Algoritmos

Pedro Hokama

Fontes

- [*clrs*] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest, Ronald L. Rivest e Clifford Stein.
- [*timr*] Algorithms Illuminated Series, Tim Roughgarden

Apresentação Baseada:

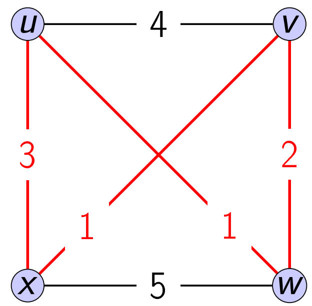
- Stanford Algorithms
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V`
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZl7A5HEK6`
- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende
- Conjunto de Slides do Professores Cid C. de Souza para a disciplina MO420

Qualquer erro é de minha responsabilidade.

Caixeiro Viajante

Problema do Caixeiro Viajante - TSP

Dado um grafo completo não direcionado $G = (V, E)$, custos inteiros $c(i, j)$ para $i, j \in V$ e um inteiro k decidir se existe um ciclo que passa exatamente uma vez por cada vértice (hamiltoniano) com custo menor ou igual a k.



Existe uma solução com custo menor ou igual a 7? Sim.

Teorema

TSP é NP-completo

Lema

$TSP \in NP$

Lema

$TSP \in NP\text{-Difícil}$

Lema

$TSP \in NP$

- Dado uma instância do problema, podemos usar como certificado a sequência de n vértices do percurso.
- O algoritmo de verificação confirma se a sequência contém cada vértice uma vez, e se a soma dos custos das arestas é menor ou igual a k.
- Esse algoritmo pode ser feito em tempo polinomial.

Lema

$TSP \in NP\text{-Difícil}$

- Mostraremos que HAM-CYCLE $\leq_p$ TSP.
- Seja $G = (V, E)$ uma instância do HAM-CYCLE.
- Construímos uma instância do TSP da seguinte maneira: Formamos o grafo completo $G' = (V, E')$, e custos:

$$c(i,j) = \begin{cases} 0 & \text{se } (i,j) \in E, \\ 1 & \text{se } (i,j) \notin E. \end{cases}$$

- Essa instância pode ser criada em tempo polinomial.

- Agora mostramos que G tem um ciclo hamiltoniano, se e somente se, G' tem um percurso cujo custo é zero.
- ($\rightarrow$) Suponha que G tenha um ciclo hamiltoniano h. Cada aresta $h \in E$ e portanto tem custo 0 em G'. Dessa forma h também é um percurso em G' e tem custo 0.
- ($\leftarrow$) Suponha que G' tenha um percurso h' de custo menor ou igual a 0. Como só exitem arestas de custo 0 ou 1, o custo de h' é exatamente 0. Portanto as arestas de h' exitem em G e formam um ciclo hamiltoniano.

Problema da Soma de Subconjuntos

Problema da Soma de Subconjuntos - SUBSET-SUM

Dado um conjunto finito S de inteiros positivos e um inteiro $t > 0$. Decidir se existe um subconjunto $S' \subseteq S$ cuja a soma de seus elementos é t.

- Por exemplo: seja $S = \{1, 2, 7, 14, 49, 98, 343\}$ e $t = 108$.
- O conjunto $S' = \{1, 2, 7, 98\}$ é uma solução.

Teorema

SUBSET-SUM é NP-completo

Lema

$SUBSET\text{-}SUM \in NP$

- Exercício.

Lema

$SUBSET\text{-}SUM \in NP\text{-}Difícil$

Lema

$SUBSET\text{-}SUM \in NP\text{-}Difícil$

- Mostraremos que 3-CNF-SAT $\leq_p$ SUBSET-SUM.
- Considere uma fórmula f para as variáveis $x_1, x_2, \ldots, x_n$ com cláusulas $C_1, C_2, \ldots, C_k$.
- Construiremos uma instância $< S, t >$ para o SUBSET-SUM da seguinte forma:
- Dois números para cada variável x_i (v_i e v_i'),
- e dois números para cada cláusula C_j (s_j e s_j')
- cada número terá $n + k$ dígitos.

- Rotulamos os n primeiros dígitos para cada uma das variáveis,
- e os k últimos dígitos para cada uma das cláusulas.

- O alvo t tem os n primeiros dígitos iguais a 1 e o restante iguais a 4.
- v_i e v_i tem 1 nos dígitos rotulados por x_i.
- v_i tem 1 nos dígitos rotulados pelas cláusulas em que x_i aparece (não negado).
- v_i' tem 1 nos dígitos rotulados pelas cláusulas em que $\neg x_i$ aparece.
- v_i e v_i' tem 0 nos demais dígitos.
- s_j tem 1, e s_j' tem 2 nos dígitos rotulados por C_j e 0 em todos os outros.

$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge$
$(\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge$
$(\neg x_1 \vee \neg x_2 \vee x_3) \wedge$
$(x_1 \vee x_2 \vee x_3)$

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
$v_1 =$	1	0	0	1	0	0	1
$v_1' =$	1	0	0	0	1	1	0
$v_2 =$	0	1	0	0	0	0	1
$v_2' =$	0	1	0	1	1	1	0
$v_3 =$	0	0	1	0	0	1	1
$v_3' =$	0	0	1	1	1	0	0
$s_1 =$	0	0	0	1	0	0	0
$s_1' =$	0	0	0	2	0	0	0
$s_2 =$	0	0	0	0	1	0	0
$s_2' =$	0	0	0	0	2	0	0
$s_3 =$	0	0	0	0	0	1	0
$s_3' =$	0	0	0	0	0	2	0
$s_4 =$	0	0	0	0	0	0	1
$s_4' =$	0	0	0	0	0	0	2
$t =$	1	1	1	4	4	4	4

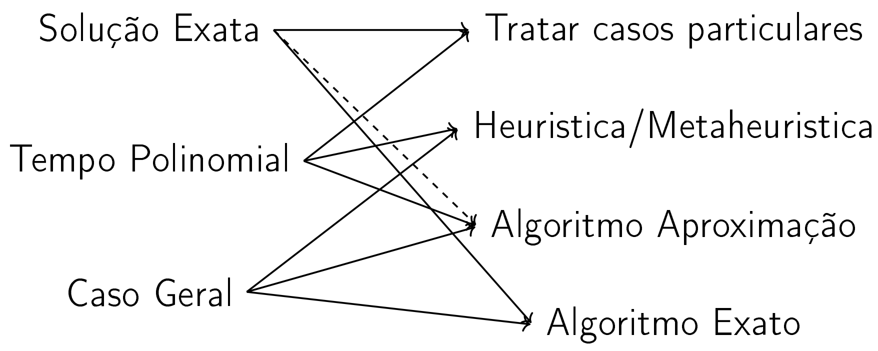
- Falta mostrar que f é satisfazível, se e somente se, a instância $< S, t >$ do SUBSET-SUM tem uma solução.
- $(\rightarrow)$ Suponha que f tem uma atribuição que a torna verdadeira. Para $i = 1, 2, \ldots, n$ se $x_i = 1$ incluímos v_i em S', se $x_i = 0$ incluímos v_i' em S'.
- Como cada cláusula é satisfeita, a soma em cada dígito rotulado como C_j é no mínimo 1 e no máximo 3, dessa forma basta colocar em S' os valores s_j e s_j' que completam 4 naquele dígito.
- $(\leftarrow)$ Exercício.

NP-Difícil

- Um problema pode pertencer a Classe NP-Difícil sem pertencer a classe NP.
- As versões de otimização dos problemas apresentados são exemplos.

O que fazer então?

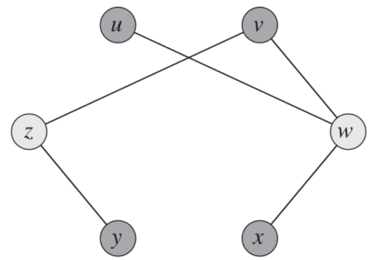
Se $P \neq NP$ não conseguiremos para um problema NP-Difícil:



Cobertura por Vértices

VERTEX-COVER

Dado um grafo não orientado $G = (V, E)$ com n vértice e m arestas, e um inteiro k decidir se existe uma cobertura por vértices $V' \subseteq V$ de tamanho k.



Qual é o tamanho de uma cobertura por vértices de tamanho mínimo em um grafo estrela com n vértices e em um grafo clique de tamanho n.

a. 1 e $n - 1$
b. 1 e n
c. 2 e $n - 1$
d. $n - 1$ e n

- Uma solução força bruta:
- Considerando que k seja pequeno em relação a n podemos tentar todos os conjuntos com k vértices.
- São $\binom{n}{k}$ conjuntos.
- Se $k <<< n$ então a complexidade do algoritmo é $\approx \Theta(n^k)$.
- Podemos fazer melhor?

Teorema

Dada uma aresta (u, v),
$G_u = G$ deletando u e todas as suas arestas adjacentes, e
$G_v = G$ deletando v e todas as suas arestas adjacentes.

G tem uma cobertura de tamanho k, se e somente se G_u ou G_v tem uma cobertura de tamanho $k - 1$.

- ($\rightarrow$) Suponha que G tem uma cobertura V' de tamanho k. Como a aresta (u, v) precisa estar coberta, pelo menos um dos seus extremos tem que estar em V'.
- Sem perda de generalidade, suponha que $u \in V'$.
- O vértice u cobre apenas as arestas adjacentes a u, e portanto todas as demais arestas devem estar cobertas pelos $k - 1$ vértices restantes de V', e portanto
- $V \backslash \{u\}$ é uma cobertura para G_u e tem $k - 1$ vértices
- ($\leftarrow$) Exercício.

Algoritmo 1: BuscaCobertura

Entrada: Um grafo $G(V, E)$ e um inteiro k
Saída: Verdadeiro se G contêm uma cobertura de tamanho k

se $|E| > 0$ *e* $k == 0$ **então devolve** Falso
se $|E| == 0$ **então devolve** Verdadeiro
Escolhe uma aresta qualquer $(u, v) \in E$
Cria $G_u = G$ sem u e suas arestas adjacentes
Cria $G_v = G$ sem v e suas arestas adjacentes
se *BuscaCobertura(G_u, $k - 1$)* **então devolve** Verdadeiro
se *BuscaCobertura(G_v, $k - 1$)* **então devolve** Verdadeiro
devolve Falso

Complexidade de BuscaCobertura:

- A cada chamada recursiva, fazemos outras 2.
- A profundidade da recursão é no máximo k.
- Portanto o número de chamadas recursivas é no máximo 2^n.
- Cada chamada recursiva leva tempo $O(m)$ para criar G_u e G_v.
- Portanto a complexidade total é $O(2^n m)$, portanto exponencial. (Claro que é).
- Ainda muito melhor que o $\Theta(n^k)$ da força bruta.

Algoritmo Exato para o TSP

- Dado um grafo não-direcionado $G = (V, E)$, encontrar o custo mínimo de ciclo que visita exatamente uma vez todos os vértice de V.
- Uma solução força bruta:
- Testar todos os $n!$ percursos possíveis.
- Resolve 12, 13, talvez 14 vértices.
- Podemos fazer melhor? Vamos tentar fazer um algoritmo de Programação Dinâmica!

Tentativa 1

- Vamos entender o ciclo como um caminho que começa no vértice 1 vai até algum vértice j e depois viaja direto j para i, fechando o ciclo.
- Podemos usar a ideia do algoritmo de Bellman-Ford, para encontrar o caminho de 1 até j que usa no máximo i arestas.
- Para todo $i \in \{0, 1, \ldots, n\}$ e todo destino $j \in \{1, 2, ..., n\}$, temos L_{ij} o custo de um caminho mínimo de 1 até j que usa no máximo i arestas.

- Podemos então pegar todos os $L_{n-1,j}$ e somar com o custo de $c_j 1$.
- Isso vai formar um ciclo de caixeiro viajante? Infelizmente não!
- O Bellman-Ford apenas indica o máximo de arestas que podem ser usadas.

Tentativa 2

- Podemos modificar um pouco a ideia do algoritmo de Bellman-Ford, para encontrar o caminho de 1 até j que usa no **EXATAMENTE** i arestas.
- Para todo $i \in \{0, 1, \ldots, n\}$ e todo destino $j \in \{1, 2, ..., n\}$, temos L_{ij} o custo de um caminho mínimo de 1 até j que usa **exatamente** i arestas.

- Novamente, podemos então pegar todos os $L_{n-1,j}$ e somar com o custo de $c_j 1$. Certo?
- Infelizmente também não. Note que o caminho mínimo de $n-2$ arestas até um vértice k pode passar por j,
- Dessa forma estaríamos repetindo vértices (e deixando alguns de fora).

Tentativa 3

- Podemos tentar modificar mais um pouco a ideia para encontrar o caminho de 1 até j que usa no **exatamente** i arestas e **NÃO REPETE** vértices.
- Para todo $i \in \{0, 1, \ldots, n\}$ e todo destino $j \in \{1, 2, ..., n\}$, temos L_{ij} o custo de um caminho mínimo de 1 até j que usa **exatamente** i arestas e **não repete** vértices..

- Novamente, podemos então pegar todos os $L_{n-1,j}$ e somar com o custo de $c_j 1$. Certo?
- Infelizmente também não. Note que ainda o caminho mínimo de $n-2$ arestas até um vértice k pode passar por j.
- Como garantir que não vai haver repetições de vértices?
- O único jeito é resolver mais subproblemas, para cada possível conjunto de vértices.

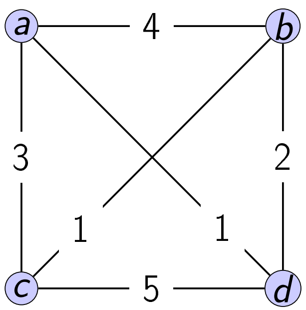
Subestrutura Ótima

- Para todo destino $j \in \{1, 2, ..., n\}$ e todo subconjunto $S \subseteq V$ que contêm 1 e j.
- L_{Sj} é o custo de um caminho mínimo de 1 até j visita todos os vértices de S (e somente eles).

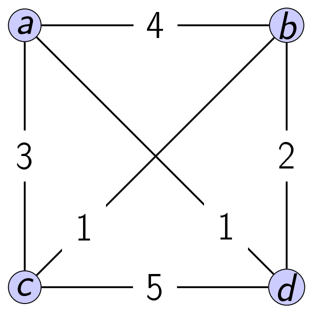
- Considere o seguinte exemplo:
- Queremos descobrir $L_{\{1,3,7,8,9\}}, 7$ ou seja queremos o caminho de 1 até 7 que visita exatamente uma vez os vértices 1, 3, 7, 8 e 9. Quais opções temos?
- Podemos ir de 1 até 3 visitando $\{1, 3, 8, 9\}$ depois para 7.
- Podemos ir de 1 até 8 visitando $\{1, 3, 8, 9\}$ depois para 7.
- Podemos ir de 1 até 9 visitando $\{1, 3, 8, 9\}$ depois para 7.

Dessa forma podemos escrever a seguinte recorrência:

$$L_{S,j} = \begin{cases} c_{1j}, \text{ se } S = \{1, j\} \\ \min_{k \in S, k \neq j} \left\{ L_{S \setminus \{j\}, k} + c_{kj} \right\} \end{cases}$$

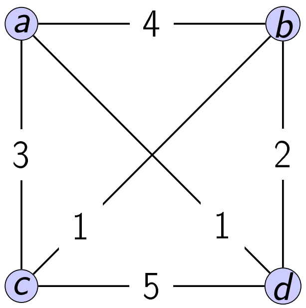


S	b	c	d
$\{a, b\}$	4	-	-
$\{a, c\}$	-	3	-
$\{a, d\}$	-	-	1



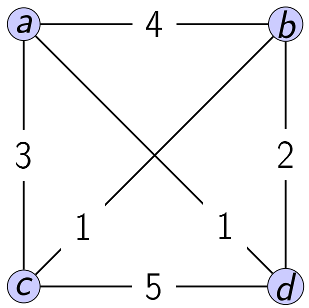
S	b	c	d
$\{a, b\}$	4	-	-
$\{a, c\}$	-	3	-
$\{a, d\}$	-	-	1

S	b	c	d
$\{a, b, c\}$	4	4	-
$\{a, b, d\}$	3	-	6
$\{a, c, d\}$	-	6	8



S	b	c	d
$\{a, b, c\}$	4	4	-
$\{a, b, d\}$	3	-	6
$\{a, c, d\}$	-	6	8

S	b	c	d
$\{a, b, c, d\}$	7	4	6



S	b	c	d
$\{a, b, c, d\}$	7	4	6

Por fim a ultima aresta,

$(b, a) = 7 + 4 = 11$

$(c, a) = 4 + 3 = 7$

$(d, a) = 6 + 1 = 7$

Algoritmo 2: BellmanHeldKarp

Entrada: $G(V, E)$, $V = \{1, 2, ..., n\}$, custos c_{ij} para $(i, j) \in E$

Saída: Custo mínimo de um percurso de caixeiro viajante em G

//$A[S][j]$ indica o custo mínimo de chegar em j passando uma vez por cada vértice de $S \subseteq V$

$A[2^{n-1} - 1][n - 1]$;

para $j = 2$ *até* n **faça**

    $A[\{1, j\}][j] = c_{1j}$;

para $s = 3$ *até* n **faça**

    para *todo* S *com* $|S| = s$ *e* $1 \in S$ **faça**

        para $j \in S \backslash \{1\}$ **faça**

            $A[S][j] = \min_{k \in S \backslash \{1, j\}}(A[S \backslash \{j\}][k] + c_{kj})$;

devolve $\min_{j \in S \backslash \{1\}}(A[V][j] + c_{j1})$;

Complexidade.

- Temos $O(2^n)$ possíveis escolhas de S.
- Para cada S temos $O(n)$ problemas (um para cada membro de S)
- O total de subproblemas é $O(n2^n)$
- Para cada subproblema temos que procurar o mínimo em $O(n)$ subproblemas.
- Dessa forma a complexidade total de BellmanHeldKarp é $O(n^2 2^n)$

Considere que vamos utilizar um computador de 4Ghz

n	$n!$	$n^2 2^n$
10	0 s	0s
15	300 s	0s
18	18 dias	0s
20	19 anos	0,1 s
23	200 milênios	1 s
35	?	3 horas
40	?	5 dias