

Busca em profundidade em grafos

Busca em profundidade em grafos

Busca em profundidade: várias informações estruturais sobre o grafo são obtidas.

Busca em profundidade em grafos

Busca em profundidade: várias informações estruturais sobre o grafo são obtidas.

- Óbvio:

Busca em profundidade em grafos

Busca em profundidade: várias informações estruturais sobre o grafo são obtidas.

- Óbvio: número de componentes (conexos)

Busca em profundidade em grafos

Busca em profundidade: várias informações estruturais sobre o grafo são obtidas.

- Óbvio: número de componentes (conexos)
 1. Existência de um caminho de v a w

Busca em profundidade em grafos

Busca em profundidade: várias informações estruturais sobre o grafo são obtidas.

- Óbvio: número de componentes (conexos)
 1. Existência de um caminho de v a w
 2. Número de vértices no componente de um dado vértice

Busca em profundidade em grafos (cont.)

Busca em profundidade em grafos (cont.)

1. Árvore geradora:

Busca em profundidade em grafos (cont.)

1. Árvore geradora: subárvore de G , com todos os vértices de G (G conexo)

Busca em profundidade em grafos (cont.)

1. Árvore geradora: subárvore de G , com todos os vértices de G (G conexo)
2. Detecção de circuitos

Busca em profundidade em grafos (cont.)

1. Árvore geradora: subárvore de G , com todos os vértices de G (G conexo)
2. Detecção de circuitos [existência:

Busca em profundidade em grafos (cont.)

1. Árvore geradora: subárvore de G , com todos os vértices de G (G conexo)
2. Detecção de circuitos [existência: trivial! (se sabemos o número de componentes e número de vértices e arestas)]
3. 2-colorabilidade/existência de circuitos ímpares

Busca em profundidade em grafos (cont.)

1. Árvore geradora: subárvore de G , com todos os vértices de G (G conexo)
2. Detecção de circuitos [existência: trivial! (se sabemos o número de componentes e número de vértices e arestas)]
3. 2-colorabilidade/existência de circuitos ímpares

Tudo isso em tempo linear $O(|V(G)| + |E(G)|)$!

Busca em profundidade em grafos (cont.)

1. Árvore geradora: subárvore de G , com todos os vértices de G (G conexo)
2. Detecção de circuitos [existência: trivial! (se sabemos o número de componentes e número de vértices e arestas)]
3. 2-colorabilidade/existência de circuitos ímpares

Tudo isso em tempo linear $O(|V(G)| + |E(G)|)$! (Ou menos, para alguns problemas)

A árvore da busca em profundidade

A árvore da busca em profundidade

1. Árvore geradora

A árvore da busca em profundidade

1. Árvore geradora
2. Conjunto de arestas:

A árvore da busca em profundidade

1. Árvore geradora
2. Conjunto de arestas: todas as arestas $\{w, t \rightarrow v\}$ para as quais executamos $\text{dfsR}(G, \text{EDGE}(w, t \rightarrow v))$ em $\text{dfsR}()$

Programa 18.2

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

Um grafo geométrico (“near neighbour graph”)

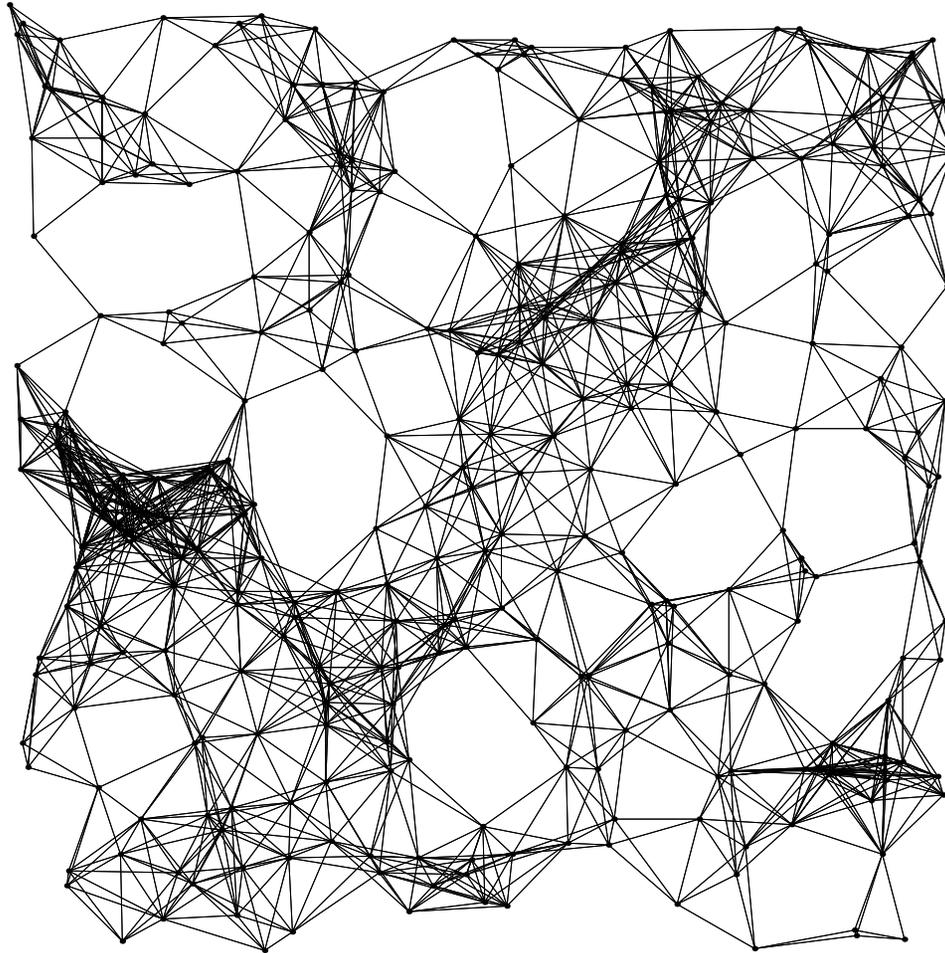
Um grafo geométrico (“near neighbour graph”)

- n vértices uniformemente sorteados em um quadrado

Um grafo geométrico (“near neighbour graph”)

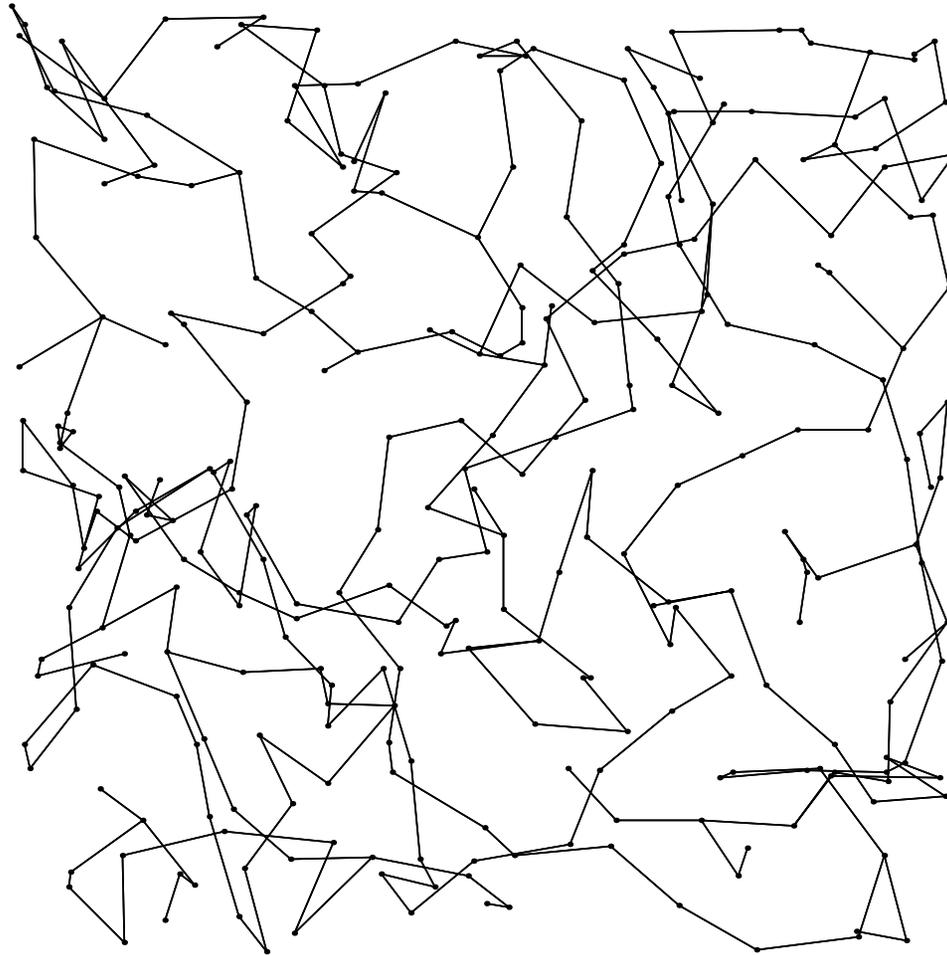
- n vértices uniformemente sorteados em um quadrado
- arestas ligam pares de vértices à distância $< d$

300 vértices, $d = 60$ (quadrado: 512×512)



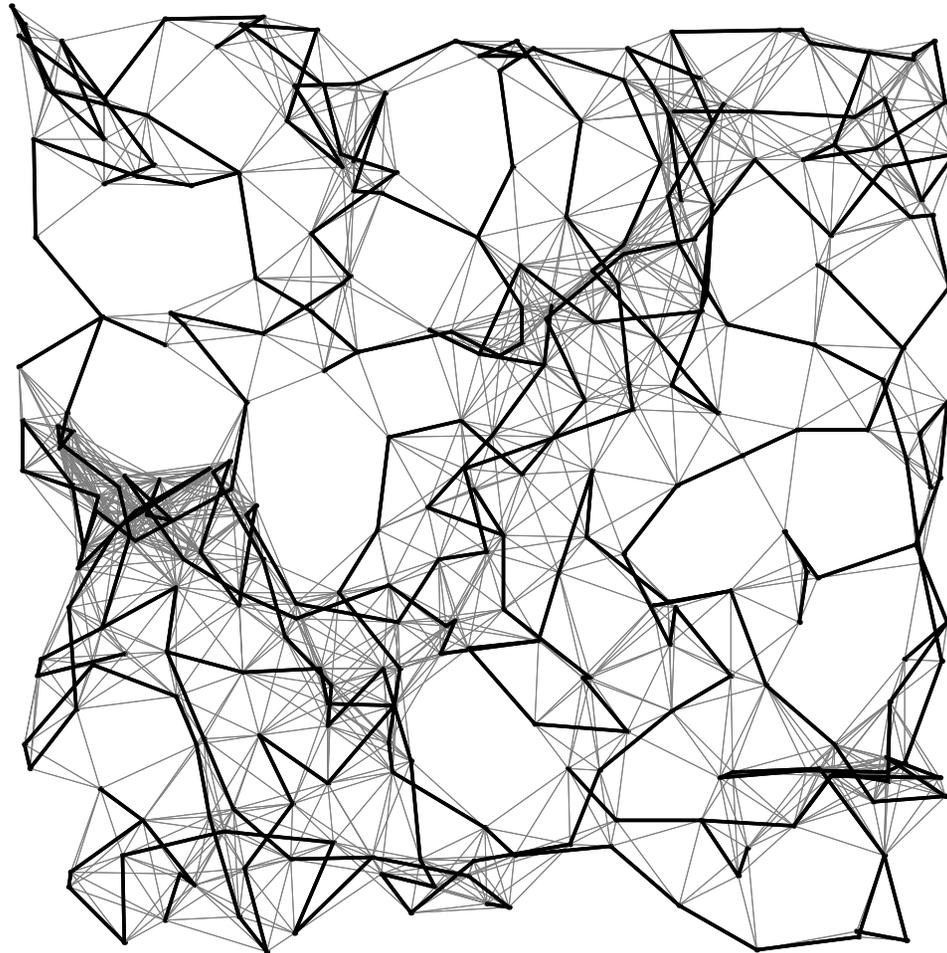
Uma ABP (DFS tree)

Uma ABP (DFS tree)

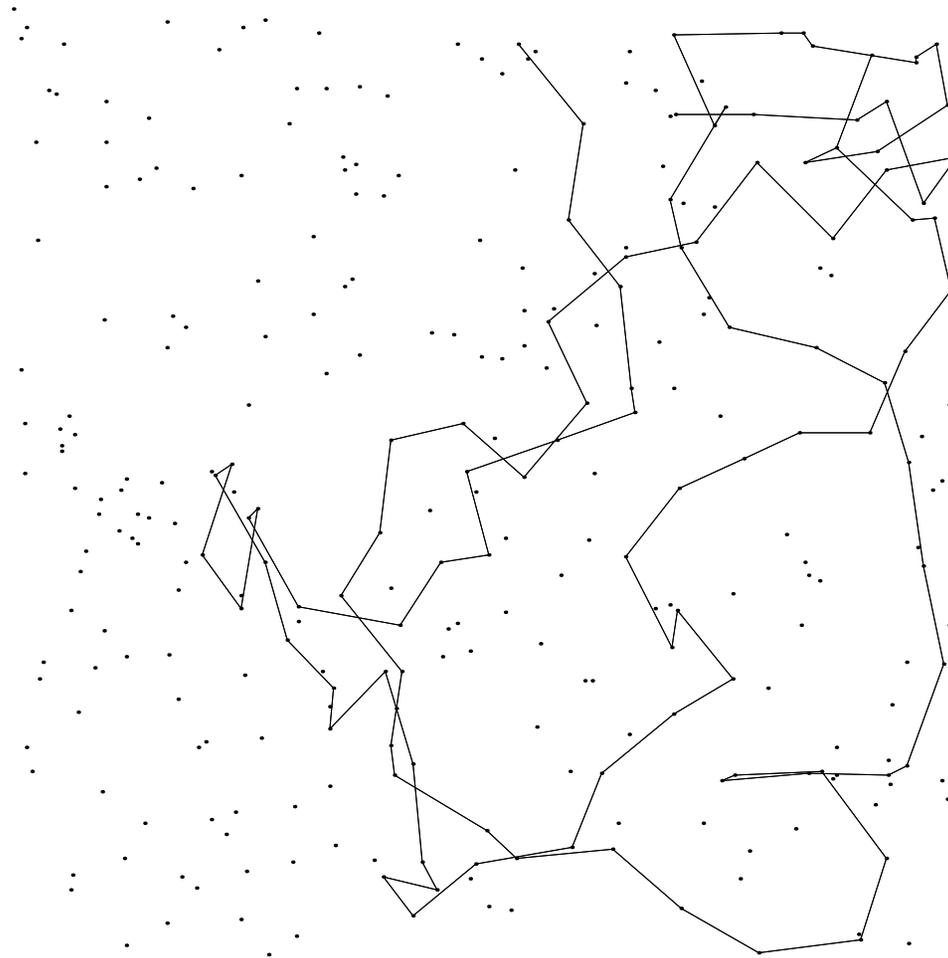


O grafo e a ABP (DFS tree)

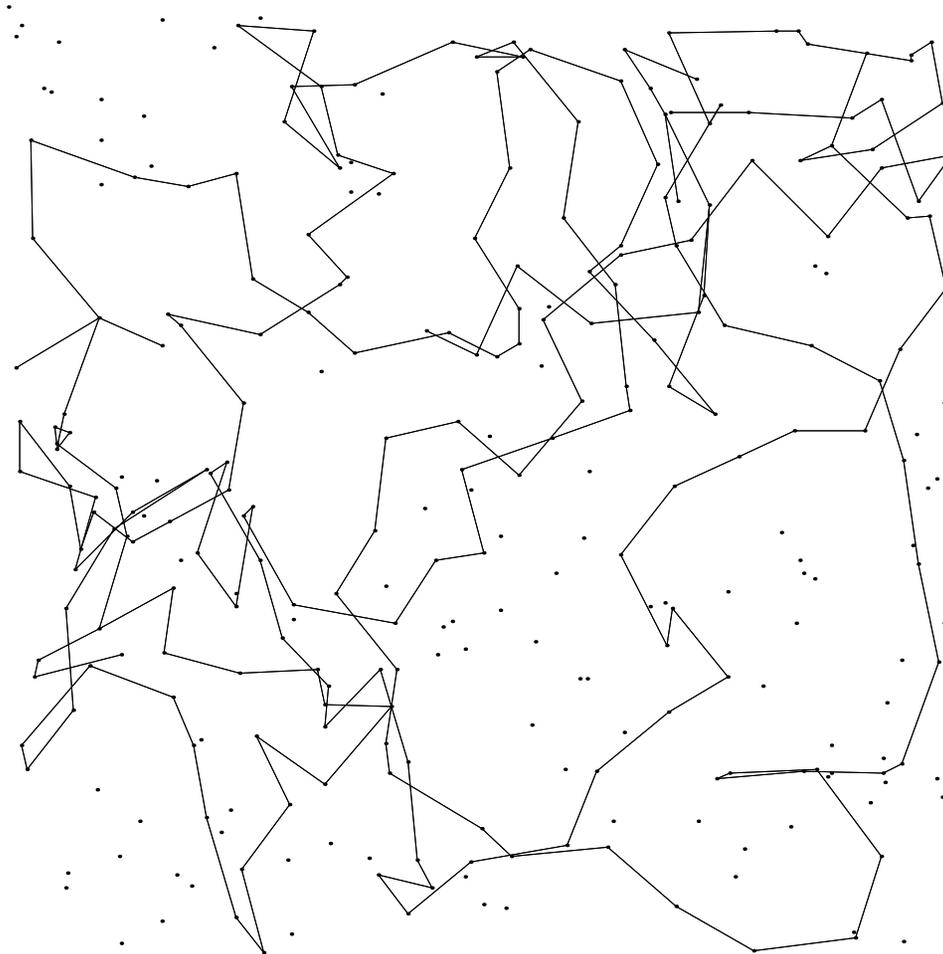
O grafo e a ABP (DFS tree)



Evolução da ABP (1/3)

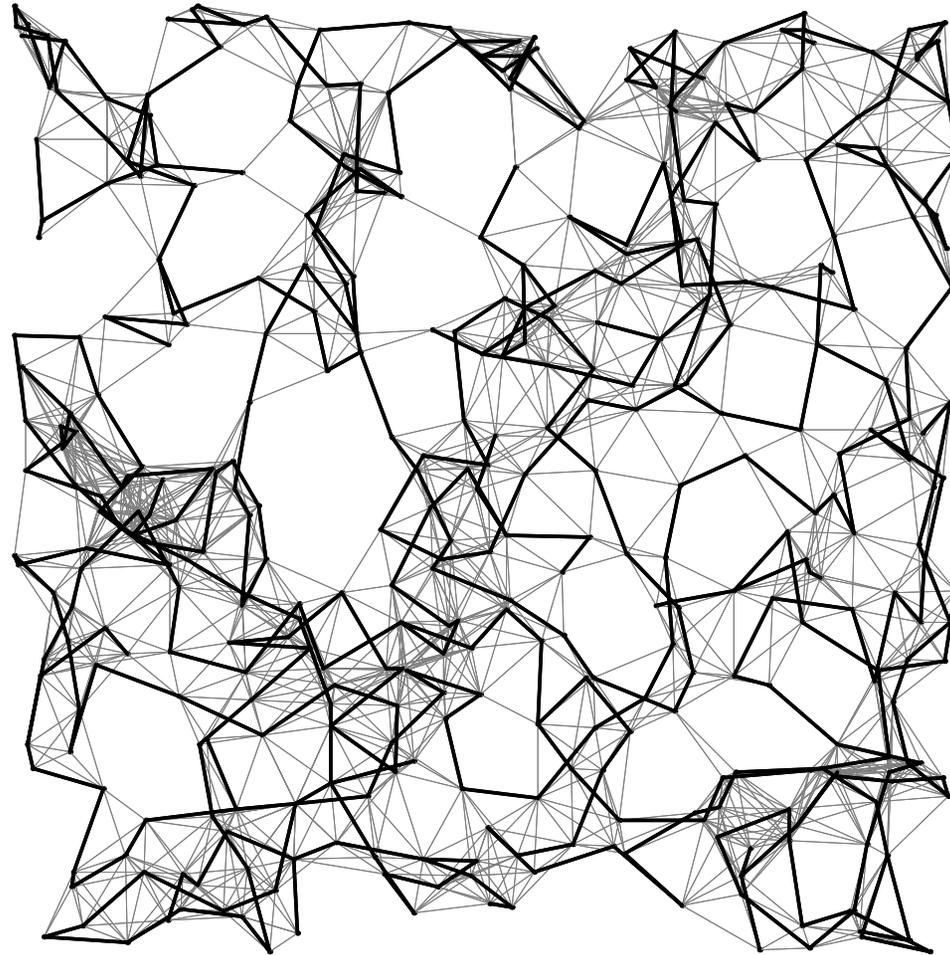


Evolução da ABP (2/3)



Outro grafo e uma ABP (DFS tree)

Outro grafo e uma ABP (DFS tree)



Classificação das arestas de G em relação à ABP T

Classificação das arestas de G em relação à ABP T

- Arestas presentes em T

Classificação das arestas de G em relação à ABP T

- Arestas presentes em T
- Arestas fora de T :

Classificação das arestas de G em relação à ABP T

- Arestas presentes em T
- Arestas fora de T : ligam vértices a ancestrais diferentes do “pai”

Classificação das arestas de G em relação à ABP T

- Arestas presentes em T
- Arestas fora de T : ligam vértices a ancestrais diferentes do “pai”
(Exercício!)

Classificação das arestas de G em relação à ABP T

- Arestas presentes em T
- Arestas fora de T : ligam vértices a ancestrais diferentes do “pai”
(Exercício!)
- Vetor $st []$: indica os “pais”

Classificação mais fina

Classificação mais fina

4 tipos para cada par ordenado (v, w) (*ponteiro/link*):

Classificação mais fina

4 tipos para cada par ordenado (v, w) (*ponteiro/link*):

1. tipo *árvore* se w tinha $\text{pre}[w] = -1$ ($\{v, w\}$ em T)

Classificação mais fina

4 tipos para cada par ordenado (v, w) (*ponteiro/link*):

1. tipo *árvore* se w tinha $\text{pre}[w] = -1$ ($\{v, w\}$ em T)

2. tipo *pai* se $\text{st}[w] = v$ ($\{v, w\}$ em T)

Classificação mais fina

4 tipos para cada par ordenado (v, w) (*ponteiro/link*):

1. tipo *árvore* se w tinha $\text{pre}[w] = -1$ ($\{v, w\}$ em T)
2. tipo *pai* se $\text{st}[w] = v$ ($\{v, w\}$ em T)
3. tipo *ascendente (back)* se $\text{pre}[w] < \text{pre}[v]$ ($\{v, w\}$ fora de T)

Classificação mais fina

4 tipos para cada par ordenado (v, w) (*ponteiro/link*):

1. tipo *árvore* se w tinha $\text{pre}[w] = -1$ ($\{v, w\}$ em T)
2. tipo *pai* se $\text{st}[w] = v$ ($\{v, w\}$ em T)
3. tipo *ascendente (back)* se $\text{pre}[w] < \text{pre}[v]$ ($\{v, w\}$ fora de T)
4. tipo *descendente (down)* se $\text{pre}[w] > \text{pre}[v]$ ($\{v, w\}$ fora de T)

Um exemplo

Um exemplo

```
yoshi@erdos:~/Main/www/2004i/mac328/exx/sec18.4$ a.out 8 10 < fig18.9
```

```
8 vertices, 10 edges
```

```
0: 7 5 2
```

```
1: 7
```

```
2: 6 0
```

```
3: 5 4
```

```
4: 5 7 3 6
```

```
5: 0 4 3
```

```
6: 4 2
```

```
7: 0 1 4
```

```
pre[]: 0 2 7 5 3 4 6 1
```

```
st[]: 0 7 6 5 7 4 4 0
```

A ABP e a classificação

A ABP e a classificação

0-0 tree

0-7 tree

7-0 parent

7-1 tree

1-7 parent

7-4 tree

4-5 tree

5-0 back

5-4 parent

5-3 tree

3-5 parent

3-4 back

A ABP e a classificação (cont.)

A ABP e a classificação (cont.)

5-3 tree

3-5 parent

3-4 back

4-7 parent

4-3 down

4-6 tree

6-4 parent

6-2 tree

2-6 parent

2-0 back

0-5 down

0-2 down

Algoritmos baseados em BP

Algoritmos baseados em BP

- Detecção de circuitos

Algoritmos baseados em BP

- Detecção de circuitos [existência:

Algoritmos baseados em BP

- Detecção de circuitos [existência: trivial! (se sabemos o número de componentes e número de vértices e arestas)]

Algoritmos baseados em BP

- Detecção de circuitos [existência: trivial! (se sabemos o número de componentes e número de vértices e arestas)]
 - Tempo: $O(|V(G)|)$

Algoritmos baseados em BP (cont.)

Algoritmos baseados em BP (cont.)

- 2-colorabilidade/não existência de circuitos ímpares

Algoritmos baseados em BP (cont.)

- 2-colorabilidade/não existência de circuitos ímpares
 - Equivalentes!

Algoritmos baseados em BP (cont.)

- 2-colorabilidade/não existência de circuitos ímpares
 - Equivalentes! Uma direção é óbvia; a outra exige prova

Algoritmos baseados em BP (cont.)

- 2-colorabilidade/não existência de circuitos ímpares
 - Equivalentes! Uma direção é óbvia; a outra exige prova
 - “Grafos bipartidos/biparticionáveis”

2-colorabilidade \times circuitos ímpares

2-colorabilidade \times circuitos ímpares

- Uma prova:

2-colorabilidade \times circuitos ímpares

- Uma prova: seja G um grafo minimal não 2-colorível

2-colorabilidade \times circuitos ímpares

- Uma prova: seja G um grafo minimal não 2-colorível (G não 2-colorível, $G - e$ 2-colorível; e não é de corte; etc.)
- Outra prova, baseada na BP:

Programa 18.6

Programa 18.6

```
int dfsRcolor(Graph G, int v, int c)
{ link t;
  G->color[v] = 1-c;
  for (t = G->adj[v]; t != NULL; t = t->next)
    if (G->color[t->v] == -1)
      { if (!dfsRcolor(G, t->v, 1-c)) return 0; }
    else if (G->color[t->v] != c) return 0;
  return 1;
}
```

Programa 18.6 (cont.)

Programa 18.6 (cont.)

```
int GRAPHtwocolor(Graph G)
{ int v, id = 0;
  G->color = malloc(G->V * sizeof(int));
  for (v = 0; v < G->V; v++)
    G->color[v] = -1;
  for (v = 0; v < G->V; v++)
    if (G->color[v] == -1)
      if (!dfsRcolor(G, v, 0)) return 0;
  return 1;
}
```

2-colorabilidade \times circuitos ímpares

2-colorabilidade \times circuitos ímpares

- Uma prova: seja G um grafo minimal não 2-colorível. ($G \times G - e$; e não é de corte; etc.)
- Outra prova, baseada na BP:

2-colorabilidade \times circuitos ímpares

- Uma prova: seja G um grafo minimal não 2-colorível. ($G \times G - e$; e não é de corte; etc.)
- Outra prova, baseada na BP: prove a correção do programa 18.6 por indução