

Complexidade de alguns problemas envolvendo grafos

Complexidade de alguns problemas envolvendo grafos

Classificação em quatro classes:

1. Fáceis

Complexidade de alguns problemas envolvendo grafos

Classificação em quatro classes:

1. Fáceis
2. Tratáveis

Complexidade de alguns problemas envolvendo grafos

Classificação em quatro classes:

1. Fáceis
2. Tratáveis
3. Intratáveis

Complexidade de alguns problemas envolvendo grafos

Classificação em quatro classes:

1. Fáceis
2. Tratáveis
3. Intratáveis
4. Problemas de complexidade desconhecida

Hipótese padrão: $P \neq NP$

Hipótese padrão: $P \neq NP$

Classes de complexidade fundamentais para problemas de decisão (problemas com resposta “sim”/“não”):

1. P : problemas que podem ser **resolvidos** por um algoritmo com complexidade de tempo polinomial

Hipótese padrão: $P \neq NP$

Classes de complexidade fundamentais para problemas de decisão (problemas com resposta “sim”/“não”):

1. P : problemas que podem ser **resolvidos** por um algoritmo com complexidade de tempo polinomial
2. NP : problemas para os quais a resposta “sim” admite uma **testemunha verificável em tempo polinomial**

Hipótese padrão: $P \neq NP$

Classes de complexidade fundamentais para problemas de decisão (problemas com resposta “sim”/“não”):

1. P : problemas que podem ser **resolvidos** por um algoritmo com complexidade de tempo polinomial
2. NP : problemas para os quais a resposta “sim” admite uma **testemunha verificável em tempo polinomial**
3. $P \subset NP$

Hipótese padrão: $P \neq NP$

Classes de complexidade fundamentais para problemas de decisão (problemas com resposta “sim”/“não”):

1. P : problemas que podem ser **resolvidos** por um algoritmo com complexidade de tempo polinomial
2. NP : problemas para os quais a resposta “sim” admite uma **testemunha verificável em tempo polinomial**
3. $P \subset NP$
4. Problema fundamental: $P \neq NP$?

As classes P e NP

As classes P e NP

Exemplos:

(CAM) Dados $G = (V, A)$ e $v, w \in V$, existe um $v-w$ caminho em G ?

As classes P e NP

Exemplos:

(CAM) Dados $G = (V, A)$ e $v, w \in V$, existe um $v-w$ caminho em G ?

(HAM) Dados $G = (V, A)$ e $v, w \in V$, existe um $v-w$ caminho em G que é hamiltoniano?

As classes P e NP

As classes P e NP

▷ CAM, HAM \in NP;

As classes P e NP

▷ CAM, HAM \in NP; mais ainda: CAM \in P

As classes P e NP

- ▷ CAM, HAM \in NP; mais ainda: CAM \in P
- ▷ Não se sabe se HAM \in P

As classes P e NP

- ▷ CAM, HAM \in NP; mais ainda: CAM \in P
- ▷ Não se sabe se HAM \in P
- ▷ Se HAM \notin P, então P \neq NP

As classes P e NP

- ▷ CAM, HAM \in NP; mais ainda: CAM \in P
- ▷ Não se sabe se HAM \in P
- ▷ Se HAM \notin P, então P \neq NP
- ▷ A hipótese P \neq NP implica que HAM \notin P

As classes P e NP

- ▷ CAM, HAM \in NP; mais ainda: CAM \in P
- ▷ Não se sabe se HAM \in P
- ▷ Se HAM \notin P, então P \neq NP
- ▷ A hipótese P \neq NP implica que HAM \notin P
- ▷ HAM é um problema em NP “maximalmente difícil”

As classes P e NP

- ▷ CAM, HAM \in NP; mais ainda: CAM \in P
- ▷ Não se sabe se HAM \in P
- ▷ Se HAM \notin P, então P \neq NP
- ▷ A hipótese P \neq NP implica que HAM \notin P
- ▷ HAM é um problema em NP “maximalmente difícil”
 - HAM é um problema NP-completo

O problema $P \neq NP$

1. Há inúmeros problemas interessantes em NP .

O problema $P \neq NP$

1. Há inúmeros problemas interessantes em NP . Excelente se $P = NP$

O problema $P \neq NP$

1. Há inúmeros problemas interessantes em NP . Excelente se $P = NP$
2. $P \neq NP$?: É estritamente mais difícil de se **resolver** um problema que **verificar** que uma solução está correta?

O problema $P \neq NP$

1. Há inúmeros problemas interessantes em NP . Excelente se $P = NP$
2. $P \neq NP$?: É estritamente mais difícil de se **resolver** um problema que **verificar** que uma solução está correta?
3. Acredita-se que $P \neq NP$

O problema $P \neq NP$

1. Há inúmeros problemas interessantes em NP . Excelente se $P = NP$
2. $P \neq NP$?: É estritamente mais difícil de se **resolver** um problema que **verificar** que uma solução está correta?
3. Acredita-se que $P \neq NP$
 - ▷ Em particular: $HAM \notin P$

O problema $P \neq NP$

1. Há inúmeros problemas interessantes em NP . Excelente se $P = NP$
2. $P \neq NP$?: É estritamente mais difícil de se **resolver** um problema que **verificar** que uma solução está correta?
3. Acredita-se que $P \neq NP$
 - ▷ Em particular: $HAM \notin P$ ($HAM \in P \Rightarrow P = NP$)

O problema $P \neq NP$

1. Há inúmeros problemas interessantes em NP . Excelente se $P = NP$
2. $P \neq NP$?: É estritamente mais difícil de se **resolver** um problema que **verificar** que uma solução está correta?
3. Acredita-se que $P \neq NP$

▷ Em particular: $HAM \notin P$ ($HAM \in P \Rightarrow P = NP$)

(*Importante*) Frequentemente, quando dizemos que um problema $P \notin P$ (não pode ser resolvido em tempo polinomial), supomos tacitamente que $P \neq NP$

Alguns problemas computacionais envolvendo grafos

Alguns problemas computacionais envolvendo grafos

1. $v-w$ caminho, trilha euleriana fechada, circuito hamiltoniano

Alguns problemas computacionais envolvendo grafos

1. $v-w$ caminho, trilha euleriana fechada, circuito hamiltoniano
2. conexidade,

Alguns problemas computacionais envolvendo grafos

1. $v-w$ caminho, trilha euleriana fechada, circuito hamiltoniano
2. conexidade, 2-colorabilidade (bipartido?),

Alguns problemas computacionais envolvendo grafos

1. $v-w$ caminho, trilha euleriana fechada, circuito hamiltoniano
2. conexidade, 2-colorabilidade (bipartido?), planaridade,

Alguns problemas computacionais envolvendo grafos

1. $v-w$ caminho, trilha euleriana fechada, circuito hamiltoniano
2. conexidade, 2-colorabilidade (bipartido?), planaridade, o problema dos casamentos,

Alguns problemas computacionais envolvendo grafos

1. $v-w$ caminho, trilha euleriana fechada, circuito hamiltoniano
2. conexidade, 2-colorabilidade (bipartido?), planaridade, o problema dos casamentos, emparelhamentos em grafos genéricos

Alguns problemas computacionais envolvendo grafos

1. $v-w$ caminho, trilha euleriana fechada, circuito hamiltoniano
2. conexidade, 2-colorabilidade (bipartido?), planaridade, o problema dos casamentos, emparelhamentos em grafos genéricos
3. 3-colorabilidade

Alguns problemas computacionais envolvendo grafos

1. $v-w$ caminho, trilha euleriana fechada, circuito hamiltoniano
2. conexidade, 2-colorabilidade (bipartido?), planaridade, o problema dos casamentos, emparelhamentos em grafos genéricos
3. 3-colorabilidade
4. isomorfismo de grafos

Alguns problemas computacionais envolvendo grafos

1. $v-w$ caminho (F), trilha euleriana fechada (F), circuito hamiltoniano (I)
2. conexidade (F), 2-colorabilidade (F), planaridade (T), o problema dos casamentos (F), emparelhamentos em grafos genéricos (T)
3. 3-colorabilidade (I)
4. isomorfismo de grafos (O)

Teoria da Complexidade Computacional

▷ $P \neq NP$?

1. [MAC0338](#) Análise de Algoritmos (uma/duas aulas?)
2. [MAC0430](#) Algoritmos e Complexidade de Computação

Buscas em grafos

Buscas em grafos

1. Busca em largura

Buscas em grafos

1. Busca em largura
2. Busca em profundidade

Busca em profundidade/Mz de adj.

Busca em profundidade/Mz de adj.

```
#define maxV 10000
static int cnt, pre[maxV];

#define dfsR search
void dfsR(Graph G, Edge e)
{ int t, w = e.w;
  pre[w] = cnt++;
  for (t = 0; t < G->V; t++)
    if (G->adj[w][t] != 0)
      if (pre[t] == -1)
        dfsR(G, EDGE(w, t));
}
```

A chamada inicial

A chamada inicial

```
void GRAPHsearch(Graph G)
{ int v;
  cnt = 0;
  for (v = 0; v < G->V; v++) pre[v] = -1;
  for (v = 0; v < G->V; v++)
    if (pre[v] == -1)
      search(G, EDGE(v, v));
}
```

Programa 18.3.driver

Programa 18.3.driver

```
#include <stdio.h>
#include <stdlib.h>
#include "GRAPH.h"
main(int argc, char *argv[])
{ int V = atoi(argv[1]), E = atoi(argv[2]);
  Graph G = GRAPHscan_noST(V, E);
  if (V < 20) GRAPHshow(G);
  else {
    printf("%d vertices, %d edges, ", V, E);
    printf("Too big!\n");
  }
  GRAPHsearch(G);
  GRAPHshow_pre(G);
}
```

Exemplo

Exemplo

8 vertices, 10 edges

0: 2 5 7

1: 7

2: 0 6

3: 4 5

4: 3 5 6 7

5: 0 3 4

6: 2 4

7: 0 1 4

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

Busca em profundidade/Ls de adj.

Busca em profundidade/Ls de adj.

```
#define maxV 10000
static int cnt, pre[maxV];

#define dfsR search
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));
}
```

A chamada inicial: a mesma!

A chamada inicial: a mesma!

```
void GRAPHsearch(Graph G)
{ int v;
  cnt = 0;
  for (v = 0; v < G->V; v++) pre[v] = -1;
  for (v = 0; v < G->V; v++)
    if (pre[v] == -1)
      search(G, EDGE(v, v));
}
```

Exemplo

Exemplo

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

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

v e w no mesmo componente?

v e w no mesmo componente?

```
struct graph { int V; int E; link *adj; int *cc;};
```

```
int GRAPHconnect(Graph G, int s, int t)
{ return G->cc[s] == G->cc[t]; }
```

v e w no mesmo componente? (Cont.)

v e w no mesmo componente? (Cont.)

```
void dfsRcc(Graph G, int v, int id)
{ link t;
  G->cc[v] = id;
  for (t = G->adj[v]; t != NULL; t = t->next)
    if (G->cc[t->v] == -1) dfsRcc(G, t->v, id);
}

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

Exemplo

Exemplo

10 vertices, 10 edges

0: 8 3 9 4

1: 9

2: 6 3

3: 0 8 2

4: 0

5: 9 6

6: 2 5

7:

8: 0 3

9: 5 1 0

2 component(s)

Mais exemplos

Mais exemplos

```
yoshi@RANDOM ~/Main/www/2005i/mac328/exx/prog18.4
$ a 100 200
100 vertices, 200 edges, 3 component(s)
```

```
yoshi@RANDOM ~/Main/www/2005i/mac328/exx/prog18.4
$ a 100 223
100 vertices, 223 edges, 2 component(s)
```

```
yoshi@RANDOM ~/Main/www/2005i/mac328/exx/prog18.4
$ a 100 224
100 vertices, 224 edges, 1 component(s)
```

```
yoshi@RANDOM ~/Main/www/2005i/mac328/exx/prog18.4
$
```

Complexidade de tempo da busca em prof.

Complexidade de tempo da busca em prof.

- Matrizes de adjacência: $O(|V(G)|^2)$

Complexidade de tempo da busca em prof.

- Matrizes de adjacência: $O(|V(G)|^2)$
- Listas de adjacência: $O(|V(G)| + |E(G)|)$