

Buscas em grafos: profundidade, largura, generalizada

Buscas em grafos: profundidade, largura, generalizada

Recaptação:

1. Busca em profundidade

Buscas em grafos: profundidade, largura, generalizada

Recaptulação:

1. Busca em profundidade
2. Busca em largura

Buscas em grafos: profundidade, largura, generalizada

Recaptulação:

1. Busca em profundidade
2. Busca em largura

Diferença crucial:

Buscas em grafos: profundidade, largura, generalizada

Recaptulação:

1. Busca em profundidade
2. Busca em largura

Diferença crucial: **pilhas** × **filas**

Buscas em grafos: profundidade, largura, generalizada

Recaptulação:

1. Busca em profundidade
2. Busca em largura

Diferença crucial: **pilhas** × **filas**

- ▶ Busca generalizada: políticas generalizadas de processamento de arestas

Buscas em grafos: profundidade, largura, generalizada

Recaptulação:

1. Busca em profundidade
2. Busca em largura

Diferença crucial: **pilhas** × **filas**

- ▶ Busca generalizada: políticas generalizadas de processamento de arestas (*fringes*)

Programa 18.10

```
void pfs(Graph G, Edge e)
{ link t; int v, w;
  GQput(e); pre[e.w] = cnt++;
  while (!GQempty())
  {
    e = GQget(); w = e.w; st[w] = e.v;
    for (t = G->adj[w]; t != NULL; t = t->next)
      if (pre[v = t->v] == -1)
        { GQput(EDGE(w, v)); pre[v] = cnt++; }
      else if (st[v] == -1)
        GQupdate(EDGE(w, v));
  }
}
```

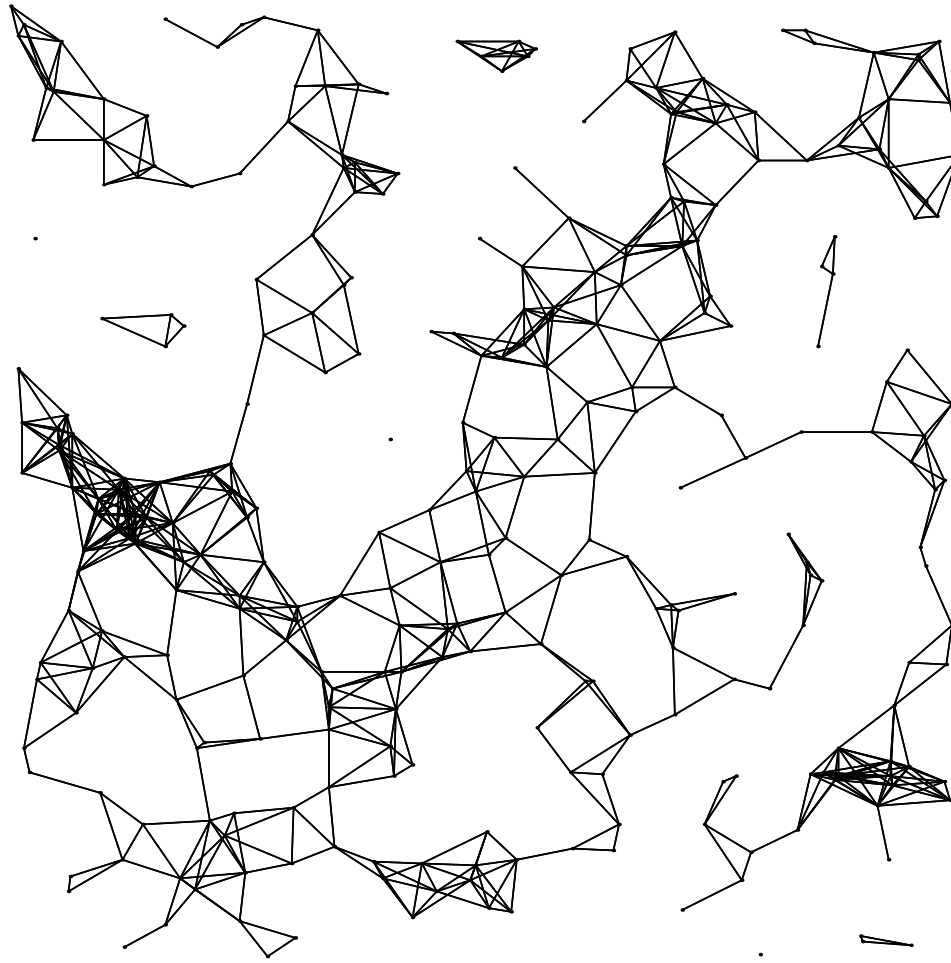

Programa 18.11

```
#include <stdlib.h>
#include "RQ.h"
static Item *s;
static int N;
void RQinit(int maxN)
    { s = malloc(maxN*sizeof(Item)); N = 0; }
int RQempty()
    { return N == 0; }
void RQput(Item x)
    { s[N++] = x; }
void RQupdate(Item x)
    { }
```

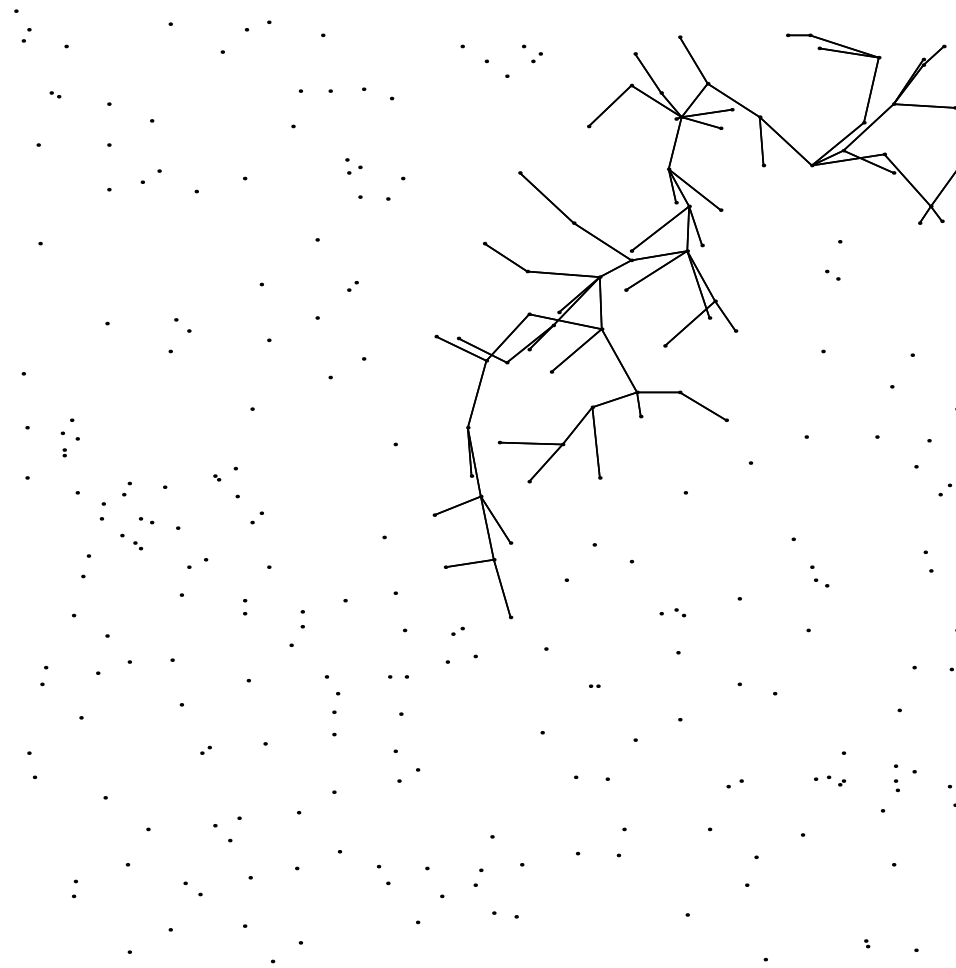
Programa 18.11

```
Item RQget()  
{ Item t;  
  int i = N*(rand()/(RAND_MAX + 1.0));  
  t = s[i]; s[i] = s[N-1]; s[N-1] = t;  
  return s[--N];  
}
```

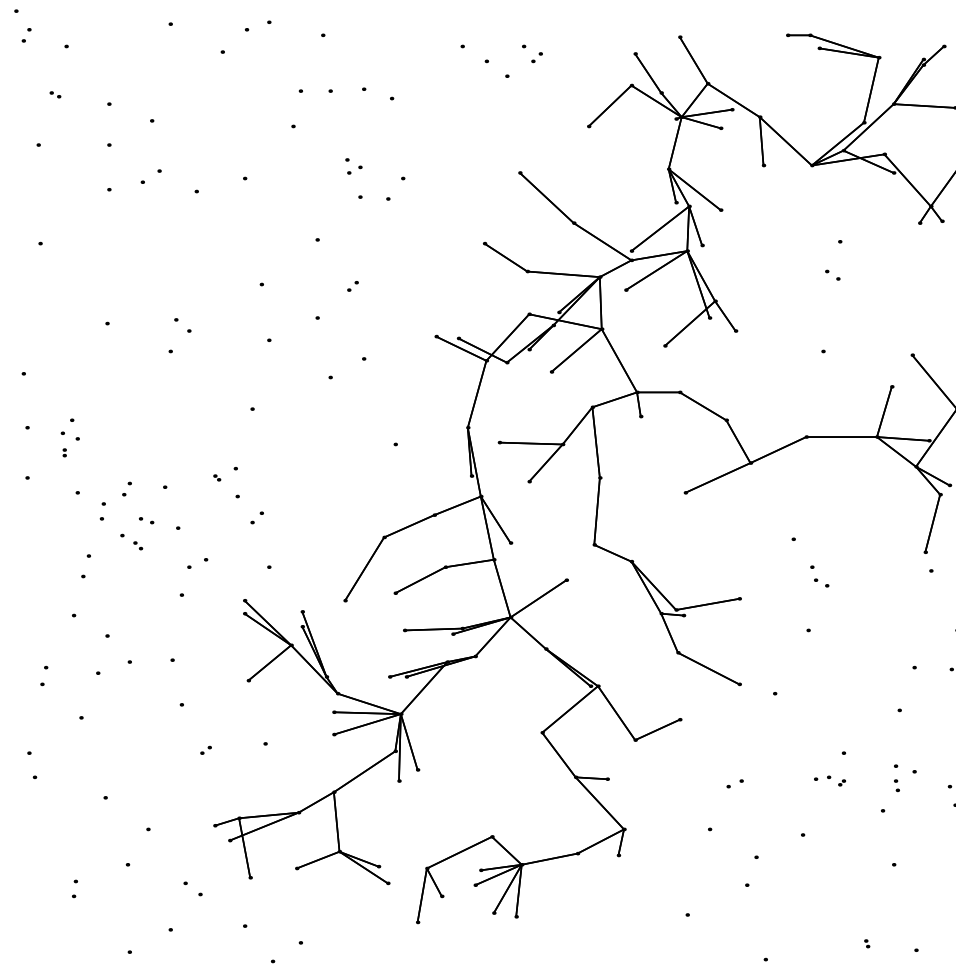
Um grafo geométrico



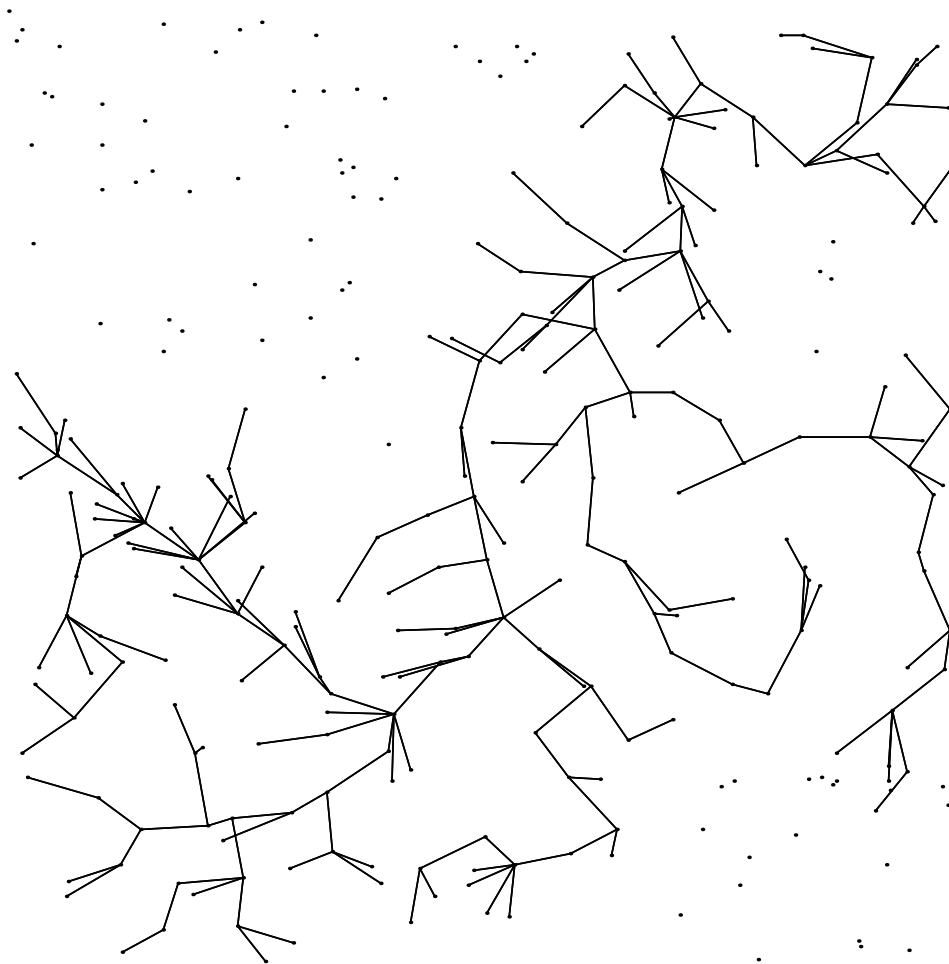
Um grafo geométrico



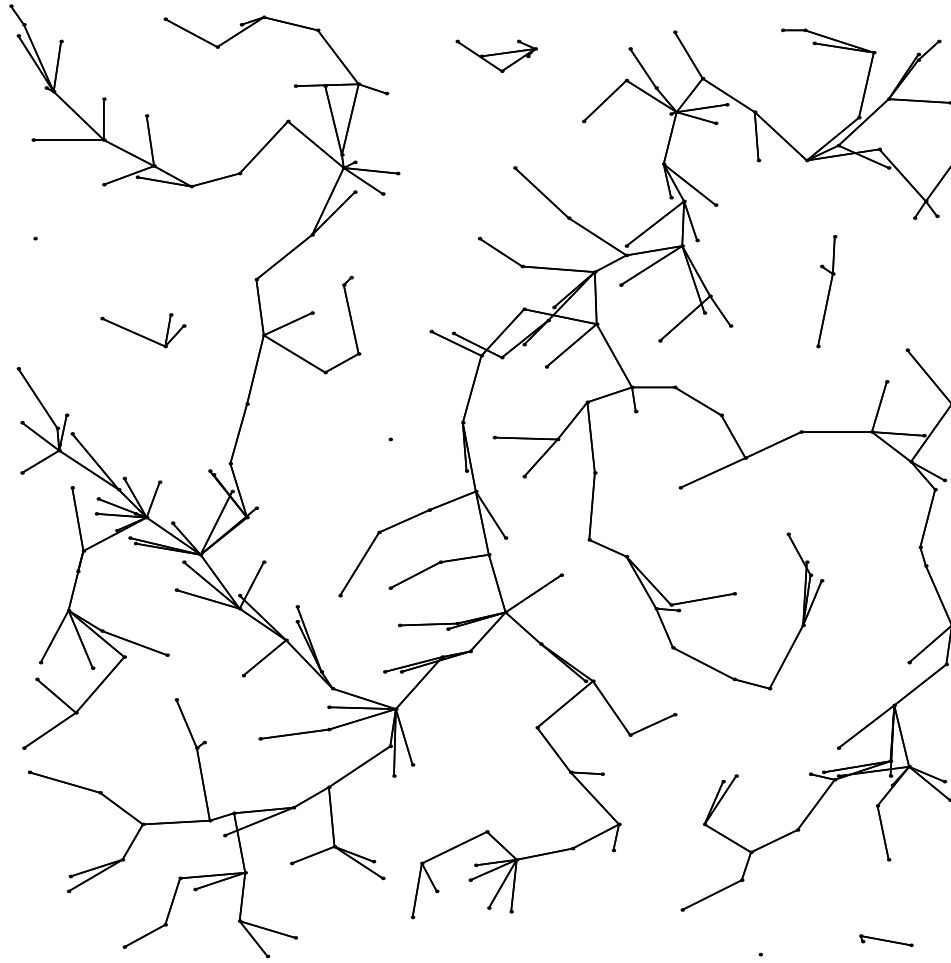
Um grafo geométrico



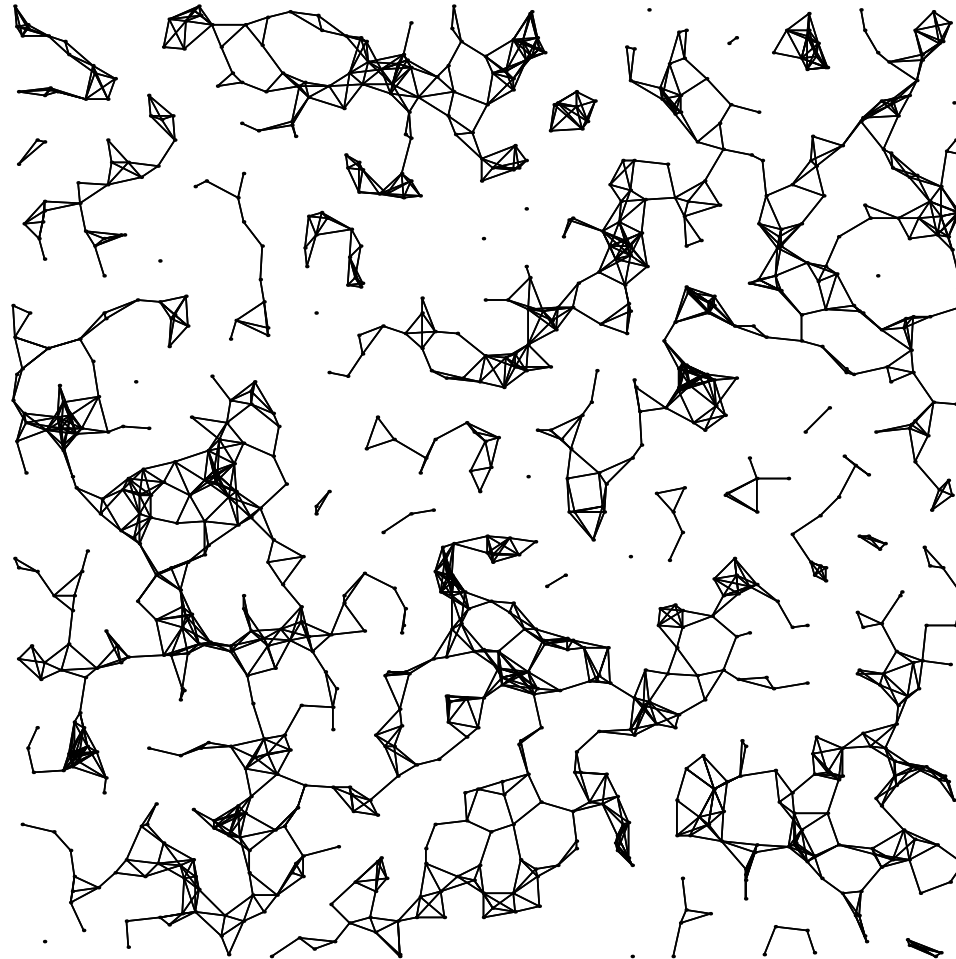
Um grafo geométrico



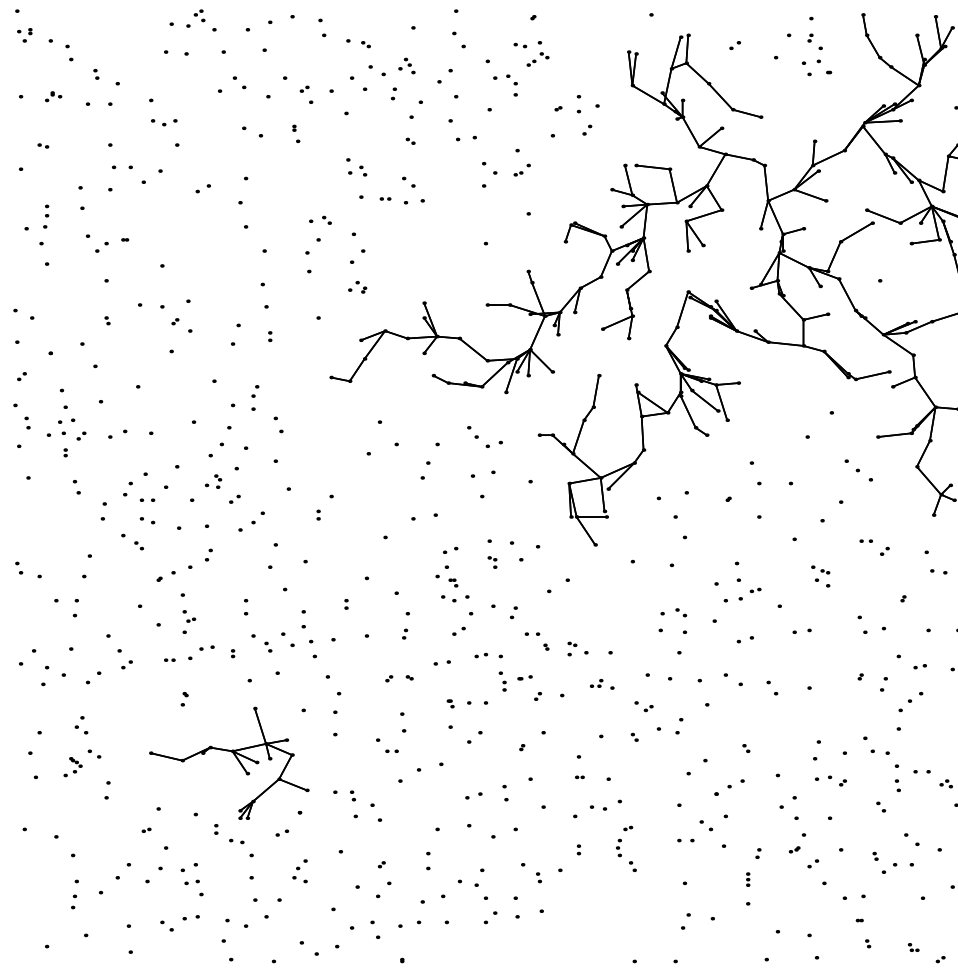
Um grafo geométrico



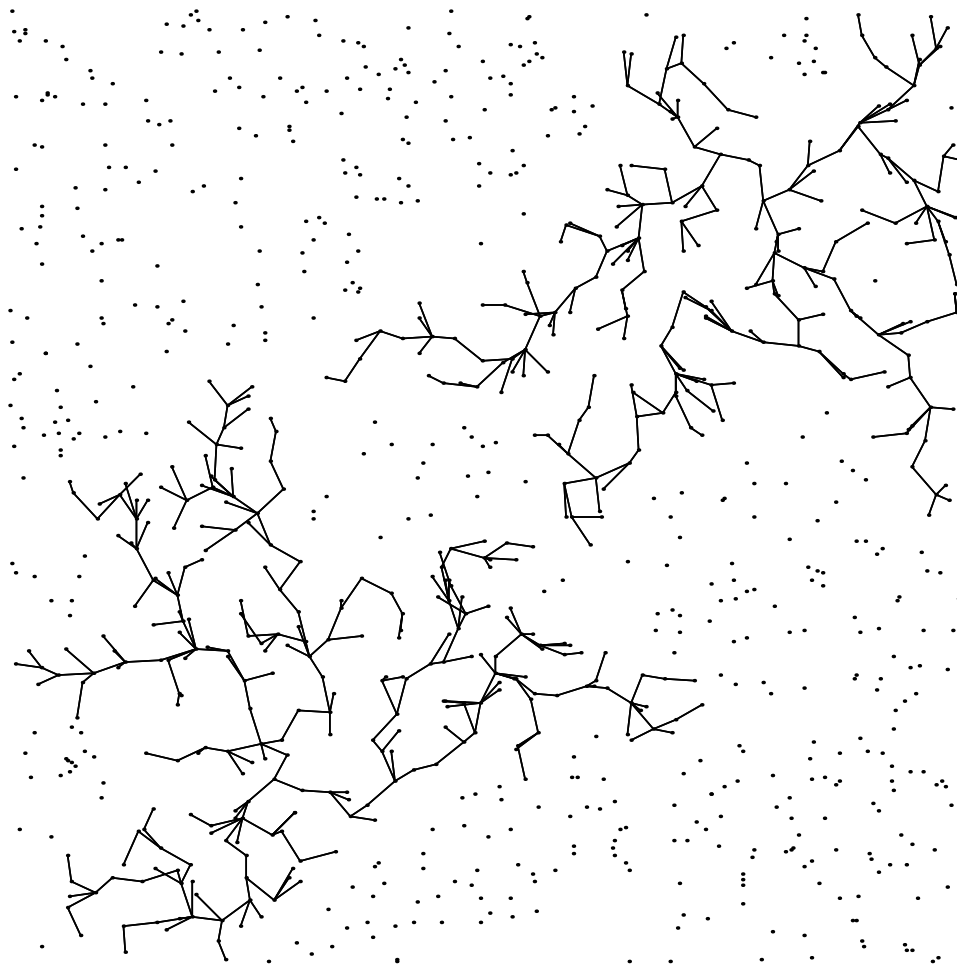
Um grafo geométrico



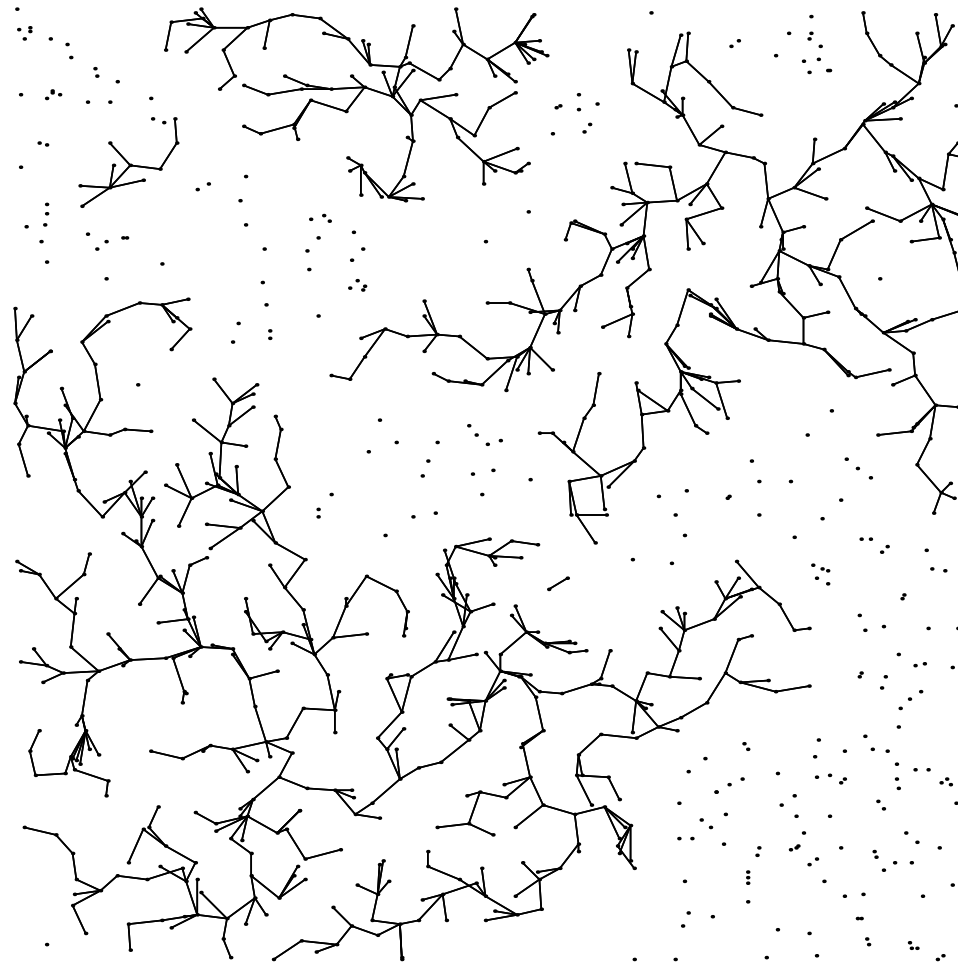
Um grafo geométrico



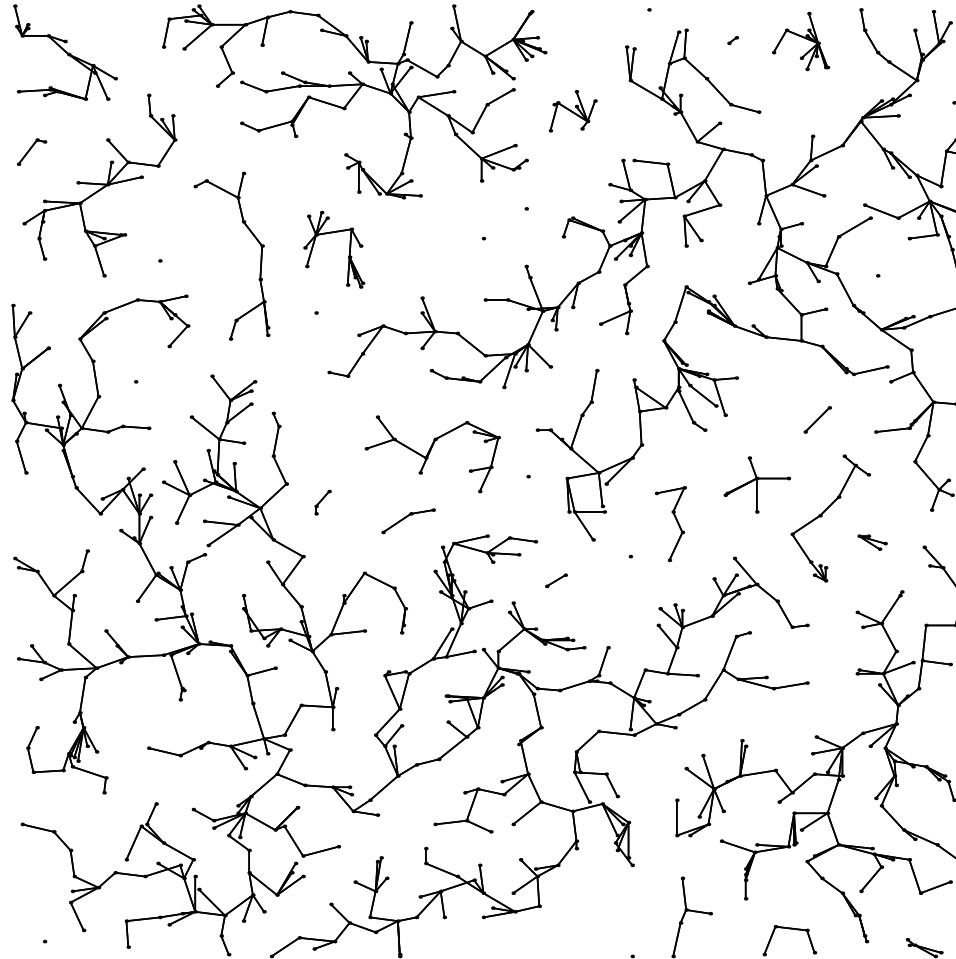
Um grafo geométrico



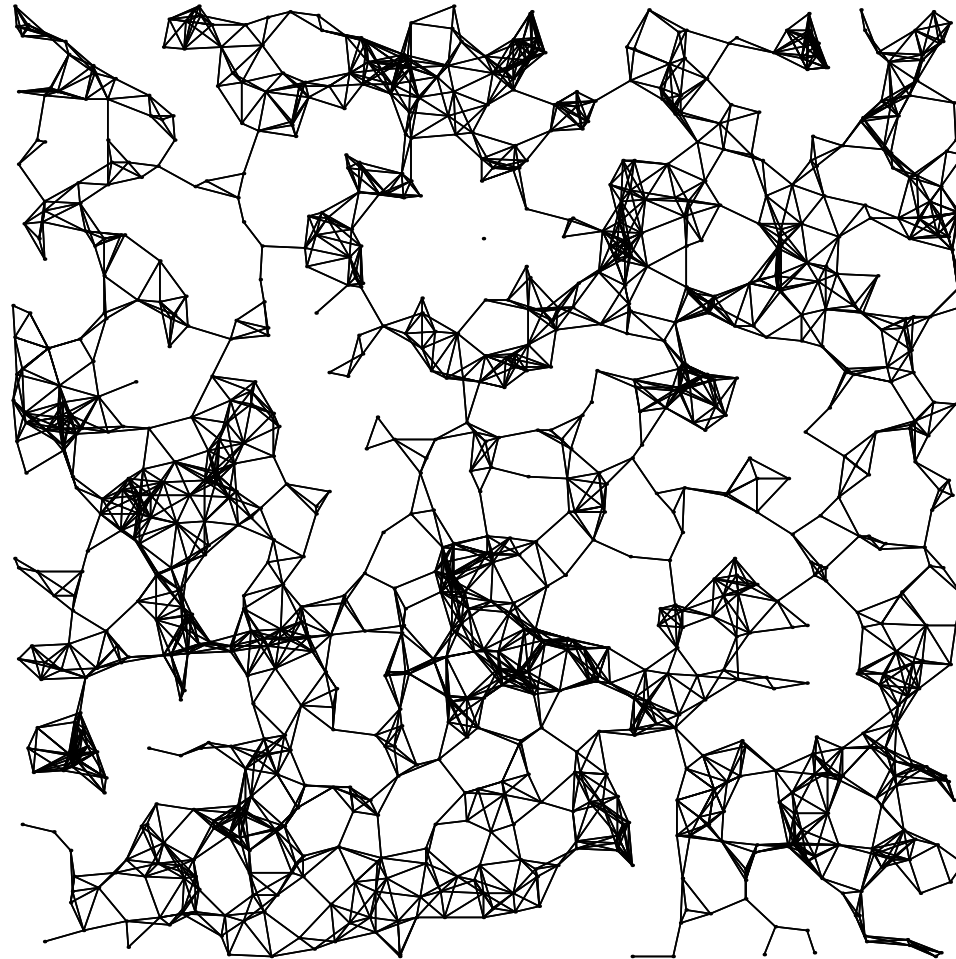
Um grafo geométrico



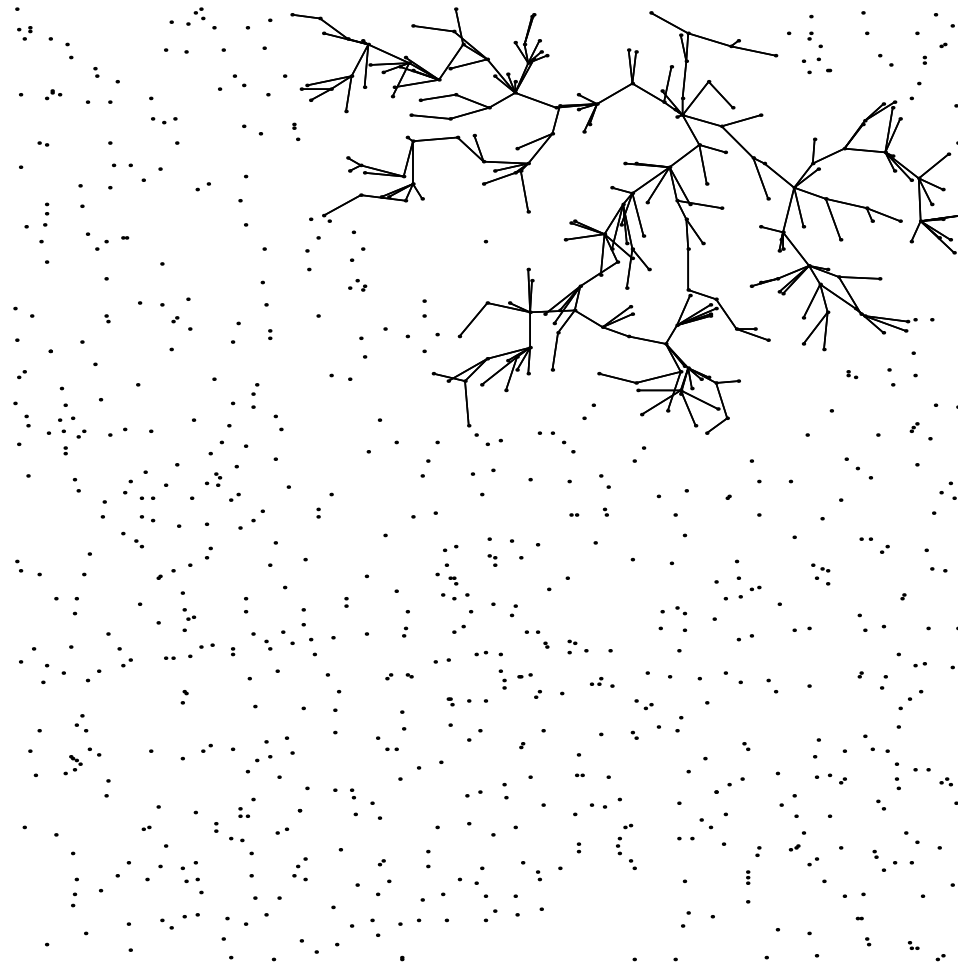
Um grafo geométrico



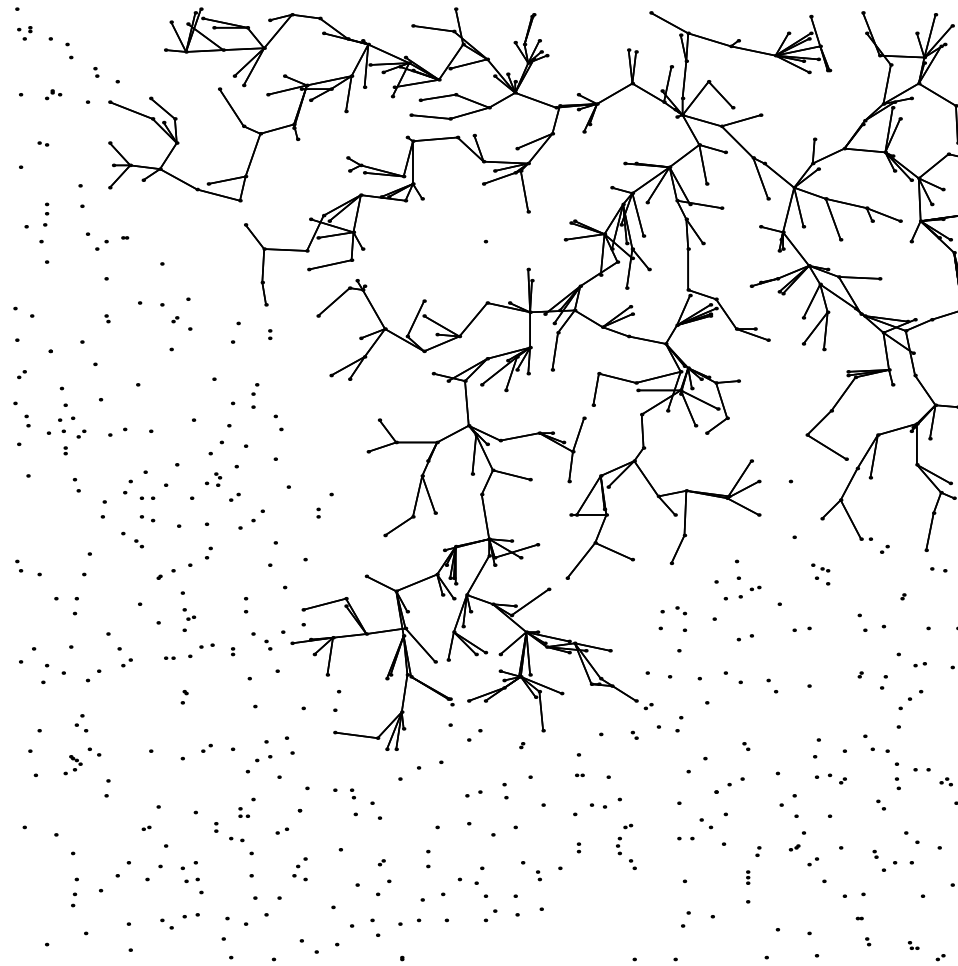
Um grafo geométrico



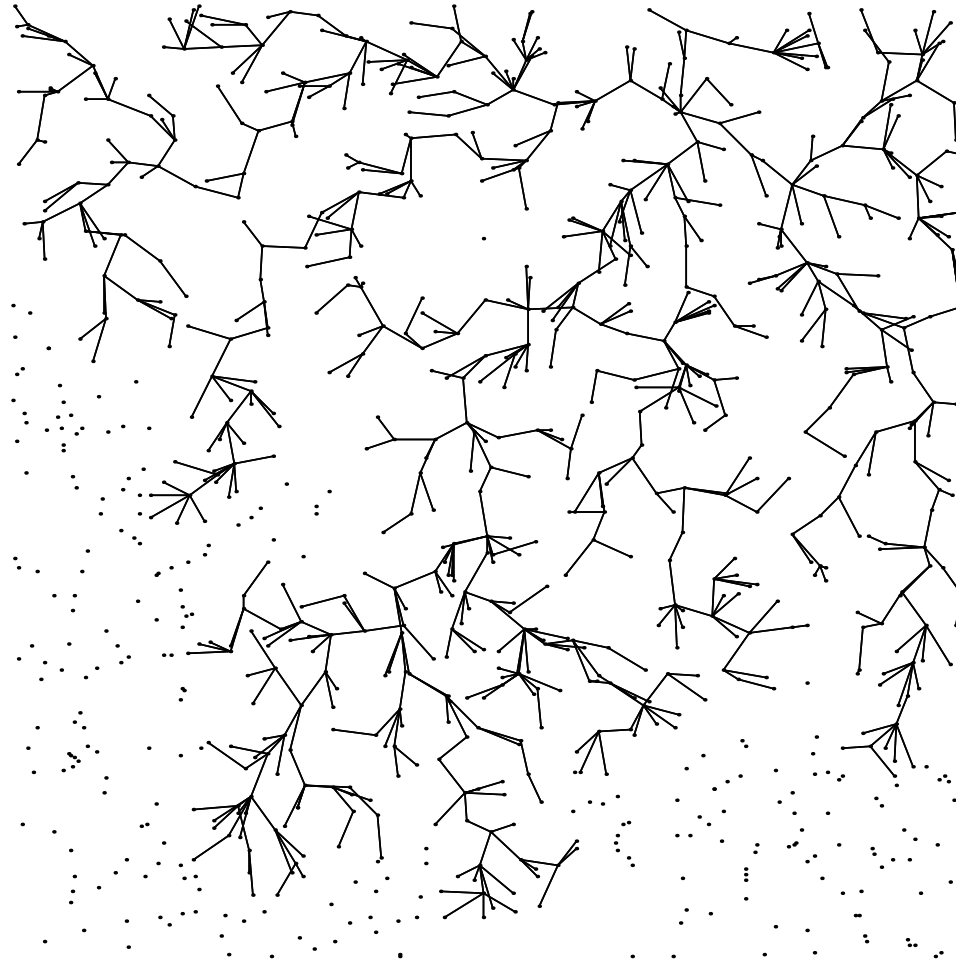
Um grafo geométrico



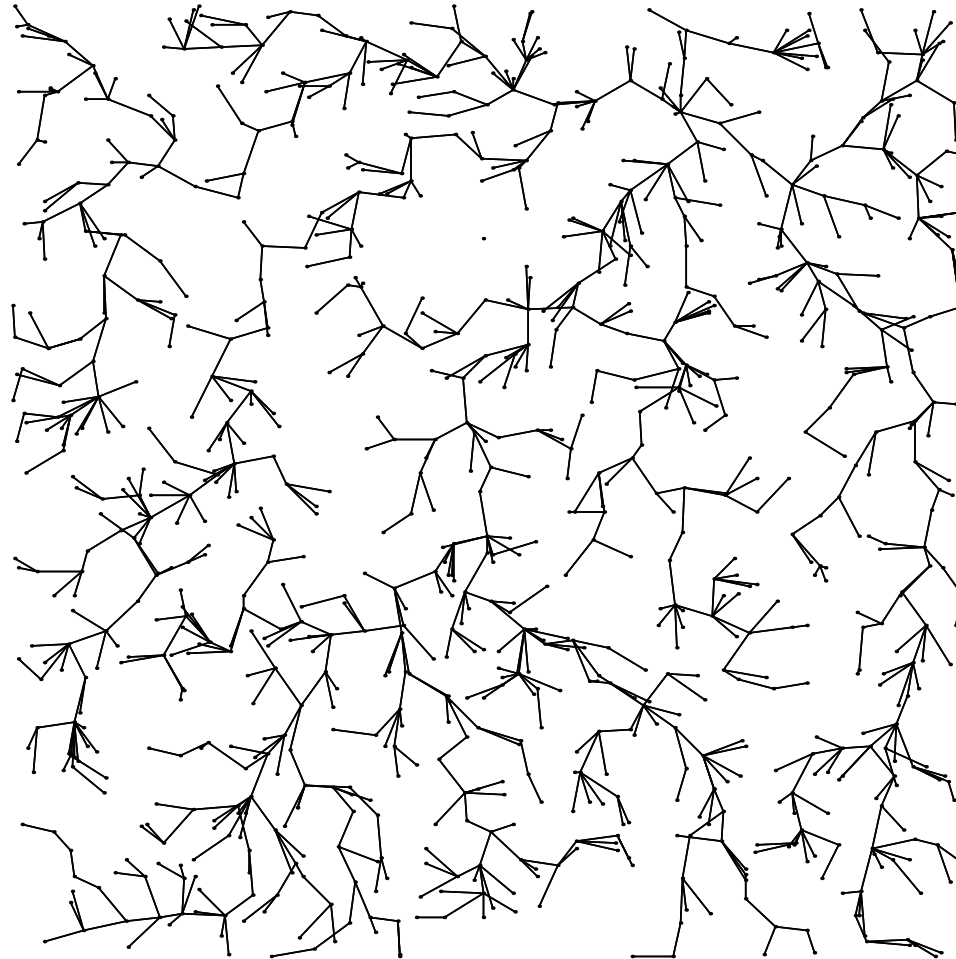
Um grafo geométrico



Um grafo geométrico



Um grafo geométrico



Grafos Dirigidos e Grafos Dirigidos Acíclicos (DAGs)

Sedgewick: Capítulo 19

Grafos, vários sabores

Grafos, vários sabores

Grafos:

1. $G = (V, E)$, onde $E \subset \binom{V}{2}$

Grafos, vários sabores

Grafos:

1. $G = (V, E)$, onde $E \subset \binom{V}{2}$ [quase sempre, V finito]

Grafos, vários sabores

Grafos:

1. $G = (V, E)$, onde $E \subset \binom{V}{2}$ [quase sempre, V finito]
2. Às vezes, consideramos *multigrafos*: arestas múltiplas e laços são permitidos.

Grafos dirigidos, vários sabores

Grafos dirigidos, vários sabores

Grafos dirigidos:

1. $G = (V, E)$, onde $E \subset V \times V$

Grafos dirigidos, vários sabores

Grafos dirigidos:

1. $G = (V, E)$, onde $E \subset V \times V$
2. Às vezes, consideramos multigrafos dirigidos: arcos múltiplos e laços são permitidos (*arco* = “aresta dirigida” ou “orientada”).

Grafos dirigidos, vários sabores

Grafos dirigidos:

1. $G = (V, E)$, onde $E \subset V \times V$
2. Às vezes, consideramos multigrafos dirigidos: arcos múltiplos e laços são permitidos (*arco* = “aresta dirigida” ou “orientada”).

Exemplo: Figura 19.1

Algumas definições

Algumas definições

1. Passeio dirigido, trilha dirigida, caminho dirigido, acessibilidade

Algumas definições

1. Passeio dirigido, trilha dirigida, caminho dirigido, acessibilidade
2. Passeio dirigido fechado, passeio dirigido fechado não-trivial (pelo menos um arco), circuito dirigido

Algumas definições

1. Passeio dirigido, trilha dirigida, caminho dirigido, acessibilidade
2. Passeio dirigido fechado, passeio dirigido fechado não-trivial (pelo menos um arco), circuito dirigido
3. Grafos dirigidos acíclicos (DAGs) = grafos dirigidos sem passeios dirigidos fechados não-triviais

Algumas definições

1. Passeio dirigido, trilha dirigida, caminho dirigido, acessibilidade
2. Passeio dirigido fechado, passeio dirigido fechado não-trivial (pelo menos um arco), circuito dirigido
3. Grafos dirigidos acíclicos (DAGs) = grafos dirigidos sem passeios dirigidos fechados não-triviais
 - ▶ Exemplo básico: relação de precedência
 - ▶ Exemplo: Figura 19.6/19.7

Algumas definições (cont.)

Algumas definições (cont.)

1. Grafos dirigidos fortemente conexos

Algumas definições (cont.)

1. Grafos dirigidos fortemente conexos
2. Relação de equivalência: $x \leftrightarrow y$ (acessibilidade mútua, conexidade forte)

Algumas definições (cont.)

1. Grafos dirigidos fortemente conexos
2. Relação de equivalência: $x \leftrightarrow y$ (acessibilidade mútua, conexidade forte)
3. Classes de equivalência: C_x ($x \in V(G)$)

Algumas definições (cont.)

1. Grafos dirigidos fortemente conexos
2. Relação de equivalência: $x \leftrightarrow y$ (acessibilidade mútua, conexidade forte)
3. Classes de equivalência: C_x ($x \in V(G)$)
4. Componentes fortemente conexas: $G[C_x]$ ($x \in V(G)$)

Algumas definições (cont.)

1. Grafos dirigidos fortemente conexos
2. Relação de equivalência: $x \leftrightarrow y$ (acessibilidade mútua, conexidade forte)
3. Classes de equivalência: C_x ($x \in V(G)$)
4. Componentes fortemente conexas: $G[C_x]$ ($x \in V(G)$)
5. Não necessariamente $G = \bigcup_{x \in V(G)} G[C_x]$

A estrutura de um grafo dirigido

A estrutura de um grafo dirigido

1. O 'núcleo' $K(G)$ de um grafo dirigido G ("condensação de G ")

A estrutura de um grafo dirigido

1. O 'núcleo' $K(G)$ de um grafo dirigido G ("condensação de G ")
2. $K(G)$ é um DAG; exemplo: Figura 19.1/19.8

A estrutura de um grafo dirigido

1. O 'núcleo' $K(G)$ de um grafo dirigido G ("condensação de G ")
2. $K(G)$ é um DAG; exemplo: Figura 19.1/19.8
3. Cortes dirigidos: para $S \subset V(G)$, definimos $\partial^+(S) = \{(x, y) \in E(G) : x \in S \text{ e } y \in V(G) \setminus S\}$. Ademais, $\partial^-(S) = \partial^+(V(G) \setminus S)$

A estrutura de um grafo dirigido

1. O 'núcleo' $K(G)$ de um grafo dirigido G ("condensação de G ")
2. $K(G)$ é um DAG; exemplo: Figura 19.1/19.8
3. Cortes dirigidos: para $S \subset V(G)$, definimos $\partial^+(S) = \{(x, y) \in E(G) : x \in S \text{ e } y \in V(G) \setminus S\}$. Ademais, $\partial^-(S) = \partial^+(V(G) \setminus S)$
4. Grau de entrada, grau de saída; fontes e sorvedouros

A estrutura de um grafo dirigido

1. O 'núcleo' $K(G)$ de um grafo dirigido G ("condensação de G ")
2. $K(G)$ é um DAG; exemplo: Figura 19.1/19.8
3. Cortes dirigidos: para $S \subset V(G)$, definimos $\partial^+(S) = \{(x, y) \in E(G) : x \in S \text{ e } y \in V(G) \setminus S\}$. Ademais, $\partial^-(S) = \partial^+(V(G) \setminus S)$
4. Grau de entrada, grau de saída; fontes e sorvedouros

Figura 19.7

Vários sabores de conexidade

Vários sabores de conexidade

1. Conexidade: grafos não-dirigidos

Vários sabores de conexidade

1. Conexidade: grafos não-dirigidos
2. Acessibilidade: grafos dirigidos, relação não simétrica

Vários sabores de conexidade

1. Conexidade: grafos não-dirigidos
2. Acessibilidade: grafos dirigidos, relação não simétrica
3. Conexidade forte

Representações de grafos dirigidos

Representações de grafos dirigidos

1. Knuth: [Stanford GraphBase](#) ...

Representações de grafos dirigidos

1. Knuth: [Stanford GraphBase](#) ...
2. Sedgewick: matrizes de adjacência e listas de adjacência

Representações de grafos dirigidos

1. Knuth: [Stanford GraphBase](#) ...
2. Sedgewick: matrizes de adjacência e listas de adjacência
 - ▶ **Peculiaridade:** colocamos laços em todos os vértices (pura conveniência)

Representações de grafos dirigidos

1. Knuth: [Stanford GraphBase](#) ...
2. Sedgewick: matrizes de adjacência e listas de adjacência
 - ▶ **Peculiaridade:** colocamos laços em todos os vértices (pura conveniência)
 - ▶ Assim: 1s na diagonal das mzs de adj. e ao menos uma célula em cada lista de adj.

Programa 19.1

```
Graph GRAPHreverse(Graph G)
{ int v; link t;
  Graph R = GRAPHinit(G->V);
  for (v = 0; v < G->V; v++)
    for (t = G->adj[v]; t != NULL; t = t->next)
      GRAPHinsertE(R, EDGE(t->v, v));
  return R;
}
```