MC448 — Análise e Projeto de Algoritmos

Prof. Pedro J. de Rezende

2o. Semestre de 2003

Versão de 27.04.2002

Modelo Computacional e Redução entre Problemas *

1 Introdução

Neste capítulo estudaremos modelos computacionais, redução de problemas e projeto de algoritmos por indução. Nesse momento, pode não ser claro para o leitor como esses assuntos se relacionam ao estudo de complexidade de algoritmos e por isso daremos aqui uma breve explicação.

De maneira bastante informal, podemos definir um algoritmo A para um problema P como uma sequência de passos a serem efetuados para que se obtenha uma solução para P. Fazer a análise de complexidade de um algoritmo A envolve avaliar o custo computacional das operações efetuadas em cada passo. No entanto, o custo de uma dada operação pode variar conforme o modelo computacional no qual o algoritmo é representável, ou seja, a complexidade do algoritmo depende do modelo computacional em que é implementado. Portanto, só faz sentido comparar a complexidade de algoritmos distintos para um mesmo problema P se considerarmos implementações dos algoritmos em um modelo computacional comum. É por isso que a partir desse momento analisaremos algoritmos com relação a um mesmo modelo computacional. A definição de um modelo computacional para a análise de um problema também é essencial para se avaliar a dificuldade intrínseca do problema P, ou seja, qual é a complexidade mínima que deve ter um algoritmo arbitrário que resolve P.

O mecanismo de redução de problemas relaciona dois problemas, permitindo a inferência de características potencialmente desconhecidas de um destes, tais como quota inferior ou a existência de um algoritmo, a partir das informações conhecidas do outro. Conforme veremos na Seção 4 existe uma série de condições que devem ser satisfeitas para que possamos fazer as inferências mencionadas. Portanto, o mecanismo de redução é bastante útil quando estudamos problemas novos, pois através dele podemos adquirir rapidamente conhecimento sobre o novo problema a partir do conhecimento previamente obtido de problemas já estudados.

Conforme vimos no capítulo anterior, as demonstrações por indução são construtivas e, portanto, sugerem algoritmos para a solução de problemas. É por isso que estudaremos como projetar algoritmos a partir de demonstrações indutivas de que é possível se resolver um problema. Além disso, ao construirmos

*Escriba: Cândida Nunes da Silva.

o algoritmo dessa forma, ainda ganhamos uma demonstração de que o algoritmo está correto, ou seja, de fato resolve o problema em questão.

2 Modelo RAM e Árvores de Decisão

Conforme mencionamos na introdução, a análise de complexidade de um algoritmo, bem como a determinação da dificuldade intrínseca de um problema, depende do modelo computacional considerado. De agora em diante vamos desenvolver e analisar algoritmos no modelo RAM (Random Access Machine). Uma descrição bastante completa desse modelo é dada em [2] e por isso não a repetiremos aqui.

O modelo RAM foi escolhido por ser um modelo de máquina bastante simples no qual operações elementares como somas, subtrações, multiplicações, divisões e comparações podem ser feitos em tempo constante, assim como na maioria dos computadores atuais. Assim, a análise dos algoritmos que fizermos nesse modelo corresponderão à complexidade dos programas efetivamente implementados.

Além disso, há uma série de outros modelos computacionais que podem ser simulados no modelo RAM. Por exemplo, árvores de decisão são abstrações do modelo RAM que representam a computação na forma de uma árvore binária onde cada nó indica uma decisão tomada. Primeiramente, a comparação representada dentro de um nó é efetuada e então, conforme o resultado dessa comparação, a computação segue para o descendente à direita ou à esquerda deste nó, enquanto as folhas representam as possíveis soluções computadas.

Para efetuar as comparações internas a cada nó, é permitida a computação de uma função dos elementos de entrada da computação. Essa função pode ser simplesmente a função projeção que só permite comparação direta entre elementos da entrada, ou uma função linear que permite a soma de elementos da entrada ou produtos por constantes, ou uma função quadrática que permite o produto de pares de elementos da entrada. Enfim, essa função pode ser arbitrária, desde que seja possível calculá-la com um número constante de operações no modelo RAM. Conforme o tipo da função calculada em cada nó damos um nome específico para a árvore de decisão: árvores de decisão binárias são aquelas que admitem apenas a função projeção, árvores de decisão lineares são aquelas que admitem apenas funções lineares, árvores de decisão quadráticas são aquelas que admitem apenas funções quadráticas, e assim por diante.

Na Seção 3 apresentada a seguir veremos como é possível demonstrar quotas inferiores para certos problemas nos modelos de árvores de decisão. Para que uma demonstração de quota inferior de um problema em um modelo de árvore de decisão seja útil, é necessário, em primeiro lugar, que o problema seja solúvel no modelo dado. Por exemplo, o problema de determinar quais pontos de um conjunto de n pontos no plano está acima de uma dada reta (fixa), não pode ser resolvido no modelo de árvore de decisões binárias pois exige a computação do produto do coeficiente da reta pela abscissa do ponto de entrada. Portanto, é necessário utilizar pelo menos o modelo de árvore de decisões lineares para resolver esse problema.

Como lidaremos de agora em diante com diversos problemas que podem exigir a utilização de diferentes modelos, optamos por trabalhar com o modelo de árvore de decisões algébricas, no qual é permitido que a função em cada nó seja uma função algébrica, e, portanto, este é um modelo mais geral que os mencionados anteriormente.

Uma função $f : \mathbb{A}^n \to \mathbb{R}$ é *algébrica* se $f(x)$ é:

- um polinômio com coeficientes inteiros, ou
- o quociente de duas funções algébricas;

onde $\mathbb{A}$ é o conjunto do números algébricos. Podemos definir o conjunto dos *números algébricos* da seguinte forma:

Um número real é chamado algébrico *se ele é raiz de algum polinômio* $P(x) = 0$ *de coeficientes inteiros.*

Note que todo número racional é algébrico, e que alguns irracionais (mas não todos) também o são. Os números (irracionais) não algébricos são chamados de transcendentes e denotados por $\mathbb{T}$. Assim, temos: $\mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R}$, $\mathbb{R} = \mathbb{A} \dot{\cup} \mathbb{T}$, $\sqrt{2} \in \mathbb{A}$, $(\frac{p}{q})^{\frac{1}{r}} \in \mathbb{A}, \forall\ p, q, r \in \mathbb{Z}^*$, porém $\pi \in \mathbb{T}$, $e \in \mathbb{T}$ (veja [1]).

Além disso, conforme enunciado no Teorema 1, o qual não demonstraremos aqui, toda função algébrica f com domínio $\mathbb{A}^n$ possui imagem $\mathbb{A}$. Portanto, podemos considerar que em nossa definição de função algébrica temos $f : \mathbb{A}^n \to \mathbb{A}$.

Teorema 1 *Seja* $f : \mathbb{A}^n \to \mathbb{R}$ *uma função algébrica. Então, para qualquer* $(a_1, a_2, \ldots, a_n) \in \mathbb{A}^n$*, temos que*

$$f(a_1, a_2, \ldots, a_n) \in \mathbb{A}.$$

Perceba que a definição de função algébrica inclui a função projeção, funções lineares, quadráticas, etc. Portanto, a demonstração de uma quota inferior para um problema no modelo de árvore de decisões algébricas vale para qualquer um dos outros modelos mais fracos. A recíproca não necessariamente é verdadeira.

3 Quotas Inferiores e Superiores

Nesta seção, definiremos os conceitos de quota superior e inferior e demonstraremos quotas inferiores para dois problemas no modelo de árvore de decisões algébricas.

Quota Superior:

Seja P um problema e A_P um algoritmo para a solução de P em um modelo computacional conveniente M. Seja $T(n)$ uma função que expressa a eficiência de A_P no modelo M. Dizemos que $T(n)$ é uma quota superior para P em M.

Quota Inferior:

Seja P um problema e M um modelo computacional no qual acredita-se que é possível resolver P (mesmo que nenhuma solução seja conhecida). Seja $I(n)$ uma função que expressa a eficiência mínima de qualquer algoritmo que resolva P no modelo M. Dizemos que $I(n)$ é uma quota inferior para P em M.

Nos modelos de árvore de decisão a dificuldade mínima de resolução de um problema pode ser calculada como a altura mínima da árvore que representa um algoritmo arbitrário que resolve o problema. Em outras palavras, é o número mínimo de comparações que devem ser efetuadas para se resolver uma instância qualquer de P.

Por exemplo, considere o seguinte problema P:

Dado um vetor ordenado V de n números algébricos e um valor algébrico x, determinar a posição de x em V, se x estiver em V, ou indicar que não está, no caso contrário.

Esse problema certamente pode ser resolvido no modelo de árvore de decisões algébricas. Qualquer árvore de decisões algébricas que modela um algoritmo que resolva esse problema deve conter no mínimo $n+1$ folhas, que é o número de possíveis respostas para P. Então, a altura de uma tal árvore é, no mínimo, $\log_2(n+1)$. Portanto, esse problema possui quota inferior $\Omega(\log n)$ no modelo de árvore de decisões algébricas.

Considere agora P como sendo o:

Problema da Ordenação (Ord)

Dado um vetor V de n números algébricos retornar uma permutação dos índices tal que a seqüência dos números do vetor segundo essa permutação está ordenada.

Novamente, P pode ser resolvido no modelo de árvore de decisões algébricas e qualquer árvore de decisões algébricas que modela um algoritmo que resolva o problema da ordenação deve conter no mínimo $n!$ folhas, que é o número de possíveis respostas para P. Portanto, esse problema possui quota inferior $\Omega(\log(n!))$ no modelo de árvore de decisões algébricas. Isso equivale a dizer que o problema da ordenação possui quota inferior $\Omega(n \log n)$, pois, conforme demonstraremos a seguir, $\log(n!) \in \Omega(n \log n)$.

$$\begin{aligned}
\log(n!) &= \log \prod_{i=1}^{n} i \\
&= \sum_{i=1}^{n} \log i \\
&= \log 1 + \log 2 + \log 3 + \cdots + \log(\frac{n}{2}) +
\end{aligned}$$

$$
\begin{aligned}
& + \log n + \log(n-1) + \log(n-2) + \cdots + \log(\frac{n}{2}+1) \\
= \quad & \log n + \log(2n-2) + \log(3n-6) + \cdots + \log(\frac{n^2}{4}+\frac{n}{2})
\end{aligned}
$$

Se conseguirmos demonstrar que, para $n \geq n_0$, cada parcela dessa soma é maior ou igual a $c \log n$, onde c é uma constante positiva, a demonstração estará concluída. Ou seja, queremos demonstrar que, para algum $c > 0$,

$$\log k + \log(n-k+1) \geq c \log n,$$

para $1 \leq k \leq n$. Vejamos:

$$
\begin{aligned}
\log k + \log(n-k+1) &= \log(nk - k^2 + 1) \\
&\geq \log(nk - k^2) \\
&= \log(k(n-k)) \\
&\geq \log(\frac{n}{2}), \text{ pois } k \geq 1 \text{ e } n-k \geq \tfrac{n}{2} \\
&= \log n - 1 \\
&\geq \frac{\log n}{2}, \text{ para } n \geq 4
\end{aligned}
$$

Portanto, a inequação é de fato verdadeira para $c = \frac{1}{2}$ e $n \geq 4$.

Logo, temos que, para $n \geq n_0 = 4$,

$$
\begin{aligned}
\sum_{i=1}^{n} \log i &\geq \sum_{i=1}^{\frac{n}{2}} \frac{\log n}{2} \\
&= \frac{1}{2} \sum_{i=1}^{\frac{n}{2}} \log n \\
&= \frac{1}{4} \sum_{i=1}^{n} \log n \\
&= \frac{1}{4} n \log n.
\end{aligned}
$$

CQD

4 Redução de Problemas

O conhecimento de classes de funções ω, Ω, Θ, O e o é pré-requisito para uma boa compreensão desta seção. As referências [3, 6] trazem explicações bastante completas sobre estas classes de funções. Uma explicação sucinta sobre redução de problemas também pode ser encontrada em [6].

Sejam P e Q dois problemas e $\mathbb{I}_P$, $\mathbb{I}_Q$, $\mathbb{S}_P$ e $\mathbb{S}_Q$ os conjuntos de instâncias e de soluções de P e de Q, respectivamente. Dizemos que P é redutível a Q se existem funções τ_I e τ_S:

$$\begin{aligned} \tau_I &: \mathbb{I}_P \to \mathbb{I}_Q \\ \tau_S &: \overline{\mathbb{S}_Q} \to \mathbb{S}_P \end{aligned}$$

de modo que se S_Q é solução de $\tau_I(I_P)$, então $\tau_S(S_Q)$ é solução de I_P, onde $\overline{\mathbb{S}_Q} \subseteq \mathbb{S}_Q$ é o conjunto das soluções das instâncias $\tau_I(\mathbb{I}_P) \subseteq \mathbb{I}_Q$. Em outras palavras, a função τ_I leva instâncias de P em instâncias de Q e τ_S mapeia soluções de Q (para as instâncias de P transformadas por τ_I) em soluções de P, conforme ilustrado pela Figura 1.

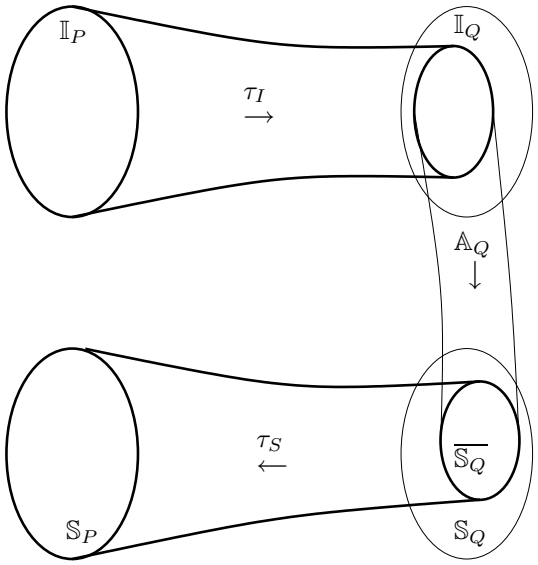

Figura 1: Redução do problema P ao problema Q

Dessa forma, se existe um algoritmo A_Q que resolve Q, então

$$A_P = \tau_S \circ A_Q \circ \tau_I$$

é um algoritmo que resolve P. Ou seja, dada uma instância $I_P \in \mathbb{I}_P$, temos que

$$A_P(I_P) = \tau_S(A_Q(\tau_I(I_P))) \tag{1}$$

é uma solução de I_P.

Seja n o tamanho das entradas de P e de Q e $T_{\tau_I}(n)$ e $T_{\tau_S}(n)$ funções que representam o comportamento assintótico da complexidade das transformações τ_I e τ_S. Se

$$T_{\tau_I}(n) + T_{\tau_S}(n) \in O(f(n)),$$

então, dizemos que P é redutível em tempo $f(n)$ a Q e denotamos este fato por

$$P \propto_{f(n)} Q.$$

A partir do conceito de redução apresentado acima, podemos tirar duas conclusões interessantes:

1. Se Q tem quota superior $O(g(n))$ e $P \propto_{f(n)} Q$, então P tem quota superior $O(g(n) + f(n))$.

   Exercício: Analisando a equação (1), prove esta afirmação.

   ---

2. Se P tem quota inferior $\Omega(h(n))$ e se $P \propto_{f(n)} Q$ e $f(n) \in o(h(n))$, então Q tem quota inferior $\Omega(h(n))$.

   Exercício: Analisando a equação (1), prove esta afirmação.

Portanto, o mecanismo de redução de problemas provê uma forma mais simples de determinar quotas inferiores de problemas do que a que apresentamos na Seção 3. Além disso, também é possível desenvolver algoritmos para problemas novos através da redução destes a problemas para os quais já conhecemos soluções.

Exercício: Prove que redutibilidade é transitiva.

A seguir utilizaremos redução para encontrar quotas inferiores para diversos problemas.

Problema da Envoltória Convexa (EC)

Dados n pontos em $\mathbb{A}^2$, construir o menor polígono convexo que os contém.

Vamos supor que a resposta desejada para esse problema é a seqüência ordenada dos vértices do polígono. Por enquanto, só conhecemos quota inferior para dois problemas, o de busca em vetor ordenado e o de ordenação, no modelo de árvore de decisões algébricas. Como a quota inferior do problema de busca é $\Omega(\log n)$ e o problema da envoltória convexa requer $\Omega(n)$, não é interessante reduzir o problema de busca ao da envoltória convexa. Então, vamos tentar reduzir o problema da ordenação ao problema da envoltória convexa.

Sabemos que $\Omega(n \log n)$ é uma quota inferior para o problema de ordenação no modelo de árvore de decisões algébricas. Será que existe $f(n) \in o(n \log n)$ tal que $Ord \propto_{f(n)} EC$?

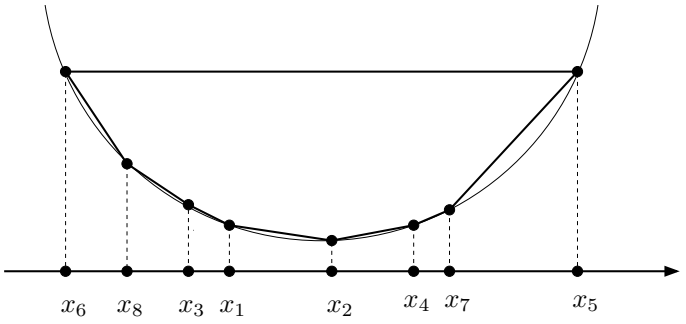

Figura 2: A função τ_I na redução de ordenação à envoltória convexa

Considere a seguinte função τ_I: dada uma instância $I_{Ord} = (x_1, x_2, \ldots, x_n)$, τ_I a leva ao conjunto de pontos $I_{EC} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ onde $y_i = x_i^2$, para $1 \leq i \leq n$. A Figura 2 ilustra a transformação τ_I.

A função τ_S é muito simples. Basta varrer a lista de vértices retornada até encontrar o vértice de menor abscissa e, a partir desse ponto, retornar seqüencialmente as abscissas dos vértices que formam a envoltória convexa.

Exercício: Demonstre para esta redução que $\tau_S(A_{EC}(\tau_I(I_{Ord})))$ é de fato uma solução de I_{Ord}.

A função τ_I é $O(n)$ pois apenas computa x_i^2 para $1 \leq i \leq n$. A função τ_S é $O(n)$ pois varre no máximo duas vezes a lista de vértices. Logo $f(n) \in O(n)$ e, portanto, $f(n) \in o(n \log n)$. Então, $\Omega(n \log n)$ é quota inferior para o problema da envoltória convexa no modelo de árvore de decisões algébricas.

Note que a quota inferior de $\Omega(n \log n)$ para o problema da ordenação também vale para o modelo mais simples de árvore de decisões binárias, no entanto, não faz sentido dizer que esta quota inferior também vale para o problema da envoltória convexa *nesse modelo* pois o problema da envoltória convexa não pode ser resolvido nesse modelo.

Exercício: Por que *EC* não pode ser resolvido no modelo de árvore de decisões binárias?

Existe uma classe de problemas, denominados *problemas de decisão*, cuja resposta é apenas sim ou não, ou seja, possui cardinalidade 1. Não é possível reduzir um problema P cuja solução tem cardinalidade $\omega(1)$ a um problema de decisão Q pois a função τ_S teria que ser capaz de construir a solução de P a partir da solução de Q.

Exercício: Por que esta dificuldade não surge quando reduzimos um problema de decisão P a um problema que não é de decisão Q?

Portanto, para que possamos demonstrar quotas inferiores para problemas de decisão através de redução é fundamental conhecermos uma quota inferior para algum problema de decisão. O problema da unicidade de elementos apresentado a seguir é um problema de decisão que possui quota inferior $\Omega(n \log n)$ no modelo de árvore de decisões algébricas.

Problema da Unicidade de Elementos (UE)

Dada uma coleção de n objetos de um domínio ordenado, determinar se eles são todos distintos.

Não demonstraremos aqui a quota inferior $\Omega(n \log n)$ para o problema da unicidade de elementos, mas esta pode ser encontrada em [4]. Reduzindo o problema da unicidade de elementos ao problema enunciado a seguir, que também é um problema de decisão, conseguimos demonstrar que este novo problema também possui quota inferior $\Omega(n \log n)$.

Problema das Retas Paralelas (RP)

Dada uma coleção de n retas no plano, determinar se há duas delas paralelas.

Queremos achar $f(n) \in o(n \log n)$ tal que $UE \propto_{f(n)} RP$. Seja τ_I a função que leva os pontos $(a_1, a_2, \ldots, a_n)$ de uma instância I_{UE} em uma instância de I_{RP} com as retas $y_i = a_i x$, para $1 \leq i \leq n$. A função τ_S simplesmente leva a resposta sim de S_{RP} na resposta não de S_{UE} e a resposta não de S_{RP} na resposta sim de S_{UE}.

Como $T_{\tau_I}(n)$ é $O(n)$ e $T_{\tau_S}(n)$ é $O(1)$, então $f(n) \in o(n \log n)$ e, portanto, $\Omega(n \log n)$ é quota inferior para o problema das retas paralelas no modelo de árvore de decisões algébricas.

Exercício: Demonstre para esta redução que $\tau_S(A_{RP}(\tau_I(I_{UE})))$ é de fato uma solução de I_{UE}.

Problema do Par mais Próximo em Uma Dimensão (PMP[1])

Dada uma coleção de valores algébricos $(x_1, x_2, \ldots, x_n)$*, determinar*

$$\min\{|x_i - x_j| : 1 \leq i, j \leq n \text{ e } i \neq j\}.$$

O problema da unicidade de elementos também pode ser reduzido ao problema do par mais próximo em uma dimensão. Basta tomar τ_I como a função identidade, ou seja, τ_I mapeia uma instância $I_{UE} = (x_1, x_2, \ldots, x_n)$ na instância $I_{PMP^1} = (x_1, x_2, \ldots, x_n)$. A solução de PMP^1 sempre é um valor real (na verdade, é algébrico) não negativo. Portanto, τ_S simplesmente leva uma resposta nula de S_{PMP^1} na resposta não de S_{UE} e uma resposta estritamente positiva de S_{PMP^1} na resposta sim de S_{UE}. Note que o problema do par mais próximo não é um problema de decisão, mas sua resposta também possui cardinalidade um.

Exercício: Demonstre para esta redução que $\tau_S(A_{PMP^1}(\tau_I(I_{UE})))$ é de fato uma solução de I_{UE}.

Exercício: É possível reduzir um problema de decisão P a um problema Q cuja solução possui cardinalidade $O(n)$, onde n é a cardinalidade da entrada de Q e de P?

Exercício: Prove que se $\Omega(h(n))$ é quota inferior para um problema P em uma dimensão k, então $\Omega(h(n))$ também é quota inferior para P em qualquer dimensão $l \geq k$.

Problema do Par mais Distante em Uma Dimensão ($\mathbf{PMD^1}$)

Dada uma coleção de valores algébricos $(x_1, x_2, \ldots, x_n)$*, determinar*

$$\max\{|x_i - x_j| : 1 \leq i, j \leq n \text{ e } i \neq j\}.$$

Será que também é possível demonstrar que o problema do par mais distante em dimensão 1 possui quota inferior $\Omega(n \log n)$? A resposta é não, pois existe algoritmo $O(n)$ que resolve o problema. Basta encontrar o maior e o menor valor da coleção, o que leva tempo $O(n)$ e retornar a diferença entre estes.

Problema do Par mais Distante em Duas Dimensões ($\mathbf{PMD^2}$)

Dada uma coleção de pontos no plano $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$*, sendo que* x_i *e* y_i *são valores algébricos para* $1 \leq i \leq n$*, determinar*

$$\max\left\{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} : 1 \leq i, j \leq n \text{ e } i \neq j\right\}.$$

E para o problema em dimensões maiores, será que também é possível encontrar algoritmo $O(n)$? Para responder essa pergunta, enunciaremos antes um novo problema, o da disjunção de conjuntos.

Problema da Disjunção de Conjuntos (DC)

Dados dois conjuntos de números algébricos A e B, decidir se $A \cap B = \emptyset$.

O problema da disjunção de conjuntos possui quota inferior $\Omega(n \log n)$ no modelo de árvore de decisões algébricas. Novamente, não demonstraremos esse fato aqui, mas essa demonstração pode ser encontrada em [5].

Agora sim estamos prontos para responder a questão. O problema do par mais distante para dimensões maiores que 1 possui quota inferior $\Omega(n \log n)$ pois é possível reduzir o problema da disjunção de conjuntos ao problema do par mais distante em duas dimensões.

Primeiramente, a função τ_I deve somar uma mesma constante positiva c a todos os elementos de A e de B de forma que os dois conjuntos sejam compostos apenas por elementos não negativos. Sejam A' e B' os conjuntos assim modificados. Agora, considere uma circunferência C de raio 1, e duas semi-retas tangentes a C em pontos diametralmente opostos e orientadas em sentidos opostos, conforme ilustrado na Figura 3.

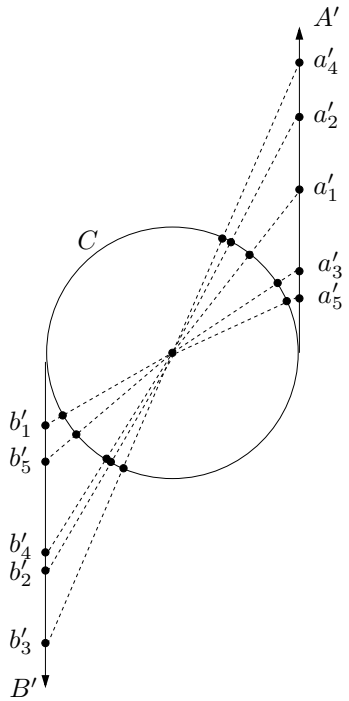

Figura 3: A função τ_I na redução de disjunção de conjuntos ao par mais distante em duas dimensões

Considere que os pontos de tangência representam o zero em cada uma das semi-retas e situe os valores de A' em uma das semi-retas e os valores de B'

na outra. O conjunto de pontos que formará a instância I_{PMD^2} do problema do par mais distante em duas dimensões é dado pelos pontos na intersecção da circunferência C com os segmentos de reta entre o centro de C e cada um dos pontos das retas tangentes, conforme indicado na Figura 3.

Observe que, para qualquer instância I_{PMD^2} do problema do par mais distante em duas dimensões construída dessa forma, a resposta do problema está no intervalo $(\sqrt{2}, 2]$, e será 2 se, e somente se, os conjuntos A e B não forem disjuntos. Então, τ_S simplesmente leva a resposta 2 de S_{PMD^2} na resposta não de S_{DC} e qualquer resposta menor que 2 na resposta sim de S_{DC}. Como $\tau_I \in O(n)$ e $\tau_S \in O(1)$, temos que $\Omega(n \log n)$ também é quota inferior para o problema do par mais distante em duas dimensões.

Exercício: Demonstre para esta redução que $\tau_S(A_{PMD^2}(\tau_I(I_{DC})))$ é de fato uma solução de I_{DC}.

Referências

[1] `http://www.cut-the-knot.com/do_you_know/numbers.html`.

[2] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 2001.

[4] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[5] E. M. Reingold. On the optimality of some set algorithms. *Journal of the ACM*, 19:649–659, 1972.

[6] P. J. Rezende and J. Stolfi. *Fundamentos de Geometria Computacional*. IX Escola de Computação, 1994.