

Três problemas envolvendo caminhos em grafos

1. Existência de um $v-w$ caminho
2. Caminhos hamiltonianos/circuitos hamiltonianos
3. Trilhas eulerianas (trilhas e trilhas fechadas)

Passeios, trilhas, e caminhos

$v_0v_1 \dots v_\ell$: seqüência de vértices de G , com $v_{i-1}v_i$ aresta em G

1. Passeio (sem restrição adicional)
2. Trilhas: todas as arestas $v_{i-1}v_i$ ($1 \leq i \leq \ell$) distintas
3. Caminho: todos os vértices v_i ($0 \leq i \leq \ell$) distintos

Fechado: $v_0 = v_\ell$

Algoritmos

- ▷ $v-w$ caminho: busca em profundidade
- ▷ Hamilton: busca exaustiva
- ▷ Euler: decomposição em trilhas fechadas

$v-w$ caminho (matriz de adjacências); prog17.11

```
static int visited[maxV];
int pathR(Graph G, int v, int w)
{ int t;
  if (v == w) return 1;
  visited[v] = 1;
  for (t = 0; t < G->V; t++)
    if (G->adj[v][t] == 1)
      if (visited[t] == 0)
        if (pathR(G, t, w)) return 1;
  return 0;
}
```

v-w caminho (matriz de adjacências); prog17.11

```
int GRAPHpath(Graph G, int v, int w)
{ int t;
  for (t = 0; t < G->V; t++) visited[t] = 0;
  return pathR(G, v, w);
}
```

$v-w$ caminho (listas de adjacências)

- ▷ Exercício!
- ▷ Quanto tempo consome o algoritmo acima (com matrizes de adjacência)?
- ▷ Quanto tempo consome o algoritmo acima (com listas de adjacência)?

Algoritmos lineares para grafos $G = (V, E)$: algoritmos que consomem tempo $O(|V| + |E|)$ [tamanho de G : $|V| + |E|$]

Caminhos hamiltonianos (matriz de adjacências); prog17.12

```
static int visited[maxV];
int pathR(Graph G, int v, int w, int d)
{ int t;
  if (v == w)
    { if (d == 0) return 1; else return 0; }
  visited[v] = 1;
  for (t = 0; t < G->V; t++)
    if (G->adj[v][t] == 1)
      if (visited[t] == 0)
        if (pathR(G, t, w, d-1)) return 1;
  visited[v] = 0;
  return 0;
}
```

Caminhos hamiltonianos (matriz de adjacências); prog17.12

```
int GRAPHpathH(Graph G, int v, int w)
{ int t;
  for (t = 0; t < G->V; t++) visited[t] = 0;
  return pathR(G, v, w, G->V-1);
}
```


Caminhos hamiltonianos

Exemplo ruim simples: $K^{n-1} \cup K^1$

▷ Tempo proporcional a $(n - 2)!$ (com v fixo no completo e w no K^1)

Stirling etc

n	Tempo
15	1s
19	1 dia
21	> 1 ano
23	> 6×10^2 anos

Se $\times 200.000$ mais rápido: para $n = 23$ teríamos > 1 dia!

Caminhos hamiltonianos

- ▷ Stirling:

$$n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$$

- ▷ Mais precisamente:

$$\left(\frac{n}{e}\right)^n \sqrt{2\pi n} \leq n! \leq e^{1/12n} \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$$

- ▷ Fácil de provar por indução em n :

$$e \left(\frac{n}{e}\right)^n \leq n! \leq ne \left(\frac{n}{e}\right)^n$$

(use a desigualdade fundamental $1 - x \leq e^x$, válida para todo x).

Trilhas eulerianas (existência)

```
/* prog17.13 (modificado) */
int GRAPHpathE(Graph G, int v, int w)
{ int t;
  if ((GRAPHdeg(G, v) % 2 == 0) || (GRAPHdeg(G, w) % 2 == 0))
    return 0;
  for (t = 0; t < G->V; t++)
    if ((t != v) && (t != w))
      if ((GRAPHdeg(G, t) % 2) != 0) return 0;
  return 1;
}
```

Trilhas eulerianas; prog17.14 (modificado)

```
#include "STACK.h"
int path(Graph G, int v)
{ int w;
  for (; G->adj[v] != NULL; v = w)
  {
    STACKpush(v);
    w = G->adj[v]->v;
    GRAPHremoveE(G, EDGE(v, w));
  }
  return 1;
}
```

Trilhas eulerianas; prog17.14 (modificado)

```
void pathEshow(Graph G, int v)
{
    STACKinit(G->E);
    printf("%d", v);
    while (path(G, v) && !STACKempty())
        { v = STACKpop(); printf("-%d", v); }
    printf("\n");
}
```

Exercício

▷ Gere vários grafos aleatórios com n vértices e m arestas e estime a probabilidade de eles serem conexos. Use os seguintes valores de m e n grandes:

- $m = \lceil (1/4)n \log n \rceil$

- $m = \lceil (1/2)n \log n \rceil$

- $m = \lceil n \log n \rceil$

- $m = \lceil 2n \log n \rceil$

($\log = \ln$)

Trilhas eulerianas; (Ex. 17.96/97; para meditar)

```
#define GRAPHiso(G, v) (GRAPHdeg(G, v) == 0)
void pathEshow(Graph G, int v, int w)
{ int t;
  if ((v == w) && (G->E == 0)) return;
  for (t = 0; t < G->V; t++)
    if (G->adj[v][t] != 0)
      <examine a aresta {v,t}>
}
```


Trilhas eulerianas; (Ex. 17.96/97; para meditar)

```
<examine a aresta {v,t}> =
{
    GRAPHremoveE(G, EDGE(v, t));
    if (GRAPHiso(G, v) || GRAPHpath(G, t, v))
    {
        printf("%d-%d\n", v, t);
        pathEshow(G, t, w);
        GRAPHinsertE(G, EDGE(v, t));
        return;
    }
    GRAPHinsertE(G, EDGE(v, t));
}
```