Projeto e Análise de Algoritmos

Redução entre problemas

Lehilton Pedrosa

Primeiro Semestre de 2020

Cotas para um problema

Formalizando problema

Como formalizar um problema genericamente?

Problema computacional

Um problema computacional é uma relação $P \subseteq \mathcal{I} \times \mathcal{S}$, onde

- $\mathcal{I}$ é o conjunto de entradas e
- $\mathcal{S}$ é o conjunto de saídas.

Algoritmo para um problema

Definição

Dizemos que um algoritmo $\textsc{Alg}$ resolve um problema $P = (\mathcal{I}, \mathcal{P})$ se para toda entrada $I \in \mathcal{I}$, ele devolve uma saída $S \in \mathcal{S}$ tal que $(I, S) \in P$.

- escrevemos $I \in P$ para representar uma entrada
- escrevemos $A(I)$ para representar a saída do algoritmo
- denotamos por n o "tamanho" de I
- normalmente n é o números de bits de I

Revisitando a complexidade de um algoritmo

Seja $\textsc{Alg}$ um algoritmo para um problema P e n um parâmetro.

Notação O:

- Se o algoritmo leva tempo no máximo $f(n)$ para toda entrada de tamanho n, então dizemos que $\textsc{Alg}$ executa em tempo $O(f(n))$.

Notação Ω:

- Se o algoritmo leva tempo pelo menos $g(n)$ para alguma entrada de tamanho n, então dizemos que $\textsc{Alg}$ executa em tempo $\Omega(g(n))$.

Cotas superior e inferior de um problema

Seja P um problema e seja n um parâmetro:

Cota superior

Uma função $f(n)$ é chamada de cota superior para P se existe algum algoritmo que resolve P em tempo $O(f(n))$.

Cota inferior

Uma função $g(n)$ é chamada de cota inferior para P se todo algoritmo que resolve P leva tempo $\Omega(f(n))$.

Algoritmo ótimo

Um algoritmo Alg é ótimo para um problema P se:

1. Alg resolve P em tempo $O(f(n))$ e
2. $f(n)$ é uma cota inferior de P.

- Heap-Sort e Merge-Sort são ótimos para ordenação:
  - eles têm complexidade $O(n \lg n)$
  - ordenação tem cota inferior $\Omega(n \lg n)$
- Busca-Binaria é ótimo para busca em vetor ordenado:
  - tem complexidade $O(\lg n)$
  - qualquer algoritmo leva tempo $\Omega(\lg n)$

Comparando problemas

Como comparamos dois algoritmos para um único problema?

- comparamos a complexidade de cada algoritmo
- descobrimos se um algoritmo é mais "rápido" que outro

E se quisermos comparar dois problemas A e B?

- queremos descobrir se então A é mais "fácil" do que B
- podemos comparar as cotas de cada algoritmo

Exemplo: Achar o máximo é mais fácil que ordenar um vetor!

- máximo tem cota superior $O(n)$
- ordenação tem cota inferior $\Omega(n \lg n)$

Reduções

Combinando problemas

Uma analogia

Um certo editor de literatura internacional é especialista em publicar livros em português. Ele conta com um time de tradutores, entre os quais:

- Cook, responsável por traduzir de **inglês para português**.
- Levin, responsável por traduzir de **russo para inglês**.

Em uma edição especial, ele irá publicar Crime e Castigo; como então traduzir de russo para português?

- Em geral, lidamos com problemas bem conhecidos
- Mas eventualmente, topamos com problemas novos

  Pergunta: como relacionar esses problemas?

Redução

Problema A:

- Instância: I_A
- Solução: S_A

Problema B:

- Instância: I_B
- Solução: S_B

Definição

Uma redução do problema A ao problema B é um par de sub-rotinas τ_I e τ_S tais que:

- τ_I transforma instância I_A de A em uma instância I_B de B
- τ_S transforma solução S_B de I_B em uma solução S_A de I_A.

Redução como um algoritmo

Como podemos resolver o problema A?

1. Suponha que existe um algoritmo $\textsc{Alg}_B$ para o problema B.
2. Podemos usar $\textsc{Alg}_B$ como uma caixa-preta

$\textsc{Redução}(I_A)$
$\quad I_B \leftarrow \tau_I(I_A)$
$\quad S_B \leftarrow \textsc{Alg}_B(I_B)$
$\quad S_A \leftarrow \tau_S(I_A, S_B)$
$\quad$ **devolva** S_A

Em outras palavras:

- se sei resolver B, então também sei resolver A!
- A é mais "fácil"que B
- denotamos $A \preccurlyeq B$

Um problema de origem

Problema da Alocação de Centros (AC)

Entrada:

- um grafo bipartido conexo $G = ((X \cup Y), E)$ e
- uma função de pesos nas arestas $\omega : E \to \mathbb{R}_+$

Saída:

- alocar cada vértice v em X a um vértice $\phi[v]$ em Y tal que o peso $\omega(v, \phi[v])$ seja mínimo

Um problema de destino

Problema do Caminho Mínimo (CM)

Entrada:

- grafo direcionado acíclico $G = (V, E)$
- função de peso $\omega : E \to \mathbb{R}_+$ nas arestas
- origem s.

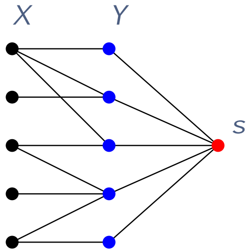
Saída:

- vetor d com $d[v] = \mathrm{dist}(s, v)$ para $v \in V$
- vetor π definindo uma árvore de caminhos mínimos

Reduzindo: transformação da entrada

Recebemos uma entrada do problema de origem AC:

$\tau_I(G, \omega)$
- $G' \leftarrow G$, $\omega' \leftarrow \omega$
- Adicione um novo vértice s a G'.
- **para cada** $v \in Y$ **faça**
    - Adicione aresta (s, v) a G'.
    - $\omega'(s, v) \leftarrow 0$
- **devolva** (G', ω', s)



Tempo: $O(Y)$

Reduzindo: transformação da saída

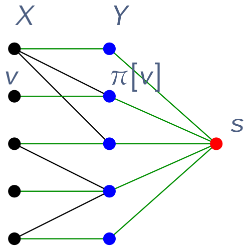
Também recebemos uma solução do problema de destino CM:

$\tau_S(G, \omega, d, \pi)$

para cada $v \in X$ **faça**

$\quad \phi[v] \leftarrow \pi[v]$

devolva ϕ



Tempo: $O(X)$

Reduzindo: AC $\preccurlyeq$ CM

Seja ALG_{CM} um algoritmo para Caminho Mínimo.

- ALG_{CM} poderia ser DIJKSTRA, BELLMAN-FORD...
- Pode ser que não conheçamos um algoritmo para o problema de destino!

REDUÇÃO-AC-CM(G, ω)
1 $(G', \omega', s) \leftarrow \tau_I(G, \omega)$
2 $(d, \pi) \leftarrow \text{ALG}_{\text{CM}}(G', \omega', s)$
3 $\phi \leftarrow \tau_S(G, \omega, d, \pi)$
4 **devolva** ϕ

Tempo total: [tempo da redução] + [tempo deALG_{CM}]

Tempo da redução

Quanto tempo gastamos só com a redução?

- não contamos o tempo do algoritmo para o problema B
- a complexidade de uma redução $f(n)$ é a soma dos tempos das transformações τ_I e τ_S
- escrevemos $A \leqslant_{f(n)} B$

Redução-AC-CM(G, ω)
1 $(G', \omega', s) \leftarrow \tau_I(G, \omega)$
2 $(d, \pi) \leftarrow \text{Alg}_{CM}(G', \omega', s)$
3 $\phi \leftarrow \tau_S(G, \omega, d, \pi)$
4 **devolva** ϕ

Nesse caso: $AC \leqslant_{|X|+|Y|} CM$

Parênteses: reduções polinomiais

Queremos construir algoritmos rápidos. Mas o que é "rápido"?

- normalmente, dizemos que um algoritmo é rápido se ele executa em tempo polinomial
- daí, queremos reduções de tempo polinomial
- nesse caso, escrevemos $A \leqslant_{\text{poli}} B$

Qual a consequência de $A \leqslant_{\text{poli}} B$?

1. se B tem algoritmo de tempo polinomial, então A também
2. se A não tem algoritmo de tempo polinomial, tampouco B

- Isso é útil para distinguir problemas fáceis de difíceis!
- Mas é assunto para depois...

Exemplos de reduções

Exemplos de reduções

- Sistema Linear (LS) para Sistema Linear Simétrico (SLS)

Problema de origem

Sistema Linear (LS)

Entrada:

- matriz M de dimensões $n \times n$ com determinante não nulo
- vetor b de dimensão n

Saída:

- vetor x de dimensão n que satisfaz o seguinte sistema linear:

$$Mx = b$$

Problema de destino

Sistema Linear Simétrico (SLS)

Entrada:

- matriz M simétrica de dimensões $n \times n$ com determinante não nulo
- vetor b de dimensão n

Saída:

- vetor x de dimensão n que satisfaz o seguinte sistema linear:

$$Mx = b$$

Perguntas

SLS é um caso particular de LS

- então trivialmente SLS $\leqslant$ LS
- será que LS é estritamente mais difícil?

A resposta é não!

- Iremos reduzir LS para SLS.
- Isso é, LS $\leqslant$ SLS.

Um fato simples

Lema

Um vetor x é solução de $Mx = b$ se, e só se, x é solução de $M^T Mx = M^T b$.

Demonstração:

$(\Rightarrow)$

- multiplicamos $Mx = b$ por M^T
- obtemos $M^T Mx = M^T b$

$(\Leftarrow)$

- M^T tem determinante não nulo
- então M^T tem inversa Z
- multiplicando $M^T Mx = M^T b$ por Z
- obtemos $Mx = b$

Observe que $M' = M^T M$ é uma matriz simétrica!

$\mathrm{LS} \preccurlyeq \mathrm{SLS}$

REDUÇÃO-LS-SLS(M, b)
1 $M' \leftarrow M^T M$
2 $b' \leftarrow M^T b$
3 $x \leftarrow \mathrm{ALG}_{\mathrm{SLS}}(M', b')$
4 **devolva** x

Concluímos que de fato $\mathrm{LS} \preccurlyeq \mathrm{SLS}$.

Exemplos de reduções

- Casamento Cíclico de Strings para Casamento de Strings

Problema de origem

Problema do casamento cíclico de strings (CSM)

Entrada:

- alfabeto Σ
- cadeia $A = a_0 a_1 \ldots a_{n-1}$ com n símbolos
- cadeia $B = b_0 b_1 \ldots b_{n-1}$ com n símbolos

Saída:

- decidir se B é um deslocamento cíclico de A
- se sim, então o número k de letras deslocadas

Exemplo:

- Entrada: $A = acgtact$ e $B = gtactac$
- Saída: TRUE, $k = 2$

Problema do casamento de strings (SM)

Entrada:

- alfabeto Σ
- cadeia $A = a_0 a_1 \dots a_{n-1}$ com n símbolos
- cadeia $B = b_0 b_1 \dots b_{m-1}$ com m símbolos

Saída:

- decidir se B é subcadeia de A
- se sim, qual índice k da primeira ocorrência de B em A

Exemplo:

- Entrada: $A = acgttaccgtacccg$ e $B = tac$
- Saída: TRUE, $k = 4$

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

CSM $\preccurlyeq$ SM

Redução-CSM-SM(A, B, n)

```
A' ← AA          ▷ concatena duas cópias de A
B' ← B
n' ← 2n
m' ← n
devolva Alg_SM(A', n', B', m')
```

- **Tempo da redução:** $O(n)$
- **Correção:** basta mostrar que se k é a solução de SM, então k também é solução de CSM.

Exemplo:

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

Exemplos de reduções

- Existência de Triângulo para Multiplicação de Matrizes Quadradas

Problema de origem

Problema da existência de triângulo (PET)

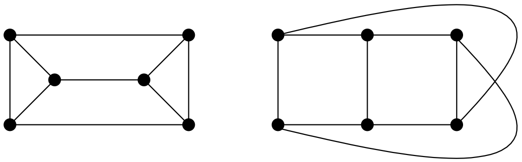
Entrada:

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

Saída:

- decidir se G contém um triângulo

Exemplo:



SIM NÃO

Observações sobre o PET

Alguns algoritmos conhecidos:

- um algoritmo trivial de tempo $O(n^3)$:
  - verificar todas as triplas de vértices
- um algoritmo $O(mn)$ que
  - é muito bom se o grafo é esparso

Vamos supor que o grafo é denso:

- G será representado por uma matriz de adjacência A

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

Um lema útil

Lema

Seja $A^2 = A \times A$, ou seja, $a^2_{ij} = \sum_{k=1}^{n} a_{ik} a_{kj}$.
Então $a^2_{ij} > 0$ se e somente se existe caminho de tamanho dois saindo de i e chegando em j.

Demonstração:

($\Rightarrow$)

- se $a^2_{ij} > 0$, então algum termo $a_{ik} a_{kj}$ é positivo
- segue que $a_{ik} = 1$ e $a_{kj} = 1$
- ou seja, há arestas (i, k) e (k, j)

($\Leftarrow$)

- seja um caminho (i, k, j) de i até j
- então $a_{ik} = 1$ e $a_{kj} = 1$
- daí $a_{ik} a_{kj} > 0$ e, portanto, $a^2_{ij} > 0$

Problema destino

Problema da Multiplicação de Matrizes Quadradas (MMQ)

Entrada:

- matriz quadrada A de ordem n.
- matriz quadrada B de ordem n.

Saída:

- produto $P = A \times B$.

Observações:

- há um algoritmo óbvio de complexidade $O(n^3)$
- MMQ pode ser resolvida mais rapidamente
  - em tempo $O(n^{2,807})$ pelo algoritmo de Strassen (1969)
  - em tempo $O(n^{2,3728639})$ pelo de François Le Gall (2014)

PET $\preccurlyeq$ MMQ

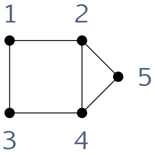
Observe que só existe triângulo com aresta (i,j) se

1. existir caminho de tamanho 2 de i a j
2. existir aresta (i,j)

```
Redução-PET-MMQ(A, n)
A² ← Alg_MMQ(A, n)
para i = 1 até n faça
    para j = 1 até n faça
        se a²_ij > 0 e a_ij = 1 então
            devolva SIM
devolva NÃO
```

- **Tempo da redução:** $O(n^2)$
- **Tempo total:** $O(n^{2,3728639})$

PET $\preccurlyeq$ 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

$A^2 = A(G) \times A(G)$

	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 para Multiplicação de Matrizes Simétricas

Considerando o caso particular

Considere um caso particular de MMQ:

Multiplicação de Matrizes Simétricas (MMS)

Entrada:

- matriz simétrica quadrada A de ordem m.
- matriz simétrica quadrada B de ordem m.

Saída:

- produto $P = A \times B$.

Claro que MMS $\preccurlyeq_{m^2}$ MMQ.

- Portanto, MMQ é pelo menos tão difícil quanto MMS
- Será que MMS também é pelo menos tão difícil quanto MMQ?

MMQ $\preccurlyeq$ MMS

Reduzindo MMQ $\preccurlyeq_{n^2}$ MMS:

1. Considere uma instância de MMQ, $I_{MMQ} = (A, B, n)$.
2. Construa uma instância de MMS, $I_{MMS} = (A', B', 2n)$, em que
$$A' = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \quad \text{e} \quad B' = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}.$$
3. A solução de MMS é:
$$P' = A'B' = \begin{bmatrix} AB & 0 \\ 0 & A^T B^T \end{bmatrix}.$$
4. Devolva o primeiro bloco da matriz P'.

Tempo da redução: $O(n^2)$.

- construir I_{MMQ} leva tempo $O(n^2)$
- copiar o bloco e P' leva tempo $O(n^2)$

Interpretando os fatos

- suponha que exista um algoritmo para MMS de tempo $O(T(m))$ para algum polinômio $T(m)$
  - relembre que m é a ordem das matrizes A' e B'
- quão pequeno pode ser $T(m)$?
  - é claro que $T(m) = \Omega(m^2)$, pois é preciso ler a entrada
  - será que pode ser mais rápido que $o(m^{2{,}3728639})$?

Para responder isso, usamos a redução MMQ $\preccurlyeq_{n^2}$ MMS:

- ela implica em um algoritmo de tempo total $O(T(m) + n^2)$
- como $m = 2n$, tempo é $O(T(2n) + n^2) = O(T(n) + n^2)$
- como $T(n)$ domina n^2, o tempo é simplesmente $O(T(n))$

Ou seja:

$\Rightarrow$ algoritmo $T(m)$ para MMS implica algoritmo $T(n)$ para MMQ!

Reduções para obtenção de cota inferior

Uma redução $A \preccurlyeq_{f(n)} B$

$\textsc{Redução}(I_A)$
1 $\quad I_B \leftarrow \tau_I(I_A)$
2 $\quad S_B \leftarrow \textsc{Alg}_B(I_B)$
3 $\quad S_A \leftarrow \tau_S(I_A, S_B)$
4 $\quad$ **devolva** S_A

Suponha que

- A tem cota inferior $h(n)$
- $\textsc{Alg}_B$ resolve B em tempo $g(n)$
- a redução gasta tempo $f(n) \leq \frac{h(n)}{2}$

$\textsc{Reducao}$ é um algoritmo para A de tempo

$$f(n) + g(n) \geq h(n) \Rightarrow g(n) \geq h(n) - f(n) \geq h(n) - \frac{h(n)}{2} = \frac{h(n)}{2}$$

Conclusão: $g(n) \geq \Omega(h(n))$

Transferindo cotas inferiores

Teorema

Considere dois problemas A e B e suponha que

1. $h(n)$ é cota inferior para A e
2. $A \preccurlyeq_{f(n)} B$,
3. $f(n) = o(h(n))$.

Então $h(n)$ é cota inferior para B.

Observação:

- a cota inferior depende do modelo de computação
- supomos o mesmo modelo para ambos problemas

Reduções para obtenção de cota inferior

- Ordenação para Envoltória Convexa

Problema de origem

Problema da Ordenação (ORD)

Entrada:

- sequência $X = (x_1, x_2, \ldots, x_n)$ de n elementos comparáveis

Saída:

- permutação X cujos elementos estejam ordenados

Observações:

- só podemos comparar dois elementos por uma sub-rotina caixa-preta de tempo constante (chamada oráculo)
- esse problema tem cota inferior $\Omega(n \lg n)$

Problema de destino

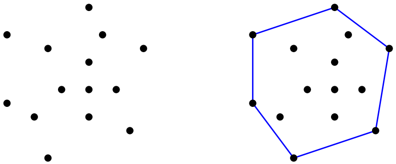
Problema da Envoltória Convexa (EC)

Entrada:

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

Saída:

- **menor polígono convexo** que contém os n pontos



Observações:

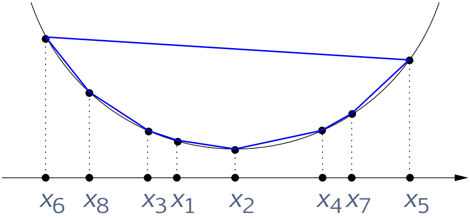
- os vértices são representados em ordem anti-horária
- problema clássico de Geometria Computacional
- pode ser resolvido em tempo $O(n \lg n)$

ORD $\preccurlyeq_n$ EC

Reduzindo ORD $\preccurlyeq_n$ EC:

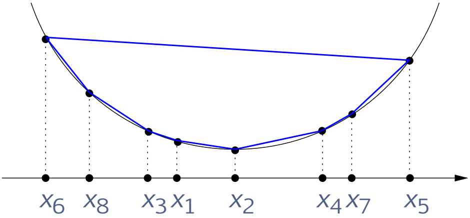
1. Considere uma instância de $I_{ORD} = (x_1, x_2 \ldots, x_n)$.
2. Construa instância

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



ORD $\preccurlyeq_n$ EC (cont)

3. Resolva I_{EC} e obtenha solução S_{EC}, que é uma lista cíclica dos vértices do polígono.



4. Determine índice i de S_{EC} do ponto com menor abcissa.
5. Liste os todos os índices a partir de i.

- O tempo da redução é $O(n)$.
- Portanto $\Omega(n \lg n)$ também é cota inferior para EC.

Reduções para obtenção de cota inferior

- Unicidade de Elementos para Ponto Mais Próximo em 2D

Problema de origem

Problema da Unicidade de Elementos (UE)

Entrada:

- sequência $X = (x_1, x_2, \ldots, x_n)$ de n elementos comparáveis

Saída:

- decidir se os elementos são todos distintos

Observações:

- só podemos comparar dois elementos por uma sub-rotina caixa-preta de tempo constante (chamada oráculo)
- esse problema tem cota inferior $\Omega(n \lg n)$
- o problema pode ser resolvido em tempo $O(n \lg n)$. (Como?)

Problema de destino

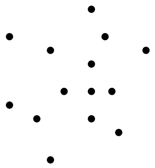
Problema do Par Mais Próximo (PMP)

Entrada:

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

Saída:

- par de pontos i e j que estejam a menor distância.



Observação:

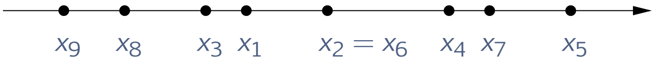
- pode ser resolvido em tempo $O(n \lg n)$

UE $\preccurlyeq_n$ PMP

Reduzindo UE $\preccurlyeq_n$ PMP:

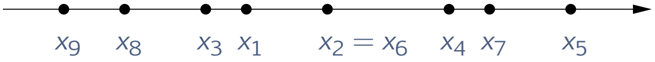
1. Considere instância $I_{UE} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância

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



UE $\preccurlyeq_n$ PMP (cont)

3. Resolva I_{PMP} e obtenha par de pontos $(x_i, 0), (x_j, 0)$.
4. Calcule a distância d entre os pontos:
   (a) Se $d = 0$, então devolva NÃO.
   (b) Se $d > 0$, devolva SIM.



- O tempo da redução é $O(n)$.
- Portanto $\Omega(n \lg n)$ também é cota inferior para PMP.

Reduções para obtenção de cota inferior

- 3-Soma para Colinearidade

Problema de origem

Problema da 3-Soma (3SUM)

Entrada:

- sequência $X = (x_1, x_2, \ldots, x_n)$ de n reais

Saída:

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

$$x_i + x_j + x_k = 0$$

Exemplo:

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

Observações:

- pode ser resolvido em $O(n^2)$ (Como?)
- acreditava-se que $\Omega(n^2)$ era cota inferior
- pode ser resolvido em $o(n^2)$ (Grønlund e Pettie, 2014)
- ainda se acredita que não dá pra fazer melhor que $n^{2-\Omega(1)}$

Problema de destino

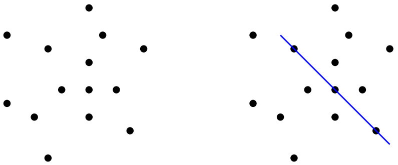
Problema da Colinearidade Não Horizontal (COL)

Entrada:

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

Saída:

- determinar se três pontos estão em alguma reta não horizontal



Observações:

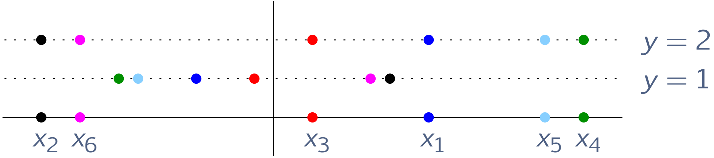
- pode ser resolvido em tempo $O(n^2)$
- acredita-se que $\Omega(n^2)$ é cota inferior

$3\text{SUM} \preccurlyeq_n \text{COL}$

Reduzindo $3\text{SUM} \preccurlyeq_n \text{COL}$:

1. Considere instância $I_{3SUM} = (x_1, x_2, \ldots, x_n)$.
2. Construa instância

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



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

3. Resolva I_{COL} e obtenha S_{COL}.
4. Se a resposta S_{COL} for SIM, responda SIM;
   mas se a resposta S_{COL} for NÃO, responda NÃO.

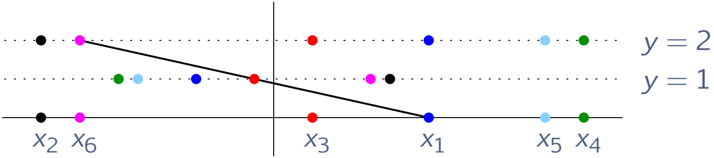
3SUM $\preccurlyeq_n$ COL (cont)

- 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 $\preccurlyeq_n$ COL

- É claro que $\Omega(n)$ é uma cota inferior para 3SUM.
- E se houver cota inferior $\Omega(h(n))$ maior para 3SUM?
  - a redução gasta tempo $f(n) = O(n)$
  - nesse caso, $f(n) = o(h(n))$
  - então $\Omega(h(n))$ seria cota inferior para COL
- Mas só conhecemos a cota trivial $\Omega(n)$ para 3SUM.

Exercício

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

1. Mostre que 3SUM $\leqslant_n$ 3SUMplus.
2. Mostre que 3SUMplus $\leqslant_n$ 3SUM.
3. Suponha que o Professor Sabit Udo descobriu uma cota inferior de $\Omega(n^{1{,}9})$ para 3SUMplus. Nesse caso, quais das afirmações abaixo 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.

Outros exemplos de reduções

Outros exemplos de reduções

- Sistema de Representantes Distintos (SRD) para Emparelhamento Máximo (EM)

Problema de origem

Sistema de Representantes Distintos (SRD)

Entrada:

- uma coleção de conjuntos $S_1, \dots, S_k$

Saída:

- conjunto $R = \{r_1, \dots, r_k\}$ tal que $r_i \in S_i$ para $i = 1, \dots, k$

- R é chamado de sistema de representantes distintos

Exemplos

Um exemplo:

- considere o sistema de conjuntos

  Ecológicos: Ana, Alberto
  Ruralistas: João, Alberto
  Feministas: Ana, Maria

- então $\{$Ana, João, Maria$\}$ é um SRD

Um não exemplo:

- considere o sistema de conjuntos

$$\begin{aligned} S_1 &= \{1, 2\} \\ S_2 &= \{3, 4\} \\ S_3 &= \{3, 4\} \\ S_4 &= \{1, 2, 4\} \\ S_5 &= \{2, 4\} \end{aligned}$$

- não existe SRD para esse sistema

Um fato útil

Teorema de Hall

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

$$\left|\left\{S_{i_1} \cup \cdots \cup S_{i_m}\right\}\right| \geq m$$

para toda coleção $\{i_1, \ldots, i_m\} \subseteq \{1, 2, 3, \ldots, k\}$.

- i.e., cada subcoleção de m conjuntos tem m elementos
- isso sugere que basta testar todas possíveis subcoleções
- mas o número de subcoleções é 2^k

Problema de destino

Problema do Emparelhamento Máximo (EM)

Entrada:

- grafo bipartido $G = (X \cup Y, E)$ com bipartição X e Y

Solução:

- subconjunto de arestas M que não compartilham vértices

Objetivo:

- maximizar $|M|$

SRD $\preccurlyeq$ EM

Redução:

1. Tome uma instância de $S_1, \ldots, S_k$ de SRD sobre A
2. Construa um grafo bipartido $G = (X \cup Y, E)$ com
   - $X = S_1 \cup \cdots \cup S_k$
   - $Y = \{1, 2, \ldots, k\}$
   - $E = \{(a, j) \ : \ a \in S_j \text{ e } 1 \leq j \leq k\}$
3. Encontre um emparelhamento máximo de G

Teorema

SRd $\preccurlyeq$ EM

- note que existe um SRD se e somente se G tem emparelhamento com k arestas

Outros exemplos de reduções

- Edição de String para Problema do Caminho Mínimo

Problema de origem

Considere as seguintes operações sobre strings:

- inserção de um caractere
- remoção de um caractere
- troca de um caractere por outro

Edição de String

Entrada:

- duas strings A e B

Solução:

- sequência de operações para transformar A em B

Objetivo:

- minimizar número de operações

Exemplo

Considere strings $A = babb$ e $B = bbc$

- podemos transformar A em B

$$\begin{array}{cl} babb & \\ \Downarrow & \text{remova } a \\ bbb & \\ \Downarrow & \text{troque o último } b \text{ por } c \\ bbc & \end{array}$$

- realizamos duas operações

Observação:

- pode ser resolvido por programação dinâmica (exercicio)

Problema de destino

Problema do Caminho Mínimo

Entrada:

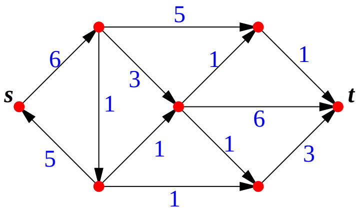
- grafo direcionado $G(V, E)$
- peso $c_{ij} \geq 0$ para cada aresta $(i, j) \in E$
- vértices s e t

Solução:

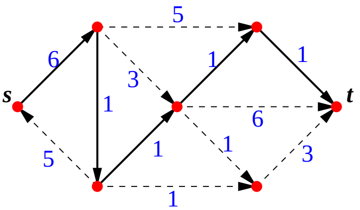
- caminho de s a t em G

Objetivo:

- minimizar o comprimento do caminho



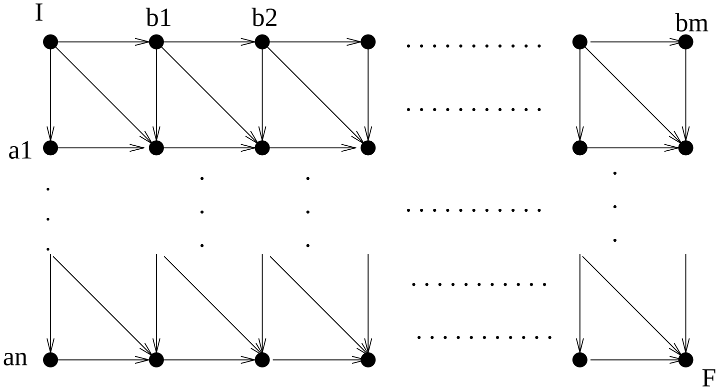
G(V,E,w)



Um caminho minimo de s para t

Redução:

1. Tome duas strings $A = a_1 a_2 \ldots a_n$ e $B = b_1 b_2 \ldots b_m$
2. Construa um grafo da seguinte forma:



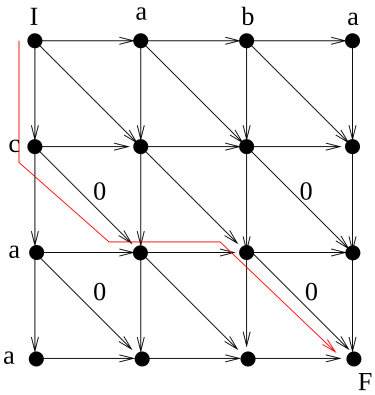
3. Encontre um caminho mínimo de I até F

Edição de String

4. Converta o caminho em operações de strings:
   - 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.
5. Devolva a sequência de operações construída

Exemplo da redução

Considere strings $A = caa$ e $B = aba$ e o grafo G



- há duas operações de custo não nulo
- o custo de edição corresponde a $1 + 0 + 1 + 0 = 2$

Outros exemplos de reduções

- Ordenação para Codificação de Huffman

Problema de origem

Ordenação

Entrada:

- sequência de números naturais distintos $x_1, x_2, \ldots, x_n$

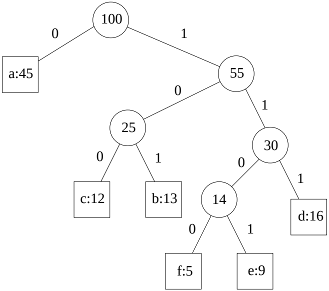
Saída:

- permutação dos números de entrada

Arvore binária para codificação variável

Relemebrando: codificação de comprimento variável

	a	b	c	d	e	f
Frequência	45	13	12	16	9	5
Código variável	0	101	100	111	1101	1100



Problema de destino

Codificação de Huffman

- Entrada:
  - alfabeto C
  - tabela de frequências f
- Solução:
  - codificação de comprimento variável
- Objetivo:
  - minimizar o tamanho do texto codificado

Observação

- já vimos um algoritmo $O(n \log n)$ para o problema
- vamos utilizá-lo para resolver ordenação

Lema útil

Lema

Considere caracteres $C_1, C_2, \ldots, C_n$ com frequências $f_1, f_2, \ldots, f_n$ tais que

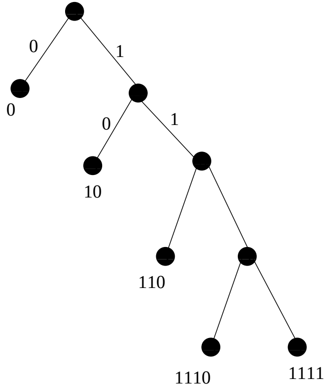
$$\sum_{i=1}^{j-1} f_i < f_j \text{ para cada } j = 2, \ldots, n.$$

Então existe uma árvore de Huffman tal que

- C_n é folha no nível 1,
- C_{n-1} é folha no nível 2
- $\ldots$
- C_2 e C_1 são folhas no nível $n-1$.

Exemplo de árvore

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



Prova do lema

Demonstração:

- seja T^* uma árvore de Huffman para instância enunciada
- suponha por contradição que C_n não esteja no nível 1
- adicione uma nova raiz em T^*
  - torne C_n filho esquerdo da nova raiz
  - e faça T^* filho direto dessa raiz
- o custo da árvore muda
  - ele diminuiu de pelo menos f_n
  - e aumenta de no máximo $\sum_{i=1}^{n-1} f_i$
- portanto o custo diminui
- isso é um absurdo porque T^* é codificação ótima
- podemos repetir o argumento para os demais caracteres

Redução

1. Tome uma instância de ordenação $x_1, x_2, \ldots, x_n$.
2. Crie caracteres $C_1, C_2, \ldots, C_n$ com frequências $f_1 = 2^{x_1}, f_2 = 2^{x_2}, \ldots, f_n = 2^{x_n}$.
3. Observe que para todo caractere j temos

$$\sum_{i=1}^{j-1} 2^i < 2^j$$

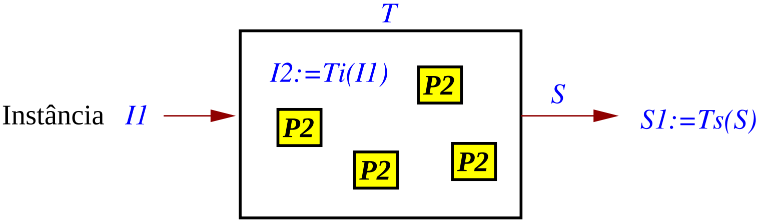
4. Encontre um codificação de huffman para a instância criada.
5. Percorra a árvore em ordem de níveis, listando as folhas percorridas.

Outros exemplos de reduções

- Reduções de Turing

Redução de Turing

Podemos reduzir 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.

Redução:

- Podemos reduzir Multiplicação de Inteiros para Quadrado.
- Fazemos apenas um número constante de somas, subtrações e divisão por dois

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