Projeto e Análise de Algoritmos

Redução entre problemas

Cid Carvalho de Souza, Cândida Nunes da Silva et al.

Primeiro Semestre de 2017

Conceitos de redução entre problemas

Contas superior e inferior de um problema

Seja P um problema e suponha que n é um parâmetro que denota o tamanho de uma instância de P.

- Uma cota superior para P é uma função $g(n)$ tal que existe algum algoritmo que resolve P com complexidade $O(g(n))$.
- Uma cota inferior para P é uma função $f(n)$ tal que todo algoritmo que resolve P tem complexidade $\Omega(f(n))$.
- Um algoritmo é ótimo para um problema P se sua complexidade coincidir com uma cota inferior de P.

  Por exemplo, o problema da ordenação tem cota inferior $\Omega(n \lg n)$ e existe algoritmo de ordenação de complexidade $O(n \lg n)$ (heapsort e mergesort).

Reduções

Esquema básico de uma redução de Turing:

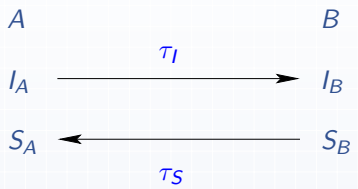
Problema A:	Problema B:
▸ Instância: I_A	▸ Instância: I_B
▸ Solução: S_A	▸ Solução: S_B

Definição. Uma redução do problema A ao problema B é um par de transformações τ_I, τ_S tal que para toda instância I_A de A:

- τ_I transforma I_A em uma instância I_B de B, **e**
- τ_S transforma uma solução S_B de I_B em uma solução S_A de I_A.

Reduções

Esquema básico de uma redução de Turing:



Quando usar reduções?

- **Situação 1:** quero encontrar um algoritmo para resolver o problema A e conheço um **algoritmo** que resolve B, ou seja, determinar uma **cota superior** para o problema A;
- **Situação 2:** quero determinar uma **cota inferior** para o problema B e conheço uma **cota inferior** para o problema A.

Exemplo

- Quero resolver um sistema linear $Mx = b$ onde M é uma matriz de posto completo. (Este é o problema A.)
- Disponho de um programa que resolve sistemas lineares $Px = d$ em que $P = (p_{ij})$ é uma matriz quadrada simétrica (isto é, $p_{ij} = p_{ji}$). (Este é o problema B.)

Podemos fazer uma redução de A para B.

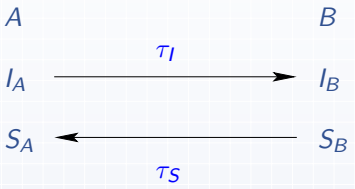
- Dado um sistema $Mx = b$, construo a matriz $P = M^T M$ e o vetor $d = M^T b$.
- Álgebra Linear: Um vetor x é solução de $Mx = b$ se e somente se é solução de $M^T Mx = M^T b$, ou seja, $Px = d$.

Exemplo

- Um vetor x é solução de $Mx = b$ se e somente se é solução de $M^T Mx = M^T b$, ou seja, $Px = d$.
- A redução mostra como resolver o problema A, compondo a redução com o algoritmo que resolve o problema B.
- **Conclusão:** resolver sistemas lineares da forma $Px = b$ quando P é **simétrica** é pelo menos tão difícil quanto resolver um sistema linear $Mx = b$ em que M é uma matriz qualquer de posto completo.

Reduções

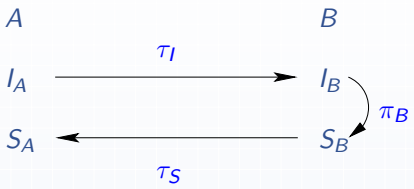
Definição: Um problema A é redutível a um problema B em tempo $f(n)$ se existe uma redução como esquematizada abaixo:



onde $n = |I_A|$ e, τ_I e τ_S custam $O(f(n))$.

Notação: $A \prec_{f(n)} B$.

Observações

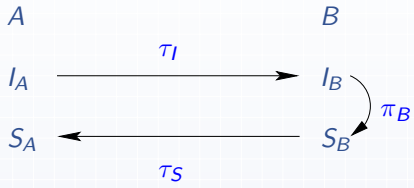


- Conhecendo um algoritmo π_B que resolve B, temos imediatamente um algoritmo π_A que resolve qualquer instância de A:

$$\pi_A = \tau_I \circ \pi_B \circ \tau_S.$$

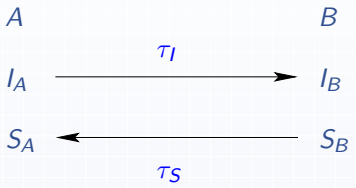
- a complexidade de π_A é a soma das complexidades de τ_I, π_B e τ_S e deve ser expressa em função do tamanho de $n = |I_A|$.
  Isto resulta em uma cota superior para A.

Observações



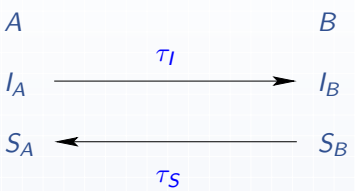
- Se π_B tem complexidade $g(n)$ (cota superior de B) e $g(n) \in \Omega(f(n))$ então $g(n)$ também é uma cota superior de A.
  - Se $g(n) \notin \Omega(f(n))$, a cota superior ainda vale?

Observações



- Se $\Omega(h(n))$ é uma cota inferior para o problema A e $f(n) \in o(h(n))$, então $\Omega(h(n))$ também é cota inferior para o problema B.
  - Por que temos a restrição de que $f(n) \in o(h(n))$?
  - Lembre-se que $o(h(n))$ e $\Omega(h(n))$ são disjuntos.

Observações



- Em uma redução **não** é necessário explicar **como resolver** o problema B, apenas como τ_I e τ_S funcionam
- a complexidade da **redução** é a soma das complexidades de τ_I e τ_S (ou equivalentemente, a maior das duas).

Exemplos de reduções

Problema do casamento cíclico de strings (CSM)

Entrada: alfabeto Σ e strings sobre Σ de tamanho n:

$$A = a_0 a_1 \ldots a_{n-1} \text{ e } B = b_0 b_1 \ldots b_{n-1}.$$

Objetivo: decidir se B é um deslocamento cíclico de A.

Ou seja, existe $k \in \{0, 1, \ldots, n-1\}$ tal que $a_{(i+k) \mod n} = b_i$ para todo $i = 0, 1, \ldots, n-1$?

Exemplo: para $A = acgtact$ e $B = gtactac$ ($n = 7$) temos $k = 2$.

Como se resolve o CSM?

Exemplos de reduções

Problema do casamento de strings (SM)

Entrada: alfabeto Σ e strings sobre Σ:

$$A = a_0 a_1 \ldots a_{n-1} \text{ e } B = b_0 b_1 \ldots b_{m-1}, \text{com } m \leq n.$$

Objetivo: encontrar a primeira ocorrência de B em A ou concluir que B não é subcadeia de A.

Ou seja, determinar o menor $k \in \{0, 1, \ldots, n-1\}$ tal que $a_{(i+k) \mod n} = b_i$ para todo $i = 0, 1, \ldots, m-1$ ou devolver $k = -1$.

Exemplo: para $A = acgttaccgtacccg$ e $B = tac$ ($n = 15$ e $m = 3$) temos $k = 4$.

Observação: o problema SM pode ser resolvido em tempo $O(n + m)$ pelo algoritmo KMP de Knuth, Morris and Pratt (1977).

CSM $\prec$ SM

Redução: CSM $\prec_n$ SM

- Instância de CSM: $I_{CSM} = (A, B, n)$.
- τ_I constrói a instância de SM:

$$I_{SM} = (A', 2n, B, n), \text{ onde } A' = A||A.$$

  Portanto, τ_I custa $O(n)$.
- Se k é a solução de SM para I_{SM}, então k também é a solução de I_{CSM}. Logo, τ_S custa $O(1)$ e a redução custa $O(n)$.

Exemplo:

- $I_{CSM} = (acgtact, gtactac, 7)$
- $I_{SM} = (acgtactacgtact, 14, gtactac, 7)$
- $S_{SM} = S_{CSM} = \{k = 2\}$

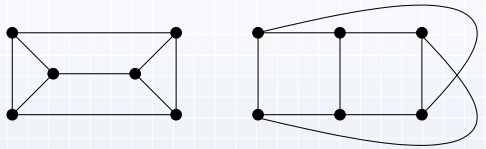
Exemplos de reduções

Problema da existência de triângulo (PET)

Entrada: grafo conexo $G = (V, E)$ sem laços com $n = |V|$ e $m = |E|$.

Objetivo: decidir se G contém um triângulo.

Exemplo:



SIM NÃO

Observações sobre o PET

- Há um algoritmo trivial de complexidade $O(n^3)$: verificar todas as triplas de vértices.
- Existe um algoritmo $O(mn)$ que é muito bom para grafos esparsos.
- Supomos que G é dada pela sua matriz de adjacência $A = A(G)$.
- Se $A^2 = A \times A$, então $a_{ij}^2 = \sum_{k=1}^{n} a_{ik} a_{kj}$. Então

$$a_{ij}^2 > 0 \Leftrightarrow \exists k \in \{1, \ldots, n\} \text{ tal que } a_{ik} = a_{kj} = 1.$$

- Portanto, (i, j, k) corresponde a um triângulo se, e somente se, $a_{ij}^2 > 0$ e $a_{ij} = 1$.
- Note que $a_{ii} = 0$ para $i = 1, \ldots n$.

Exemplos de reduções

Problema da Multiplicação de Matrizes Quadradas (MMQ)

Entrada: matrizes quadradas (de inteiros) A e B de ordem n.

Objetivo: calcular o produto $P = A \times B$.

Observações:

- há um algoritmo óbvio de complexidade $O(n^3)$;
- MMQ pode ser resolvido em tempo $O(n^{\log 7 = 2.807})$ pelo algoritmo de Strassen (1969) ou em tempo $O(n^{2.376})$ pelo algoritmo de Coppersmith e Winograd (1990).

PET $\prec$ MMQ

Redução: PET $\prec_{n^2}$ MMQ

- Instância de PET: $I_{PET} = A(G)$.
- τ_I constrói a instância de MMQ:

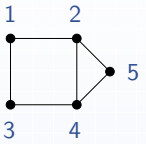
$$I_{MMQ} = (A, A, n), \text{ onde } A = A(G).$$

  Portanto, τ_I custa $O(n^2)$.
- Se $S_{MMQ} = P$ é a solução de MMQ para I_{MMQ}, então a solução de I_{PET} é obtida pelo algoritmo abaixo:

  para $i = 1$ **até** n **faça**
  para $j = 1$ **até** n **faça**
  se $p_{ij} > 0$ **e** $a_{ij} = 1$ **então devolva** SIM
  devolva NÃO

  Logo, τ_S custa $O(n^2)$.

PET $\prec$ MMQ



$A(G)$

	1	2	3	4	5
1	0	1	1	0	0
2	1	0	0	1	1
3	1	0	0	1	0
4	0	1	1	0	1
5	0	1	0	1	0

$P = A(G) \times A(G)$

	1	2	3	4	5
1	2	0	0	2	1
2	0	3	2	1	1
3	0	2	2	0	1
4	2	1	0	3	1
5	1	1	1	1	2

Exemplos de reduções

Multiplicação de Matrizes Simétricas (MMS)

Entrada: matrizes simétricas (de inteiros) A e B de ordem n.

Objetivo: calcular o produto $P = A \times B$.

Observações:

- MMS é um caso particular de MMQ: a redução MMS $\prec_{n^2}$ MMQ é imediata;
  Portanto, MMQ é pelo menos tão difícil quanto MMS.
- Será que MMS é pelo menos tão difícil quanto MMQ (menos óbvio).

MMQ $\prec$ MMS

Redução: MMQ $\prec_{n^2}$ MMS

- Instância de MMQ: $I_{MMQ} = (A, B, n)$.
- τ_I constrói a instância de MMS: $I_{MMS} = (A', B', 2n)$ onde

$$A' = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \quad \text{e} \quad B' = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}$$

  Portanto, τ_I custa $O(n^2)$.
- A solução de MMS é:

$$P' = A'B' = \begin{bmatrix} AB & 0 \\ 0 & A^T B^T \end{bmatrix}$$

MMQ $\prec$ MMS

- A função τ_S pode ser implementada pelo algoritmo abaixo:

  para $i = 1$ **até** n **faça** ▷ copia AB para P
  para $j = 1$ **até** n **faça**
  $p_{ij} \leftarrow p'_{ij}$

  Logo, τ_S custa $O(n^2)$.
- Por esta redução, se MMQ tem cota inferior em $\Omega(h(n))$, então MMS também tem cota inferior em $\Omega(h(n))$.

  Note que $h(n) \in \Omega(n^2)$. (Por quê?)

MMQ $\prec$ MMS

- **Observacao:** se $T(n)$ é a complexidade de um algoritmo para MMS e $T(2n) \in O(T(n))$[1], então pela redução acima, temos um algoritmo de complexidade $O(T(n) + n^2)$ para resolver MMQ.
- Por quê? Temos que a ordem das matrizes de I_{MMS} é $2n$. Assim, o tempo para resolver I_{MMS} é $O(T(2n)) = O(T(n))$. Somando isto ao custo da redução ($O(n^2)$), obtemos $O(T(n) + n^2)$.

[1] propriedade atendida por funções *suaves* (por exemplo, polinômios).

Erros comuns ao usar reduções

- Usar a redução na ordem inversa: por exemplo, fazer a redução $A \prec B$ e concluir que A é pelo menos tão difícil quanto B.
- Dada a redução $A \prec B$, achar que toda instância de B tem que ser mapeada em alguma instância de A. O mapeamento τ_I é injetor (não necessariamente bijetor).
- Usar o algoritmo produzido por uma redução sem se preocupar com a existência de outro mais eficiente.

Reduções polinomiais

- Nesta disciplina estamos interessados em algoritmos polinomiais para resolver problemas.
- Escrevemos A $\prec_{\text{poli}}$ B se existe uma redução de custo polinomial de A para B e dizemos que A é polinomialmente redutível a B.
- Então se B pode ser resolvido por um algoritmo polinomial, A também pode.
- Esta noção torna-se mais importante no estudo da Teoria da Complexidade quando estudamos a aparente inexistência de algoritmos polinomiais para uma grande classe de problemas: problemas NP-difíceis/NP-completos.

Reduções para obtenção de cota inferior

Reduções para obter cotas inferiores

- Veremos algumas reduções que nos permitem obter cotas inferiores para vários problemas.
- Sejam A e B dois problemas. Suponha que A tem cota inferior $\Omega(h(n))$.
- Se $A \prec_{f(n)} B$ e $f(n) \in o(h(n))$, então B também tem cota inferior $\Omega(h(n))$.
- **Atenção!** Resultados sobre cota inferior **dependem** do **modelo de computação** adotado.

  Por exemplo, o Problema da Ordenação tem cota inferior no **modelo de árvores binárias de decisão**.

Reduções para obter cotas inferiores

Problema da Ordenação (ORD)

Entrada: sequência de elementos comparáveis de comprimento n

$$X = (x_1, x_2, \ldots, x_n).$$

Objetivo: encontrar uma **permutação ordenada** de X.

Observação: no **modelo de árvores binárias de decisão**, o problema tem cota inferior $\Omega(n \lg n)$. Informalmente, qualquer algoritmo de ordenação baseado em comparações tem complexidade $\Omega(n \lg n)$.

Reduções para obter cotas inferiores

Problema da Unicidade de Elementos (UE)

Entrada: sequência de elementos comparáveis de comprimento n

$$X = (x_1, x_2, \ldots, x_n).$$

Objetivo: decidir se os elementos são **todos distintos**.

Observações:

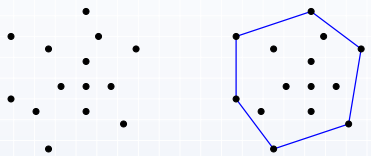
- no **modelo de árvores binárias de decisão**, o problema tem cota inferior $\Omega(n \lg n)$.
- a prova deste fato é semelhante à prova da cota inferior do Problema da Ordenação (omitimos aqui).
- o problema pode ser resolvido em tempo $O(n \lg n)$. (Como?)

Envoltória convexa

Problema da Envoltória Convexa (EC)

Entrada: conjunto $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ de n pontos no plano.

Objetivo: encontrar o **menor polígono convexo** que contém os n pontos.



Observações:

- a saída é a ordem cíclica anti-horária dos vértices do polígono;
- problema clássico de Geometria Computacional: pode ser resolvido em tempo $O(n \lg n)$.

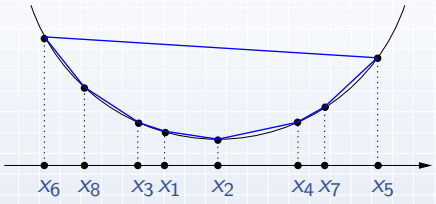
ORD $\prec_n$ EC

Redução: ORD $\prec_n$ EC

- Instância de ORD: $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
- τ_I constrói a instância de EC:

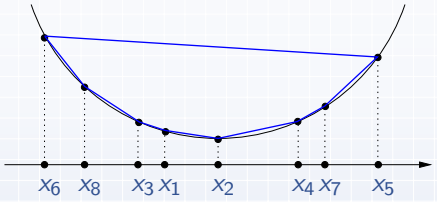
$$I_{EC} = \{(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)\}.$$

Logo, τ_I custa $O(n)$.



ORD $\prec_n$ EC

- A solução de I_{EC} é uma ordem cíclica de pontos.
- τ_S determina o ponto que tem **menor abcissa** e lista os próximos pontos seguindo a ordem cíclica.
  Claramente, τ_S custa $O(n)$.



- Segue que $\Omega(n \lg n)$ é uma cota inferior para EC.

Par Mais Próximo Em Duas Dimensões

Problema do Par Mais Próximo (PMP)

Entrada: coleção $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ de n pontos no plano.

Objetivo: encontrar um **par de pontos** que estejam **a menor distância**.

Observação:

- problema clássico em Geometria Computacional: pode ser resolvido em tempo $O(n \lg n)$.

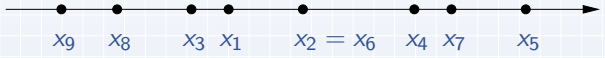
UE $\prec_n$ PMP

Redução: UE $\prec_n$ PMP

- Instância de UE: $I_{UE} = (x_1, x_2, \ldots, x_n)$.
- τ_I constrói a instância de PMP:

$$I_{PMP} = \{(x_1, 0), (x_2, 0), \ldots, (x_n, 0)\}.$$

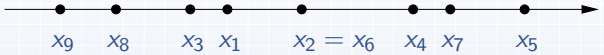
Claramente, τ_I custa $O(n)$.



UE $\prec_n$ PMP

- A solução de I_{PMP} é um par de pontos $(x_i, 0), (x_j, 0)$.
- τ_S verifica se a **distância** entre os dois pontos é **zero**.
  Se SIM então a resposta de I_{UE} é NÃO.
  Caso contrário, a resposta de I_{UE} é SIM.

  Claramente, τ_S custa $O(1)$.



- Logo, $\Omega(n \lg n)$ é uma cota inferior para PMP.

3-SOMA

Problema da 3-Soma (3SUM)

Entrada: sequência $X = (x_1, x_2, \ldots, x_n)$ de reais.

Objetivo: determinar se existem índices distintos i, j e k tais que:

$$x_i + x_j + x_k = 0.$$

Exemplo: $X = (\underline{4}, -6, \underline{1}, 8, 7, \underline{-5})$; solução $i = 1$, $j = 3$ e $k = 6$.

Observações:

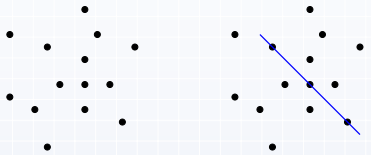
- problema pode ser resolvido em tempo $O(n^2)$; (Como?)
- acreditava-se que $\Omega(n^2)$ é uma cota inferior para 3SUM!
  - Grønlund, Pettie motraram que há algoritmo $o(n^2)$!!! (2014)
  - mas ainda se acredita que não dá pra fazer melhor que $n^{2-\Omega(1)}$

Colinearidade

Problema da Colinearidade (COL)

Entrada: conjunto $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ de n pontos no plano.

Objetivo: determinar se **três dos pontos** dados pertencem a uma mesma **reta não horizontal**.



Observação:

- problema pode ser resolvido em tempo $O(n^2)$;
- acredita-se que $\Omega(n^2)$ é uma cota inferior para COL, mas ninguém sabe provar isto!

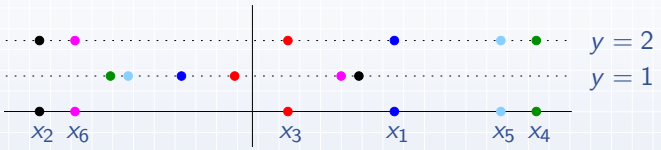
3SUM $\prec_n$ COL

Redução: 3SUM $\prec_n$ COL

- Instância de 3SUM: $I_{3SUM} = (x_1, x_2, \ldots, x_n)$.
- τ_I constrói a instância de COL:

$$I_{COL} = \{(x_i, 0), (-x_i/2, 1), (x_i, 2) : i = 1, 2, \ldots, n\}.$$

Claramente, τ_I custa $O(n)$.



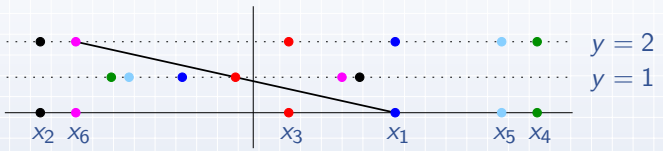
Exemplo: $X = (4, -6, 1, 8, 7, -5)$

3SUM $\prec_n$ COL

- A solução de I_{COL} (se houver) é uma tripla de pontos colineares. Claramente, cada um desses pontos deve estar em um dos eixos horizontais. Ou seja, tem a forma:

$$(x_i, 0), (-x_j/2, 1), (x_k, 2).$$

  Portanto, $x_i + x_j + x_k = 0$.
- De modo análogo, se $x_i + x_j + x_k = 0$, então os pontos citados são colineares.

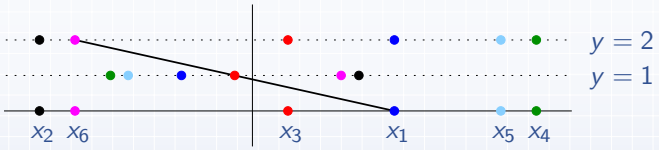


Exemplo: $X = (4, -6, 1, 8, 7, -5)$

3SUM $\prec_n$ COL

- Logo, τ_S custa $O(1)$,
- Como a redução é **linear**, **se** $\Omega(h(n))$ é uma cota inferior de 3SUM, **então** $\Omega(h(n))$ também é uma cota inferior de COL.

  Entretanto, a única cota inferior conhecida é o trivial $\Omega(n)$.



Exemplo: $X = (4, -6, 1, 8, 7, -5)$

Exercício

O Problema 3SUMplus consiste em dados uma sequência $X = (x_1, x_2, \ldots, x_n)$ de reais e um real b, determinar se existem três índices distintos i, j e k tais que $x_i + x_j + x_k = b$.

- Mostre que 3SUM $\prec_n$ 3SUMplus.
- Mostre que 3SUMplus $\prec_n$ 3SUM.
- Suponha que o Professor Sabit Udo descobriu (corretamente) uma cota inferior de $\Omega(n^{1.9})$ para 3SUMplus.

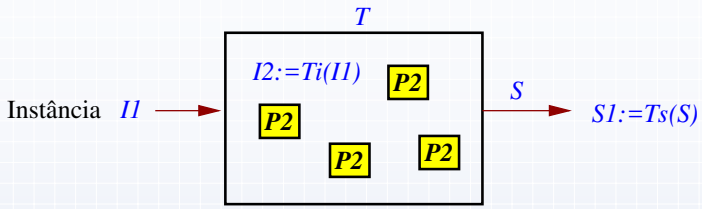
  Quais das afirmações abaixo podemos concluir que são verdadeiras?

  (I) Não existe algoritmo $O(n^{1.5})$ para 3SUMplus.
  (II) Não existe algoritmo $O(n^{1.5})$ para 3SUM.
  (III) Existe um algoritmo $O(n^{1.9})$ para 3SUMplus.
  (IV) Existe um algoritmo $O(n^{1.9})$ para 3SUM.

Exemplos de reduções

Redução de Turing

Existem reduções de P_1 para P_2, fazendo várias aplicações de P_2.



Exemplo de Redução de Turing

Problema de Multiplicação de Inteiros

Dados inteiros a e b, calcular $a \cdot b$.

Problema do Quadrado

Dados inteiro x, calcular x^2.

Proposição: Se pudermos fazer número constante de somas, subtrações e divisão por 2 então podemos reduzir Multiplicação de Inteiros para Quadrado.
Prova: Note que

$$a \cdot b = \frac{(a+b)^2 - a^2 - b^2}{2}$$

Sistema de Representantes Distintos

Dada coleção de conjuntos $S_1, \ldots, S_k$ temos que $R = \{r_1, \ldots, r_k\}$ é um Sistema de Representantes Distintos (SRD) se $r_i \in S_i$ para todo $i = 1, \ldots, k$.

Problema do Sistema de Representantes Distintos

Dada coleção de conjuntos $S_1, \ldots, S_k$, encontrar um SRD.

Sistema de Representantes Distintos

Considere os seguintes conjuntos:

Ecológicos: Ana, Alberto

Ruralistas: João, Alberto

Feministas: Ana, Maria

Então, {Ana, João, Maria} formam um SRD.

Não há SRD para a coleção abaixo:

- $S_1 = \{1, 2\}$
- $S_2 = \{3, 4\}$
- $S_3 = \{3, 4\}$
- $S_4 = \{1, 2, 4\}$
- $S_5 = \{2, 4\}$

Sistema de Representantes Distintos

Teorema de Hall

$S_1, \ldots, S_k$ tem um SRD se e somente se

$$|\{S_{i_1} \cup S_{i_2} \cup \ldots \cup S_{i_m}\}| \geq m$$

para $\{i_1, \ldots, i_m\} \subseteq \{1, 2, 3, \ldots, k\}$ e $1 \leq m \leq k$.

Isso é, qualquer subcoleção de m conjuntos tem pelo menos m itens distintos.

- Podemos usar o Teorema de Hall, de maneira direta, para testar todas possíveis subcoleções de conjuntos.
- Mas o número de possíveis subcoleções é $O(2^k)$.

Sistema de Representantes Distintos

Problema do Emparelhamento Máximo - EM

Dado grafo bipartido $G = (X, Y, E)$, onde X e Y são conjuntos de vértices e E é o conjunto de arestas, encontrar um emparelhamento (conjunto de arestas sem extremos em comum) de cardinalidade máxima.

Proposição: Problema do SRD $\prec$ Problema do EM.
Dada coleção $S_1, \ldots, S_k$, instância de um SRD, sobre conjunto A, defina grafo $G = (X, Y, E)$ onde

- $X = S_1 \cup S_2 \cup \ldots \cup S_k$
- $Y = \{1, 2, \ldots, k\}$
- $E = \{(a, j) |\ a \in S_j \ \text{ e } \ 1 \leq j \leq k\}$.

Note que o SRD tem solução sse G tem emparelhamento de tamanho k. ■

Edição de String

As seguintes operações são possíveis em strings:

- Inserção de um caracter
- Remoção de um caracter
- Troca de um caracter por outro

Problema de Edição de String - ES

Dadas strings A e B, transformar A em B com o menor número de operações.

Edição de String

Se $A = babb$ e $B = bbc$, transformamos A em B com duas operações:

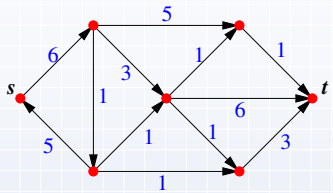
$babb$
$\Downarrow$ Remover a
bbb
$\Downarrow$ Trocar último b por c
bbc

Exercicio: O Problema de Edição de String pode ser resolvido por programação dinâmica.

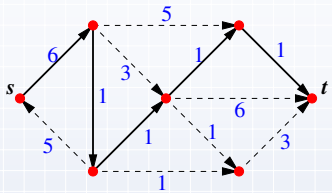
Edição de String

Problema do Caminho Mínimo em Grafo Orientado

Dado grafo orientado $G(V, E)$, onde cada aresta ij possui custo $c_{ij} > 0$, e vértices s e t, , encontrar um caminho de custo total mínimo de s a t em G.



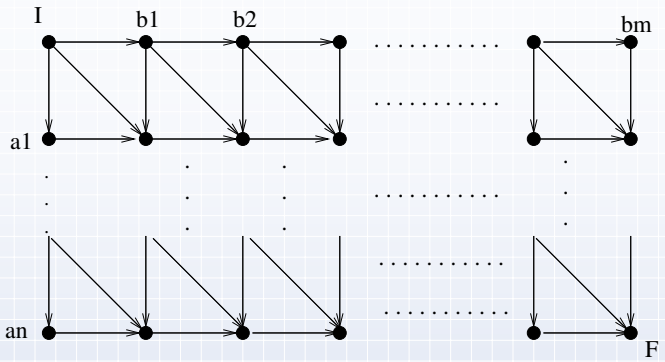
G(V,E,w)



Um caminho minimo de s para t

Edição de String

Proposição: Problema de Edição de String $\prec$ Problema do Caminho Mínimo em grafos orientados.
Prova: Dadas strings $A = a_1 a_2 \ldots a_n$ e $B = b_1 b_2 \ldots b_m$ (instância do Problema ES) construímos um grafo da seguinte forma:

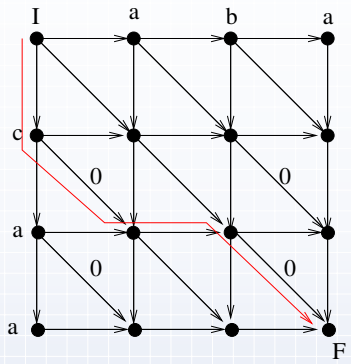


Edição de String

- Arestas horizontais correspondem a inserção de um caractere e possuem custo 1.
- Arestas verticais correspondem a remoção de um caractere e possuem custo 1.
- Arestas diagonais correspondem a uma troca e tem custo 1 caso os caracteres sejam diferentes, e 0 caso sejam iguais.
- O problema é encontrar um caminho mínimo do vértice I ao F.

Edição de String

Dados strings $A = caa$ e $B = aba$ construa grafo G:



custo de edição é $2 = 1 + 0 + 1 + 0$

Edição de String

Temos que mostrar que um caminho mínimo em G de I até F corresponde a uma edição mínima.

- Dado uma edição mínima, a partir do caracter vazio temos 4 opções (inserir, remover, trocar/match) que correspondem as arestas no grafo.
- Uma edição de strings corresponde a um caminho e este por sua vez corresponde a uma edição.
- Portanto um caminho mínimo será uma edição mínima.

Código de Huffman: Compressão de dados

Relembrando:

- Codificação para compressão de dados
- Cada caractere deve ter representação binária única
- Há codificações com número de bits fixo e variável
- Codificação Prefixa: Codificação de um caracter não é prefixo de outro
- Exemplo de codificação prefixa:

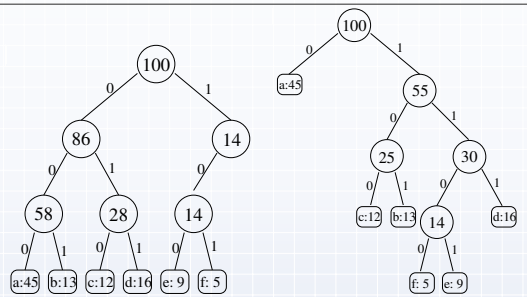
a	b	c	d	e	f
0	101	100	111	1101	1100

- Codificação prefixa pode ser representada por uma árvore de Huffman
- Caracteres são representados nas folhas

Exemplo

Considere 100.000 caracteres com frequências e codificações:

	a	b	c	d	e	f
Frequência da letra	45	13	12	16	9	5
Codificação usando 3 bits	000	001	010	011	100	101
Codificação de tamanho variável	0	101	100	111	1101	1100



Codificação Fixa: $100.000 \times 3 = 300.000$ bits

Codificação Variável: $\sum_{s \in \mathcal{A}} freq(s) \cdot 100.000 \cdot |codigo(s)| = 224.000$ bits

Código de Huffman

Dado texto com caracteres $C_1, C_2, \ldots, C_n$,

- cada caractere C_i tem frequência f_i,
- obter código S_i para C_i, onde $s_i = |S_i|$ e
- minimizamos $c = \sum_{i=1}^{n} s_i f_i$
- Os códigos devem satisfazer a restrição de prefixo.
- Construa árvore de Huffman correspondente a codificação

Teorema: Existe um algoritmo que calcula a árvore ótima para um conjunto de caracteres $C_1, C_2, \ldots, C_n$ com frequências $f_1, f_2, \ldots, f_n$ em tempo $O(n \log n)$.

Código de Huffman

Faremos uma redução do problema de ordenação de inteiros para o código de Huffman.
Proposicao: Sejam $C_1, C_2, \ldots, C_n$ caracteres com frequências $f_1, f_2, \ldots, f_n$ onde

$$\sum_{i=1}^{j-1} f_i < f_j \text{ para cada } j = 2, \ldots, n.$$

Então a árvore de Huffman ótima terá C_n como folha no nível 1, C_{n-1} como folha no nível 2, e assim sucessivamente com C_1 e C_2 no nível $n-1$.

Código de Huffman

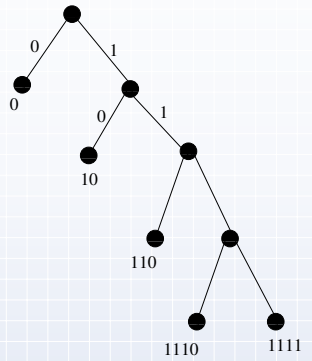
Prova: Seja T^* uma árvore ótima para uma instância como a enunciada. Suponha por absurdo que em T^* o caractere C_n não esteja no nível 1. Adicione uma nova raiz em T^* retirando C_n do seu lugar e colocando este como filho da nova raiz. T^* também é adicionada a esta nova raiz.
O custo da nova árvore diminuiu de pelo menos f_n e aumentou de

$$\sum_{i=1}^{n-1} f_i.$$

Portanto T^* não era ótima o que é um absurdo. Podemos repetir o mesmo raciocínio para os demais caracteres $C_{n-1}, C_{n-2}, \ldots$. ■

Código de Huffman

- Instâncias baseadas no último teorema terão uma árvore de Huffman com a seguinte forma:



Código de Huffman

Tentativa:

- Considere uma instância para o problema de ordenação com números $x_1, x_2, \ldots, x_n$ com uma diferença grande de valores entre eles:
$$\sum_{i=1}^{j-1} x_i < x_j \text{ para } j = 2, \ldots, n$$
- Crie uma instância para o problema de códigos de Huffman com caracteres C_i e frequência x_i para $i = 1, \ldots, n$.
- Baseado no teorema, a solução será uma árvore onde podemos facilmente ordenar os números $x_1, \ldots, x_n$.

O que está faltando?

Código de Huffman

Caso geral:

- Ao fazer a redução $P_1 \triangleright P2$ devemos considerar uma instância genérica de P_1.
- Podemos salvar a nossa redução considerando uma entrada $x_1, \ldots, x_n$ qualquer para o problema de ordenação.
- Neste caso usamos frequências $2^{x_1}, 2^{x_2}, \ldots, 2^{x_n}$.
- Note que para qualquer valor j temos
$$\sum_{i=1}^{j-1} 2^i < 2^j$$
- Portanto a instância criada gera uma árvore como a desejada.