Busca em largura e busca generalizada em grafos

Busca em largura e busca generalizada em grafos

1. Busca em profundidade

Busca em largura e busca generalizada em grafos

1. Busca em profundidade

2. Busca em largura

Busca em largura e busca generalizada em grafos

1. Busca em profundidade

2. Busca em largura

Diferença crucial:

Busca em largura e busca generalizada em grafos

1. Busca em profundidade

2. Busca em largura

Diferença crucial: **pilhas** $\times$ **filas**

Busca em largura e busca generalizada em grafos

1. Busca em profundidade

2. Busca em largura

Diferença crucial: **pilhas** $\times$ **filas**

- ▷ Busca generalizada: políticas generalizadas de processamento de arestas

Busca em largura e busca generalizada em grafos

1. Busca em profundidade

2. Busca em largura

Diferença crucial: **pilhas** $\times$ **filas**

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

Um exemplo de busca em largura

Um exemplo de busca em largura

```
yoshi@erdos:~/Main/www/2004i/mac328/exx/sec18.7$ a.out 8 13 < ../figs
8 vertices, 10 edges
 0:  5  7  2
 1:  7
 2:  6  0
 3:  5  4
 4:  7  6  5  3
 5:  4  3  0
 6:  4  2
 7:  4  1  0
    v: 0 1 2 3 4 5 6 7
pre[]: 0 6 3 5 4 1 7 2
 st[]: 0 7 0 5 5 0 2 0
```

Programa 18.8

Programa 18.8

```
void bfs(Graph G, Edge e)
  { int v;
    QUEUEput(e);
    while (!QUEUEempty())
      if (pre[(e = QUEUEget()).w] == -1)
        { link t;
          pre[e.w] = cnt++; st[e.w] = e.v;
          for (t = G->adj[e.w]; t != NULL; t = t->next)
            if (pre[v = t->v] == -1)
              QUEUEput(EDGE(e.w, v));
        }
  }
```

Propriedade fundamental da busca em largura

Propriedade fundamental da busca em largura

Os vértices de G são descobertos em uma ordem que respeita a distância ao vértice de onde se inicia a busca.

Propriedade fundamental da busca em largura

Os vértices de G são descobertos em uma ordem que respeita a distância ao vértice de onde se inicia a busca.

Se $d(x_0, u) < d(x_0, v)$, então `pre[u] < pre[v]` (busca iniciada no vértice x_0).

Propriedade fundamental da busca em largura

Os vértices de G são descobertos em uma ordem que respeita a distância ao vértice de onde se inicia a busca.

Se $d(x_0, u) < d(x_0, v)$, então `pre[u] < pre[v]` (busca iniciada no vértice x_0).

Distância entre a e $b \in V(G)$:

Propriedade fundamental da busca em largura

Os vértices de G são descobertos em uma ordem que respeita a distância ao vértice de onde se inicia a busca.

Se $d(x_0, u) < d(x_0, v)$, então `pre[u] < pre[v]` (busca iniciada no vértice x_0).

Distância entre a e $b \in V(G)$:

$$d(a, b) = d_G(a, b) = \min\{|E(P)| : P \text{ } a\text{–}b \text{ caminho em } G\}.$$

Propriedade fundamental da busca em largura

Os vértices de G são descobertos em uma ordem que respeita a distância ao vértice de onde se inicia a busca.

Se $d(x_0, u) < d(x_0, v)$, então `pre[u] < pre[v]` (busca iniciada no vértice x_0).

Distância entre a e $b \in V(G)$:

$$d(a, b) = d_G(a, b) = \min\{|E(P)| : P \text{ } a\text{–}b \text{ caminho em } G\}.$$

Podemos ainda generalizar para A, $B \subset V(G)$ da forma usual:

Propriedade fundamental da busca em largura

Os vértices de G são descobertos em uma ordem que respeita a distância ao vértice de onde se inicia a busca.

Se $d(x_0, u) < d(x_0, v)$, então `pre[u] < pre[v]` (busca iniciada no vértice x_0).

Distância entre a e $b \in V(G)$:

$$d(a, b) = d_G(a, b) = \min\{|E(P)| : P \text{ } a\text{–}b \text{ caminho em } G\}.$$

Podemos ainda generalizar para A, $B \subset V(G)$ da forma usual:

$$d(A, B) = d_G(A, B) = \min\{d(a, b) : a \in A,\ b \in B\}.$$

Propriedade fundamental da busca em largura

Os vértices de G são descobertos em uma ordem que respeita a distância ao vértice de onde se inicia a busca.

Se $d(x_0, u) < d(x_0, v)$, então `pre[u] < pre[v]` (busca iniciada no vértice x_0).

Distância entre a e $b \in V(G)$:

$$d(a, b) = d_G(a, b) = \min\{|E(P)| : P \text{ } a\text{–}b \text{ caminho em } G\}.$$

Podemos ainda generalizar para A, $B \subset V(G)$ da forma usual:

$$d(A, B) = d_G(A, B) = \min\{d(a, b) : a \in A,\ b \in B\}.$$

(Por simplicidade, consideramos apenas grafos conexos.)

Distância em um grafo G

Distância em um grafo G

$d = d_G$ definida acima é uma *métrica* sobre $V(G)$, isto é, satisfaz as seguintes propriedades: para quaisquer x, y, e z, temos

Distância em um grafo G

$d = d_G$ definida acima é uma *métrica* sobre $V(G)$, isto é, satisfaz as seguintes propriedades: para quaisquer x, y, e z, temos

1. $d(x, y) = 0$ se e só se $x = y$,

Distância em um grafo G

$d = d_G$ definida acima é uma *métrica* sobre $V(G)$, isto é, satisfaz as seguintes propriedades: para quaisquer x, y, e z, temos

1. $d(x, y) = 0$ se e só se $x = y$,

2. $d(x, y) = d(y, x)$,

Distância em um grafo G

$d = d_G$ definida acima é uma *métrica* sobre $V(G)$, isto é, satisfaz as seguintes propriedades: para quaisquer x, y, e z, temos

1. $d(x, y) = 0$ se e só se $x = y$,

2. $d(x, y) = d(y, x)$,

3. $d(x, z) \leq d(x, y) + d(y, z)$.

Árvore de busca em largura

Árvore de busca em largura

Os caminhos da raiz até os vértices da árvore de BL são caminhos de comprimento mínimo.

Árvore de busca em largura

Os caminhos da raiz até os vértices da árvore de BL são caminhos de comprimento mínimo.

Em particular, se x_0 é a raiz da ABL T e $u \in V(G)$, então $d_G(x_0, u) = d_T(x_0, u)$.

Complexidade da busca em largura

Complexidade da busca em largura

Com a representação de G com listas de adjacência:

Complexidade da busca em largura

Com a representação de G com listas de adjacência: $O(n + m)$

Programa 18.9 (Variante do Programa 18.8)

Programa 18.9 (Variante do Programa 18.8)

```
void bfs(Graph G, Edge e)
  { int v, w;
    QUEUEput(e); pre[e.w] = cnt++;
    while (!QUEUEempty())
      {
        e = QUEUEget();
        w = e.w; st[w] = e.v;
        for (t = G->adj[w]; t != NULL; t = t->next)
          if (pre[v = t->v] == -1)
            { QUEUEput(EDGE(w, v)); pre[v] = cnt++; }
      }
  }
```

Programa 18.8 (Recap.)

```
void bfs(Graph G, Edge e)
  { int v;
    QUEUEput(e);
    while (!QUEUEempty())
      if (pre[(e = QUEUEget()).w] == -1)
        { link t;
          pre[e.w] = cnt++; st[e.w] = e.v;
          for (t = G->adj[e.w]; t != NULL; t = t->next)
            if (pre[v = t->v] == -1)
              QUEUEput(EDGE(e.w, v));
        }
  }
```

Vantagem do Programa 18.9?

Vantagem do Programa 18.9?

Tamanho máximo da fila?

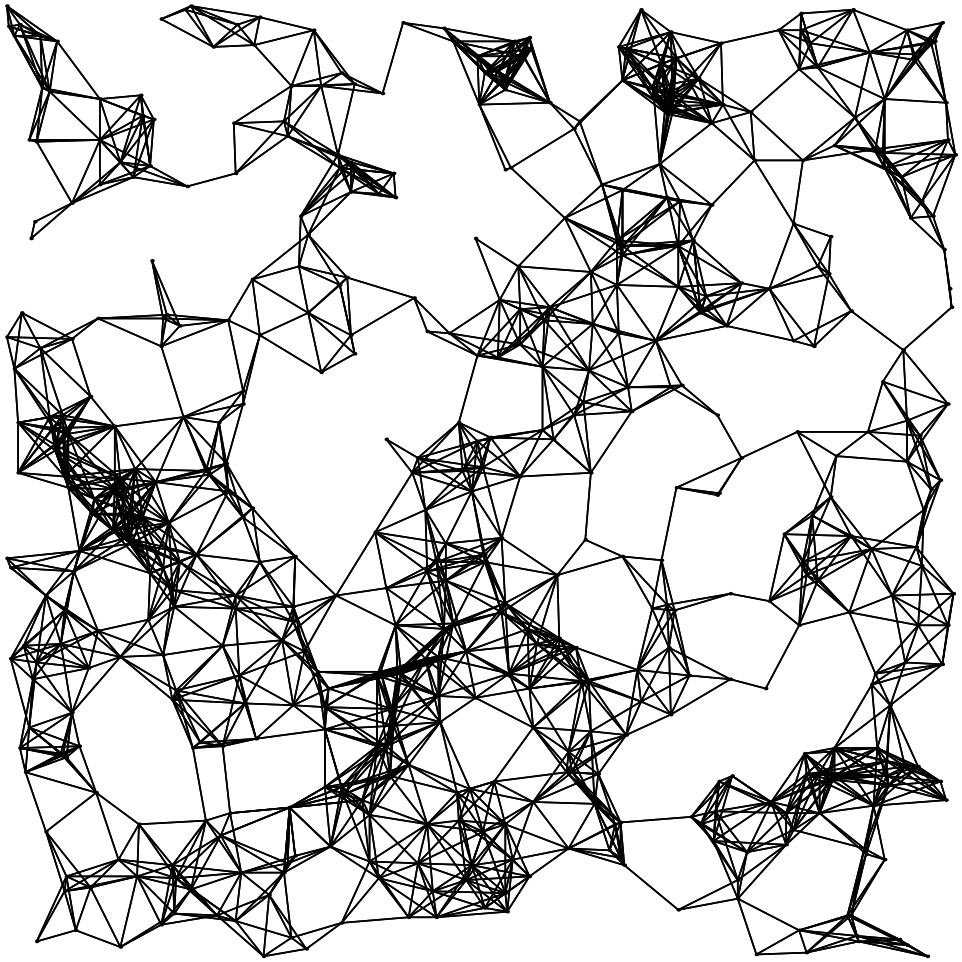
Vantagem do Programa 18.9?

Tamanho máximo da fila?

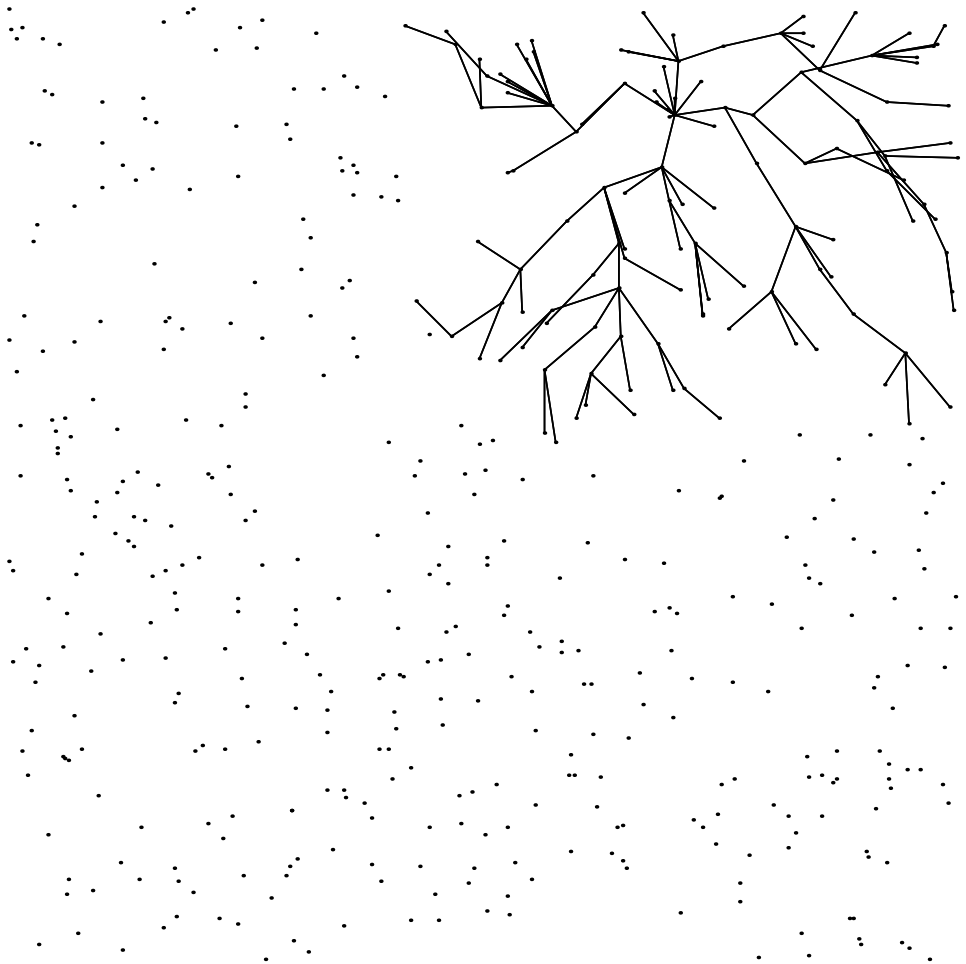
Nunca maior que $n = |V(G)|$ elementos!

Um grafo geométrico

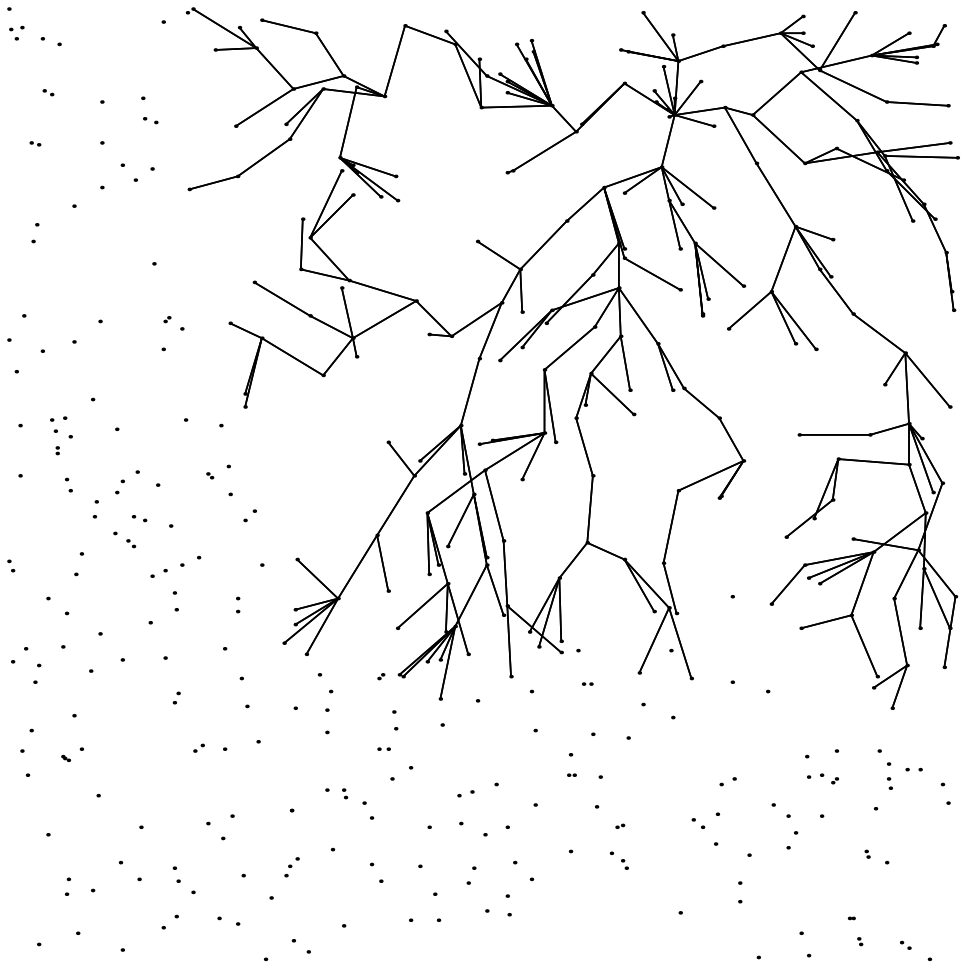
Um grafo geométrico



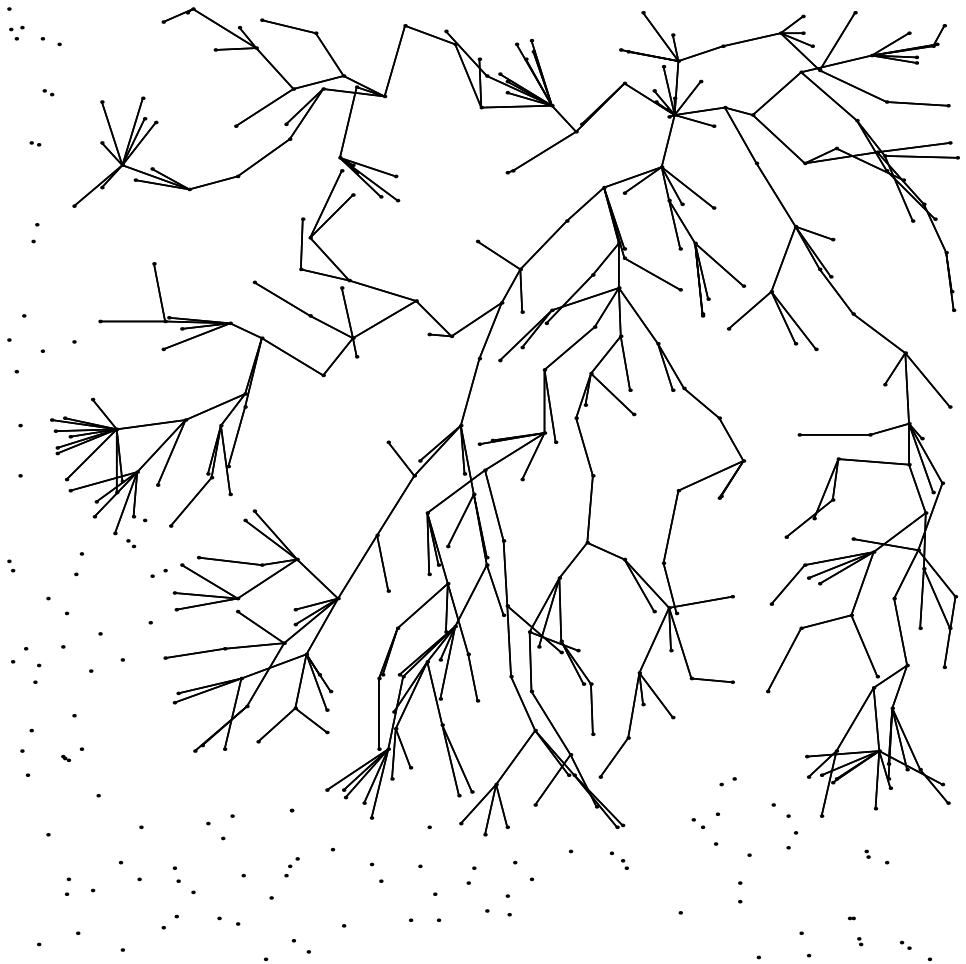
Uma ABL do grafo geométrico; $n/4$



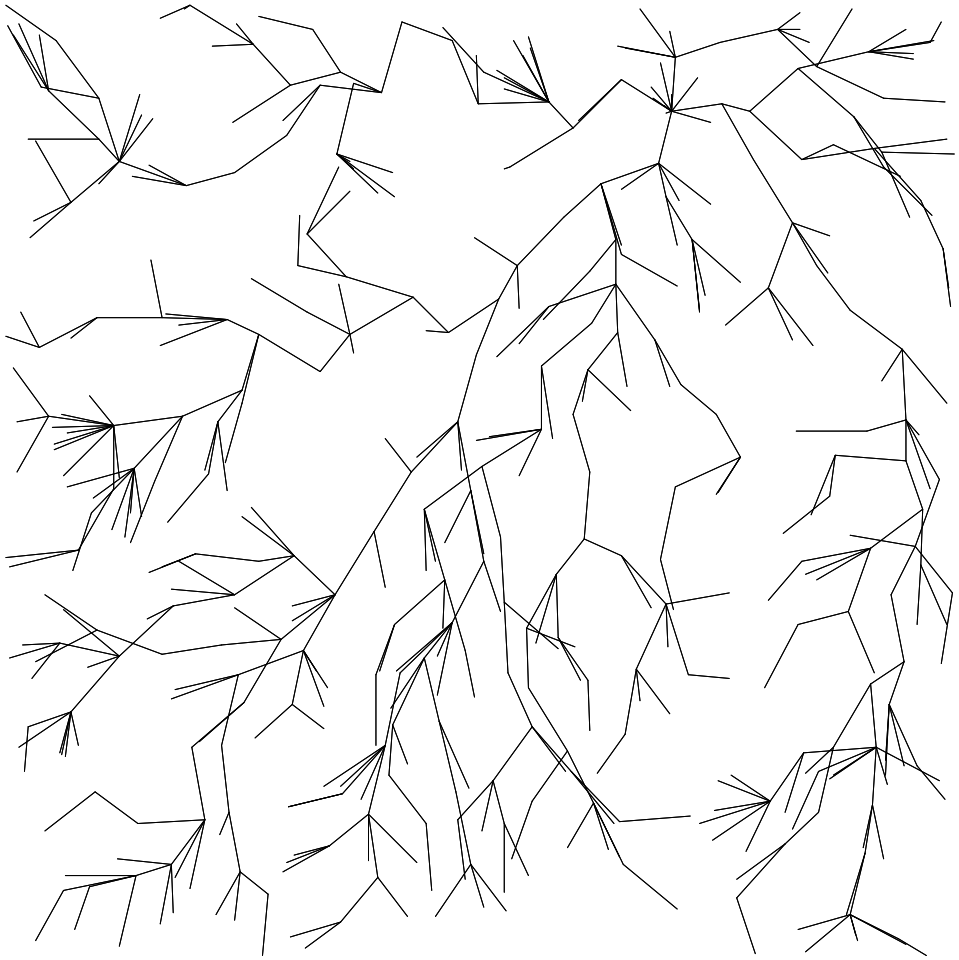
Uma ABL do grafo geométrico; $n/2$



Uma ABL do grafo geométrico; $3n/2$

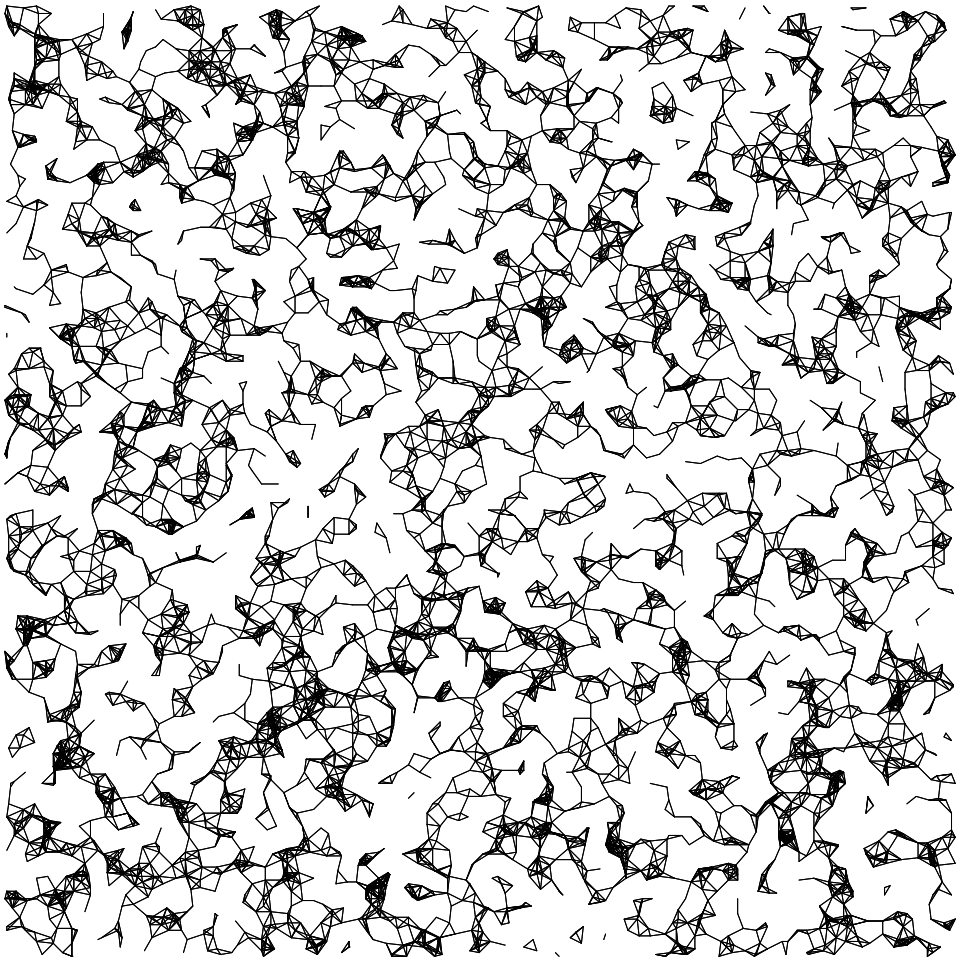


Uma ABL do grafo geométrico

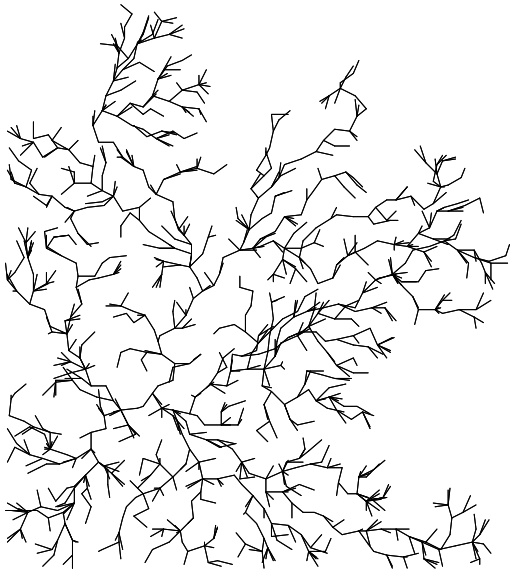


Um grafo geométrico

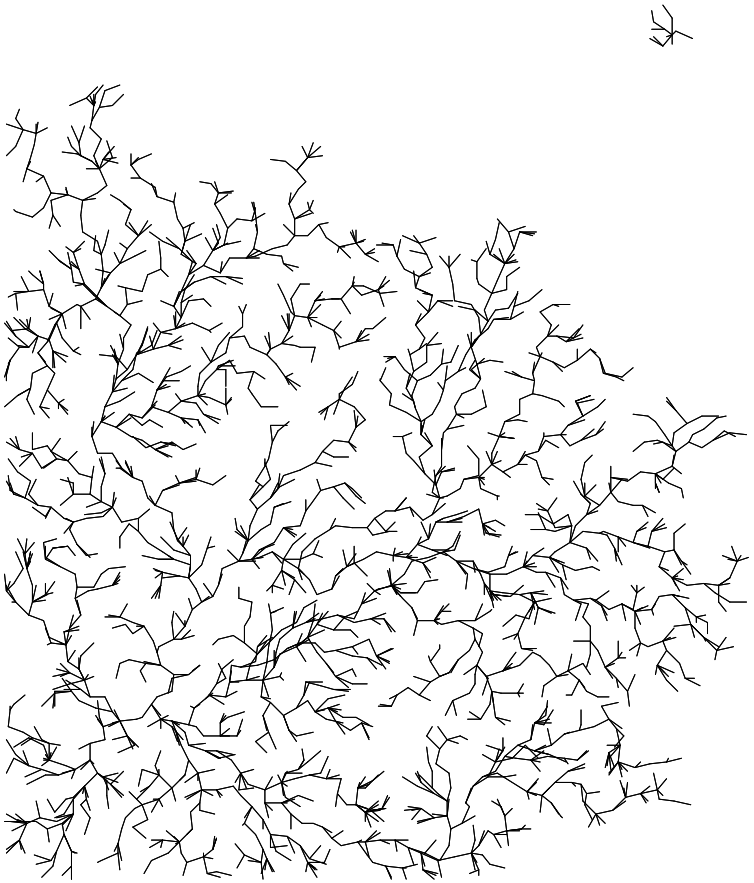
Um grafo geométrico



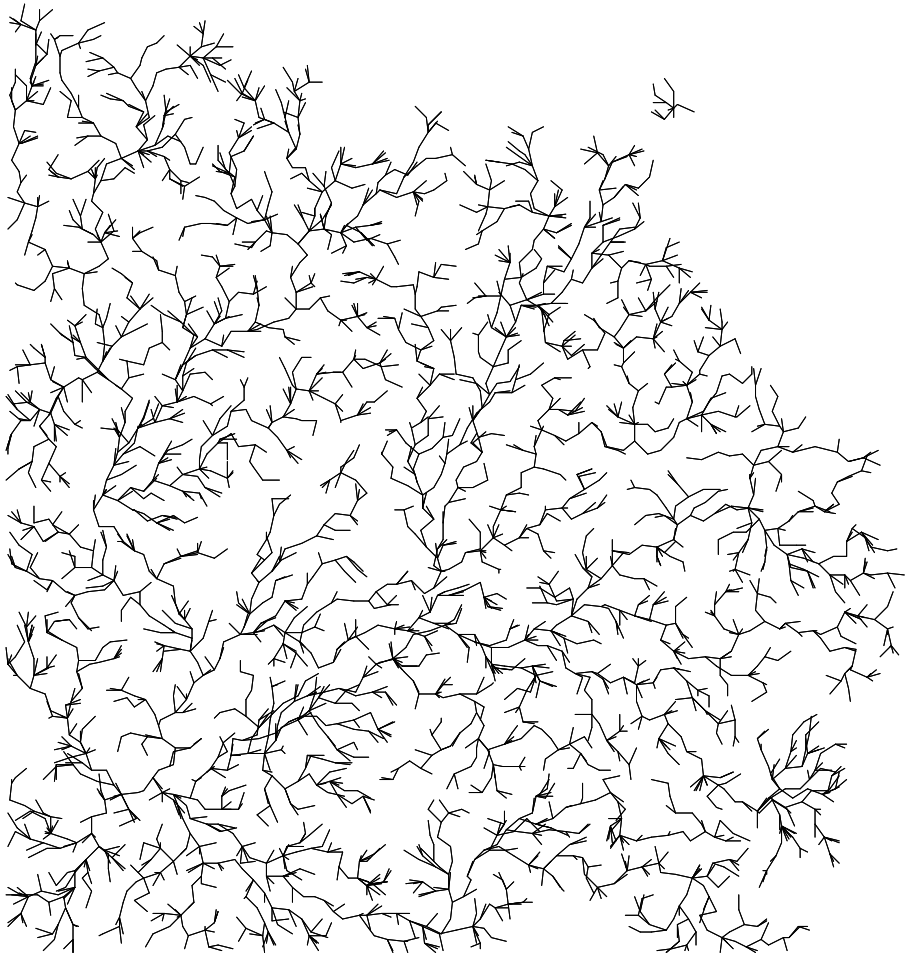
Uma ABL do grafo geométrico; $n/4$



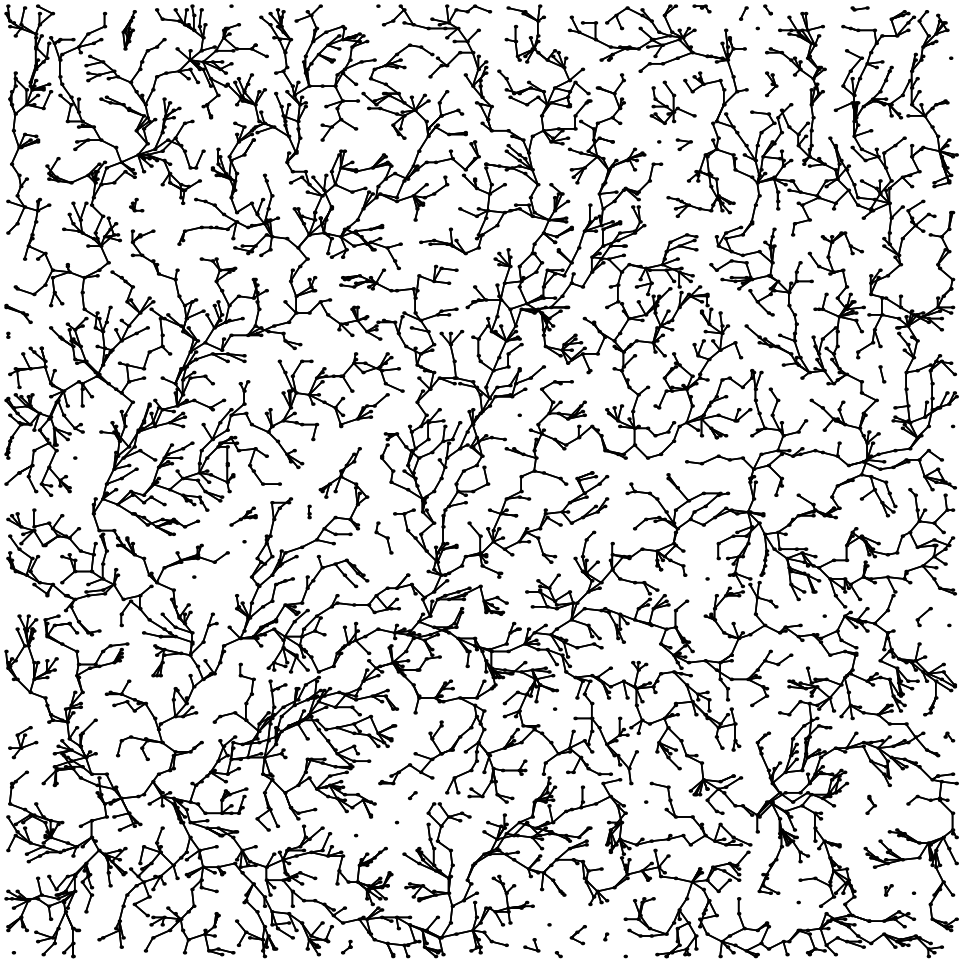
Uma ABL do grafo geométrico; $n/2$



Uma ABL do grafo geométrico; $3n/2$



Uma ABL do grafo geométrico



Usos da busca em largura

Usos da busca em largura

1. Distância/caminho mais curto entre x e y

Usos da busca em largura

1. Distância/caminho mais curto entre x e y
2. Distância/caminho mais curto de x e *todo* y

Usos da busca em largura

1. Distância/caminho mais curto entre x e y
2. Distância/caminho mais curto de x e *todo* y
3. Distância/caminho mais curto entre todos os pares x e y

   ▷ Complexidade?

Usos da busca em largura

1. Distância/caminho mais curto entre x e y
2. Distância/caminho mais curto de x e *todo* y
3. Distância/caminho mais curto entre todos os pares x e y

   ▷ Complexidade? Tempo: $O(n(n+m))$;

Usos da busca em largura

1. Distância/caminho mais curto entre x e y
2. Distância/caminho mais curto de x e *todo* y
3. Distância/caminho mais curto entre todos os pares x e y

   ▷ Complexidade? Tempo: $O(n(n+m))$; espaço: $O(n^2)$ (para armazenar as distâncias e as ABLs (os `st[]`))