INF 02779 Análise de Algoritmos - 2002/2 01/03/03

Aula 14: Classe NPC

Instrutor: Berilhes Borges Garcia Escriba: Kellen

DRAFT

Muitos problemas citados anteriormente têm sido estudados por décadas. Todos têm aplicações reais e práticas. Contudo, nenhum deles tem até hoje um algoritmo de tempo polinomial, algoritmos exponenciais são o que de "melhor" existem para eles. Todos estes problemas possuem a propriedade comum que um certificado (ou prova) curto existe, de modo que você pode eficientemente verificar (em tempo polinomial) se uma resposta Sim é correta. Note que este pedaço especial de informação pode ser difícil de encontrar.
Example:
· Versão de decisão do TSP: o próprio tour;
· Conjunto Independente, Clique: o conjunto U;
· Satisfabilidade: uma atribuição que faz uma fórmula θ verdade. ⊠

Definição 1. *(Definição Classe NP) A classe NP consiste de todos os problemas onde:*
Você pode verificar as respostas Sim eficientemente (em tempo polinomial) dado um certificado curto (de tamanho polinomial).
Nenhum certificado pode enganar seu verificador de tempo polinomial, de modo a que este diga Sim para uma instância Não.

Definição 2. *(Uma definição mais precisa de NP) Um problema de decisão está em NP se e somente se existe um procedimento de tempo polinomial verificar (...), e um inteiro K tal que:*
· Para toda a entrada x que é uma instância Sim de um determinado problema existe um certificado h com $|h| \leq |x|^h$ tal que verificar (x,h) = Sim
e
· Para toda entrada x para um problema que é uma instância Não, não existe um certificado h com $|h| \leq |x|^h$ tal que verificar (x,h) = Sim.

Example: Clique está em NP

```
Procedimento verificar (x,y)
    if x é uma representação bem formada de um grafo G = (V,E) e um inteiro K,
        e
        h é uma representação bem formada de um subconjunto de vértices U de V de tamanho K,
        e
        U é um clique em G,
    then a resposta é "Sim"
    else a resposta é "Eu não estou convencido"
```

$\boxtimes$

Resolvendo Problemas em NP sem certificados
O único algoritmo óbvio para muitos destes problemas é a força bruta: Tente todos os certificados possíveis e cheque cada um deles para determinar se algum certificado funciona. Isto leva um tempo exponencial.
Example:
· 2^n atribuições verdade para variáveis;
· $n!$ possíveis tours para um problema TSP de n vértices;
·$\binom{n}{k}$possíveis subconjuntos de k elementos de n vértices;
· (etc). $\boxtimes$

O que nós sabemos: Ninguém sabe se todos os problemas em NP podem ser resolvidos em tempo polinomial, isto é, se P=NP?
· esta é uma das questões em aberto mais importante de toda a ciência;
· Enormes implicações práticas.
Note que todos os problemas em P também estão em NP, você nem mesmo necessita um certificado para os problemas em P, dessa forma você pode simplesmente ignorar qualquer certificado que lhe seja fornecido.
Em segundo lugar note que todos os problemas em NP também pertence a classe dos problemas que podem ser resolvidos em tempo exponencial.
Example: Alguns exemplos de problemas em NP
· Satisfabilidade
· Conjunto Independente
· Clique
· Cobertura por Vértice
Todos estes problemas são difíceis de resolver, em segundo lugar certificados são úteis em todos estes problemas. Todos eles compartilham um fato surpreendente: Um algoritmo eficiente para um deles fornecerá também um algoritmo eficiente para todos eles! $\boxtimes$

Alguns problemas em NP parecem ser verdadeiramente difíceis, afinal pessoas têm procurado por algoritmos eficientes para eles por séculos sem ter obtido sucesso. No entanto, ninguém sabe como provar que eles são realmente difíceis.
A abordagem alternativa consiste em mostrar que eles são pelo menos tão difíceis quanto qualquer outro problema em NP

Definição 3. *(NP-Completo) Uma definição grosseira consiste em dizer que um problema é NP-hard se e somente se este é pelo menos tão difícil quanto qualquer outro problema em NP.*
Um problema é NP completo se e somente se este é:
· NP - difícil (NP-hard)
· NP.

Como nós mostramos que um problema é "pelo menos tão difícil quanto" outro?

Nós reduzimos o problema L a um problema R, nós representamos isto por $L \leq R$. Reduzir o problema L a um problema R consiste em transformar umai nstância de L em uma instância de R tal que a resposta de L é exatamente a resposta de R.

Função L (x)

· Execute um programa T para transladar uma entrada x para L em uma entrada y para R.

· Chame uma subrotina para o problema R com a entrada y.

· A saída é a resposta produzida por R (y)

Nós devemos impor algumas restrições sobre o tipo de translação aceitável. Note que todos os problemas na classe NP são decidíveis, em tempo exponencial no pior caso para alguns problemas. Uma redução inaceitável T poderia simplesmente resolver o problema L diretamente e a seguir criar alguma entrada estúpida para R para o qual nós já sabemos a resposta. T estaria fazendo todo o trabalho. Portanto, nós não obteríamos nenhuma informação adicional acerca da relação que existe entre o problema L e o problema R.

Definição 4. *(Redução de Tempo Polinomial (Redução de Karp)) $L \leq^p R$. se existe um algoritmo de tempo polinomial para L assumindo a existência de uma subrotina de tempo polinomial para R.*

Dessa forma um algoritmo eficiente para R implica em um algoritmo eficiente para L. Se nós provarmos que não existe nenhum algoritmo eficiente para L, então isto prova que não existe nenhum algoritmo eficiente para R.

Por que o nome redução?

Porque este reduz um problema mais fácil a um problema mais difícil! Da mesma forma nós podemos dizer que Maxvell reduziu o problema de analisar eletricidade e magnetismo à resolução de equações diferenciais parciais. Note que resolver equações diferenciais parciais de uma forma gera é um problema muito mais difícil que resolver problemas de circuitos elétricos e magnéticos.