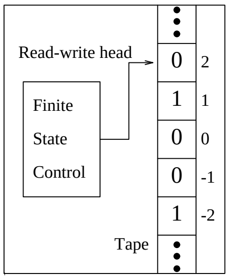
MAC 722 INTRODUÇÃO À TEORIA DA COMPLEXIDADE DE ALGORITMOS

1. Introdução



Por que alguns problemas parecem ser (computacionalmente) mais difíceis do que outros? Como podemos medir ou comprovar este diferente (e aparentemente intrínseco) grau de dificuldade?

Quais problemas podem ser resolvidos por um algoritmo? Quais não podem? Por quê?

Quais problemas podem ser resolvidos por um algoritmo eficiente? Quais não podem? Por quê?

Quanto mais poderoso deve ser um algoritmo para que façamos com que um problema incomputável passe a ser computável?

Em complexidade computacional estamos interessados em questões deste tipo.

Descrevemos a seguir algumas passagens históricas sobre complexidade computacional (principalmente sobre algoritmo polinomial como sinônimo de algoritmo eficiênte). Acreditamos que estas passagens deverão dar uma boa idéia de alguns tópicos que serão tratados por esta disciplina.

Uma das referências mais antigas conscientemente escrita sobre a análise de uma algoritmo é Lamé [9]. Neste artigo Lamé mostra que o algoritmo de Euclides para encontrar o máximo divisor comum é um algoritmo polinomial (polinomial no número de bits dos inteiros envolvidos).

Um (o mais?) antigo resultado mostrando a *intratabilidade* de um problema é o clássico resultado de Alan Turing [14]. Turing mostrou que certos problemas são *indecidíveis*, isto é alguns problemas não podem ser resolvidos através de algoritmos. Por exemplo, ele mostrou que é impossível encontrar um algoritmo que resolve o seguinte problema: *dados*: um programa de computador e uma entrada arbitrária para este programa; *pergunta:* este programa irá parar quando alimentado com a entrada dada?

A idéia de solubilidade-por-algoritmo-de-tempo-polinomial como um critério de complexidade para problemas aparece no trabalho de várias pessoas durante as décadas de 50 e 60.

Em 1965, Alan Cobham [1] em um artigo sobre máquinas de Turing e computabilidade escreveu o seguinte (o tamanho de um número n é denotado por $l(n)$):

> To obtain some idea as to how we might go about the further classification of relatively simple functions, we might take a look at how we ordinarily set about computing some of the more common of them.

> Suppose, for example, that m and n are two numbers given in decimal notation with one written above the other and their right ends aligned. Then to add m and n we start at the right and proceed digit-by-digit to the left writing down the sum. No matter how large m and n, this process terminates with the answer after a number of steps equal at most to one greater than the larger of $l(m)$ and $l(n)$. Thus the process of adding m and n can be carried out in a number of steps which is bounded by a linear polynomial in $l(m)$ and $l(n)$. Similarly, we can multiply m and n in a number of steps bounded by a quadratic polynomial in $l(m)$ and $l(n)$. So, too, the number of steps involved in the extraction of square roots, calculation of quotients, etc., can be bounded by polynomials in the lengths of the numbers involved, and this seems to be a property of simple functions in general. This suggests that we consider the class, which I will call $\mathcal{L}$, of all functions having this property.

No mesmo ano, Jack Edmonds [5] apresentou um algoritmo polinomial que encontra um emparelhamento máximo em um grafo (*maximum matching problem*):

> For practical purposes computational details are vital. However, my purpose is only to show as attractively as I can that there is an efficient algorithm. According to the dictionary, "efficient" means "adequate in operation or performance." This is roughly the meaning I want—in the sense that it is conceivable for maximum matching to have no efficient algorithm. Perhaps a better word is "good".
>
> I am claiming, as a mathematical result, the existence of a *good* algorithm for finding a maximum cardinality matching in a graph.
>
> There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether *or not* there exists an algorithm whose difficulty increases only algebraically with the size of the graph.
>
> The mathematical significance of this paper rests largely on the assumption that the two preceding senteces have mathematical meaning. I am not prepared to set up the machinery necessary to give them formal meaning, nor is the present context appropriate for doing this, but I should like to explain the idea a little further informally. It may be that since one is customarily concerned with existence, convergence, finiteness, and so forth, one is not inclined to take seriously the question of the existence of a *better-than-finite* algorithm ...
>
> One can find many classes of problems, besides maximum matching and its generalizations, which have algorithms of exponential order but seemingly none better. An example known to organic chemists is that of deciding whether two given graphs are isomorphic. For practical purposes the diference between algebraic and exponential order is often more crucial than the difference between finite and non-finite.

Edmonds [4] introduziu o termo 'boa caracterização', e deu uma boa caracterização para o número mínimo de conjuntos linearmente independentes nos quais o conjunto de colunas de uma matriz pode ser particionado.

> We seek a good characterization of the minimum number of independent sets into which the columns of a matrix of M_F can be partitioned. As the criterion of "good" for the characterization we apply the "principle of the absolute supervisor." The good characterization will describe certain information about the matrix which the supervisor can require his assistant to search out along with a minimum partition and which the supervisor can then use with ease to verify with mathematical certainty that the partition is indeed minimum. Having a good characterization does not mean necessarily that there is a good algorithm. The assistant might have to kill himself with work to find the information and the partition.

Stephen Cook [2] mostrou que dentro da chamada classe NP de problemas existem problemas que são tão difíceis quanto qualquer outro problema na classe. Estes são os chamados problemas NP-completos. Logo após, Richard Karp [8] apresentou uma coleção de resultados mostrando que existe na classe NP um grande número de problemas que são NP-completos (estes incluiam problemas de decisão correspondentes a conhecidos problemas em otimização combinatória—inclusive o Problema do Caixeiro Viajante). É crença geral de que os problemas NP-completos são intratáveis no sentido de que não existe algoritmo polinomial para resolvê-los, apesar de que isto não pôde ser provado até agora (na realidade este é o maior problema aberto em teoria da computação).

Nesta disciplina deveremos estudar várias classes de problemas.

[O que foi brevemente tratado nesta introdução foi extraído do Capítulo 1 de [6], Capítulo 1 de [12] e do Capítulo 2 de [13].]

2. Objetivos da disciplina

A compreensão das limitações do modelo computacional e da dificuldade inerente de se resolver problemas computacionais algoritmicamente é fundamental para o desenvolvimento de pesquisa em várias áreas da Ciência da Computação. Nossos objetivos nesta disciplina são:

- Definir classes de problemas computacionais baseado na quantidade de recursos (tempo e espaço) necessários para resolvê-los e investigar a relação entre estas classes.
- Entender e ser capaz de usar técnicas comuns de provas em complexidade computacional, incluindo simulação, diagonalização e reduções.
- Ser capaz de discutir resultados em complexidade computacional e descrever a sua significância.
- Adquirir alguma base para discutir tópicos em complexidade computacional.
- Estudar comparativamente os principais modelos de computação, particularmente nos aspectos de complexidade de tempo e espaço de problemas computacionais.

3. Tópicos que pretendemos cobrir

Nesta disciplina abordaremos os seguintes tópicos (não necessariamente nesta ordem):

1. *Máquinas de Turing e elementos de computabilidade.* Máquinas de Turing deterministicas. A tese de Church-Turing. Problemas de decisão, busca e otimização. Indecidibilidade. Problema da parada. Enumerabilidade recursiva.
2. *Tempo polinomial.* A classe de complexidade P. Linear Speed-up Theorem. Reduções. Algoritmos polinomiais. Diagonalização.
3. *NP e NP-completude.* Máquina de Turing não deterministica. A classe NP. NP-completude. O Teorema de Cook-Levin. Transformações polinomiais. Algoritmos pseudo-polinomiais. NP-completude forte. Oracle Turing Machine. Redução de Turing.
4. *Complexidade de Espaço.* Classes deterministicas de complexidade de espaço. Linear Space Compression Theorem. PSPACE, NPSPACE, PSPACE-completude.
5. *Outros tópicos.* Dependendo do interesse dos alunos podemos estudar alguns dos seguintes tópicos: otimização e aproximação; primalidade e criptografia; algoritmos probabilísticos; modelos de complexidade para computação paralela, etc.

4. Atividades e critério de avaliação

As atividades que pretendemos ter nesta disciplina para avaliar os alunos são as seguintes:

Listas de exercícios: Durante o andamento da disciplina serão dadas várias listas de exercícios. A nota em listas de exercícios será responsável por 30% da nota final.

Provas: Dependendo do número de alunos na disciplina podemos ter de duas a três provas. A média aritmética das notas das provas representará 60% da nota final.

Seminários: Dependendo do número de alunos na disciplina podemos ter seminários dos alunos. O seminário valerá 10% da nota final.

Observação. Nota final entre 8.6 e 10 corresponderá ao conceito A, entre 7.0 e 8.5 corresponderá ao conceito B, entre 5.0 e 6.9 corresponderá ao conceito C, ...

5. Bibliografia

Para preparar as aulas desta disciplina consultaremos principalmente o livro de Papadimitriou [12]. Outros livros que poderemos consultar são Lewis & Papadimitriou [10], Hopcroft & Ullman [7], Garey & Johnson [6] além da notas de aula de Lovász [11].

O livro de Papadimitriou [12] contém todos os tópicos que cobriremos nesta disciplina, podemos dizer que ele é o livro texto. O livro de Garey

& Johnson [6] é um texto clássico em complexidade computacional e NP-completude (foi o primeiro livro sobre o assunto).

Cormen, Leiserson & Rivest [3] é um livro enciclopédico sobre análise de algoritmos que trata de NP-completude no seu Capítulo 36.

Na biblioteca também podem ser encontrados alguns surveys sobre complexidade computacional, veja, por exemplo, Van Emde Boas [15].

Artigos em complexidade computacional podem ser encontrados em várias revistas, incluindo *Journal of Algorithms*, *Journal of the ACM*, *SIAM Journal on Computing*, e *Journal of Complexity Theory*.

Existe uma conferência anual em complexidade computacional a *IEEE Conference on Computational Complexity* (antiga: *Structure in Complexity Theory Conference*). Os proceedings podem ser encontrados na biblioteca, veja QA810.C C748). Outras conferências que também apresentam trabalhos em complexidade computacional são *STOC* (QA800.C S989) e *FOCS* (QA800.C S989).

6. Complexidade computacional na Internet

Abaixo encontra-se a lista de alguns sítios de complexidade computacional:

- *A compendium of NP optimization problems.* Um catálogo de problemas de otimização junto com a classe de complexidade a que pertencem (se conhecido) pode ser encontrado no URL
  `http://www.nada.kth.se/~viggo/problemlist/compendium.html`
- *Special Interest Areas.* Um sítio que contém vários links para outros sítios de geometria computacional e complexidade computacional esta em
  `http://robotics.stanford.edu/~suresh/theory/areas.html`
- *Arthur-Merlin site.* Se você gosta de estórias em quadrinhos e está interessado em uma explicação simples de IP e PCP de uma olhada no sítio
  `http://www.cs.uchicago.edu/groups/theory/merlin/`
- *Theoretical computer science on the web.* Contém links para páginas e artigos de interesse geral da comunidade de teoria
  `http://robotics.stanford.edu/~suresh/theory/`

7. Outras informações

A sala do José Auguto é a A-302, o número de telefone é 818-6235 e o email `jose@ime.usp.br`. A sala do Coelho é a B-164, o número do telefone é 818-6295 e o endereço eletrônico é `coelho@ime.usp.br`.

Pretendemos fazer uma lista contendo os endereços eletrônicos de todos os alunos desta disciplina. Assim, por favor, se você pretende cursar esta disciplina mande-nos um e-mail. Também sinta-se a vontade para nos escrever e fazer perguntas ou comentários sobre a disciplina. Manteremos uma página de MAC 722 no URL `http://www.ime.usp.br/~coelho/mac722/`. Nessa página também colocaremos o material da disciplina (como, por exemplo, listas de exercícios). Por favor, dê uma olhada nesta página regularmente.

Referências

1. A. Cobham, *The intrinsic computational difficulty of functions*, Logic Methodology and Phylosophy of Science (Y. Bar-Hillel, ed.), Proc. Intern. Congress 1964, North-Holland, 1965, pp. 24–30.
2. S.A. Cook, *The complexity of theorem-proving procedures*, Proceedings of Third Annual ACM Symposium on Theory of Computing (New York), ACM, 1971, (Shaker Heights, Ohio, 1971), pp. 151–158.
3. T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to algorithms*, The MIT Electrical Engineering and Computer Scienece Series, The MIT Press, MacGraw-Hill Book Company, 1990, QA758 C811i.
4. J. Edmonds, *Minimum partition of a matroid into independent subsets*, Journal of Research of the National Bureau of Standars (B) **69** (1965), 67–72.
5. ———, *Paths, trees, and flowers*, Canadian Journal of Mathematics **17** (1965), 449–467.
6. M.R. Garey and D.S. Johnson, *Computers and intractability: A guide to the theory of np-completeness*, Freeman, San Francisco, 1979.
7. J.E. Hopcroft and J.D. Ullman, *Introduction to automata theory languages, and computation*, Addison-Wesley, Reading, Mass., 1979.
8. R.M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations (New York) (R.E. Miller and J.W. Thatcher, eds.), Plenum Press, 1972, pp. 85–103.
9. G. Lamé, *Note sur la limite du nombre des divisions dans la recherche du plus grand commun diviseur entre deux nombres entiers*, Compte[s] Redu[s Hebdomadaires] des Séances de l'Académie des Sciences (Paris) (1844), 867–870.
10. H.R. Lewis and C.H. Papadimitriou, *Elements of the theory of computation*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
11. László Lovász, *Computation complexity*, Lecture Notes, March 1994, (translation, additions, modifications by Péter Gács.
12. C.H. Papadimitriou, *Computational complexity*, Addison Wesley, Reading, Mass., 1995.
13. A. Schrijver, *Theory of linear and integer programming*, John Wiley & Sons, Chichester, 1986.
14. A. Turing, *On computable numbers, with aplications to the entscheidungsproblem*, Proceedings of the London Mathematical Society **42** (1936–7), no. 2, 230–265, correction: **43** (1937) 544–546.
15. P. vam Emde Boas, *Machine models and simulations*, The Handbook of Theoretical Computer Science, vol I: Algorithms and Complexity (J. van Leeuwen, ed.), MIT Press, Cambridge, Mass., 1990, pp. 1–61.