determinação de um ciclo Hamiltoniano

Algoritmo de backtracking. O vetor M marca os vértices já visitados e o vetor P guarda o ciclo.




```
CicloH(v, w);
    P[++np] ← v;   M[v] ← 1;
    se ((E[v,w] = 1) e (np = n)):
        escrever (ciclo (P + w));
        retornar 0
    para i de 1 a n:
        se ((E[v, i] = 1) e (M[i] = 0)):
            se (CicloH(i, w) = 0):
                retornar 0
    np--;  M[v] ← 0;
    retornar 1

M[*] ← 0;   np ← 0;    CicloH(v,v);
```

Grafos – Determinação de um ciclo Hamiltoniano

Observe que a adição da instrução que desmarca o vértice após a análise de seus vizinhos torna o algoritmo exponencial.

O problema é NP-Completo (não são conhecidos algoritmos polinomiais e a suspeita é que não existam.)

Grafos - Determinação de um ciclo Hamiltoniano

Teorema 7: O algoritmo CicloH funciona corretamente, com complexidaade exponencial.

Prova:
Por indução em n podemos ver que o algoritmo funciona corretamente. Para ver que o algoritmo é exponencial, basta imaginar um grafo com n vértices onde os vértices de 2 a n formam um grafo completo e o vértice 1 é vizinho de apenas um dos demais vértices. O algoritmo examinará todos os (n-1)! caminhos correspondentes ao subgrafo completo. Portanto, sua complexidade é exponencial.

Ciclos Euleriano e Hamiltoniano

FIM