Prova que o problema da PARTITION é NP-completo utilizando uma redução de SUBSET-SUM

Marcos Straub do Nascimento

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

marcos.straub@inf.ufrgs.br

Resumo. *O artigo apresenta uma prova de que o problema da partição (PARTITION) é um problema da classe NP-completo. Essa prova é feita utilizando uma redução do problema SUBSET-SUM para o problema PARTITION.*

1. Introdução

Este artigo apresenta uma prova de que o problema PARTITION pertence a clase dos problemas NP-completo. Para isso, será mostrado uma algoritmo de verificação em tempo polinomial para provar que ele pertence a classe NP. Em seguida, será utilizado o método da redução, fazendo uma redução do problema SUBSET-SUM para o problema PARTITON.

Primeiramente, será apresentado o problema. Em seguida, teremos a prova que o problema pertence a classe NP e finalizando com a prova que ele é NP-completo utilizando a redução de SUBSET-SUM.

2. O Problema da Partição de Conjuntos (PARTITION)

O problema consiste em decidir se, dado um conjunto de números inteiros S, é possível dividir S em dois subconjuntos s1 e s2 de forma que a soma dos elementos em s1 seja igual a soma dos de s2.

Na figura 1, o conjunto S é formado pela união disjunta de s1 e s2 e o somatorio de s1 equivale ao o de s2.

```
S := {2,3,5,7,13}
s1 := {3,5,7}
s2 := {2,13}
somatorio(s1) = somatorio(s2) = 15
```

Figura 1. Exemplo de partição de conjuntos

3. PARTITION pertence à NP

É bastante simples perceber que o problema pertence a classe NP.

Para a prova, precisamos de um algorimo de verificação em tempo polinomial que, dado uma entrada com uma possível solução, o algoritmo deve decidir se essa solução é válida.

O programa consiste em somar os valores de s1, os valores de s2, e fazer a comparação entre eles. Se o valores forem iguais, a solução é válida e o algoritmo retornará *true*. Do contrário, o solução é rejeitada retornando *false*. Além disso, ele deverá verificar se s1 e s2 são subconjuntos de S, sendo que a união deles deverá formar S.

No algoritmo abaixo, não foi descrita a função "uniao". No entanto, ela pode consistir de uma simples atualização de descritores para listas encadeadas tendo uma complexidade pessimista O(1). Contudo, a função de verificação de igualdade de conjuntos deverá ter que varrer todo o conjunto "uniao" e marcando em S. Esse processo terá complexidade pessimista O(n). Além disso, existem dois comandos de laço percorrendos os subconjuntos, totalizando também uma complexidade O(n).

Portanto, o algoritmo apresentado tem uma complexidade pessimista O(n), provando que o problema pertence a classe NP, pois o algoritmo de verificação é polinomial.

```
verificapartition(S,s1,s2){
// Verifica se s1Us2 = S
if( uniao(s1,s2) != S)
     false
//Verifica se somatorio(s1) == somatorio(s2)
foreach i1 in s1:
     somas1= somas1 +i1;
foreach i2 in s2:
     somas2= somas2 +i2;
if(somas1 == somas2)
     return true;
else
     return false;
}
```

Figura 2. Algoritmo de Verificação

4. PARTITION pertence a NP-completo

Para provar que o problema PARTITION pertence a classe NP-completo, será utilizado o problema SUBSET-SUM, pois é um problema NP-completo conhecido. Se for possível encontrar um algoritmo polinomial que reduza o problema SUBSET-SUM para PARTITION, o problema será considerado NP-completo.

O problema SUBSET-SUM consite em determinar para um conjunto de inteiros S e um número t, se existe um subconjunto $s \subseteq S$ ao qual a soma dos seus elementos é igual a t?

Agora é necessário encontrar um algoritmo em tempo polinomial que transforme instâncias de SUBSET-SUM em instâncias de PARTITION. Assim, se existir uma solução para PARTITION, haverá também para SUBSET-SUM.

Então, dado uma instância de SUBSET-SUM I=(S,t), precisamos encontrar uma instância de PARTITION I’, tal que, se o problema I’ for resolvido, I também será.

Portanto, sendo I=(S,t) uma instância de SUBSET-SUM, onde S é um conjunto de valores {v1,v2,v3,...,vn} e t é um valor arbitrário que deverá equivaler ao somatório do subconjunto de S, o mapeamento para I’, será tomado como o seguinte:

I'=(S'), onde

S'={v1,v2,v3,...,vn,x} e

x=v1 + v2 + … + vn − 2t

Em suma, foi adicionado ao conjunto S um elemento x que contém o somatório total de S subtraindo 2t, formando assim o S’. Nesse caso, quando S’ for particionado em dois conjuntos com somatórios iguais, esses irão totalizar:

(v1 + v2 +…+ vn + x)/2 = (2v1 + 2v2 +…+ 2vn - 2t)/2 = (v1 + v2 +...+ vn - t)

Considerando que uma das duas partições (s1) deverá conter o elemento x, o resto de seus elementos da partição deverá ter como somatório (v1 + v2 +...+ vn - t)-x, ou seja:

s1={x,alguma_coisa}, onde o somatório de alguma_coisa será (v1 + v2 +...+ vn - t) - x

Desse modo, o somatório de s1 irá corresponder com a metade do somatório de S’, citado anteriormente.

Sendo assim, alguma_coisa tem como somatório:

(v1 + v2 +...+ vn - t) - x =

(v1 + v2 +...+ vn - t) - (v1 + v2 + … + vn − 2t) = t

Portanto, existe um subconjunto de S’ que tem como somatório t. Além disso, ambas as partições tem como somatório (v1 + v2 +...+ vn - t), totalizando metade do somatório de S. Esse mapeamento garante que, quando conseguirmos fazer a partição de conjuntos, será obtido obrigatoriamente um subconjunto de S que irá ter como somatório t. Na figura 3 segue um diagrama que resume de forma mais clara o processo de redução. Já a figura 4 apresenta um algoritmo de mapeamento que efetua a redução em tempo polinomial.

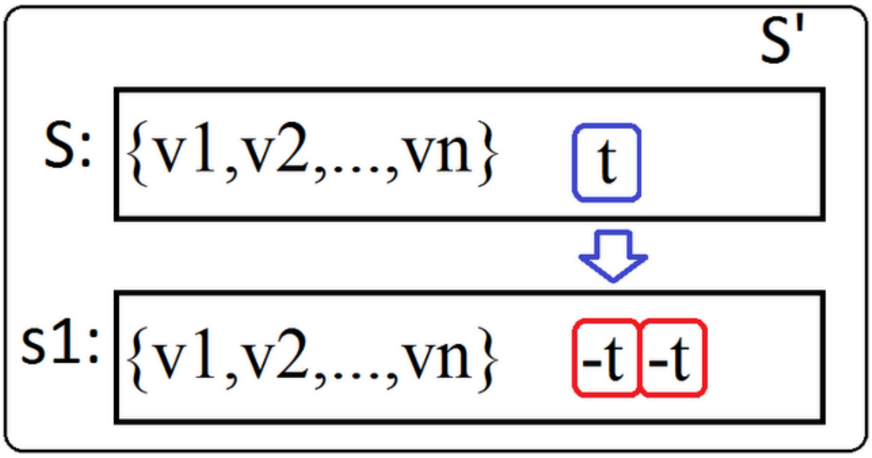

Figura 3. Processo de Redução

```
subsetsum2partition(S,t){
    foreach i in S:
        somatorio = somatorio + i;

    x = somatorio - 2*t;
    S' = append(S,x);
    return S';
}
```

Figura 4. Algoritmo de mapeamento

Na figura 4, percorremos o conjunto S para obter o somatório de seus elementos. Essa laço irá ter como complexidade pessimista O(n). Após, será feita a subtração com 2t para se obter o x e concatená-lo com o conjunto S para obter S'. Essas duas operações terão complexidade pessimista O(1) (considerando S uma estrutura de lista encadeada com descritores para o final da lista).

Como o algoritmo tem complexidade O(n), essa redução permite provar que o problema da partição de conjuntos (PARTITON) é NP-completo.

5. Conclusão

Conforme apresentado, podemos concluir que o problema PARTITION pertence a classe NP-completo, pois foi possível obter um algoritmo de verificação em tempo polinomial e o problema NP-completo SUBSET-SUM foi possível ser reduzido ao problema PARTITION.

Referências

http://valis.cs.uiuc.edu/~sariel/teach/2004/b/webpage/lec/10_npc_notes.pdf

Toscani, Laira Viera(2001), Complexidade de Algoritmos

http://umamao.com/questions/cormen-et-al-2-edicao-capitulo-34-np-completude-ex-34-5-5/answers/4d363cc279de4f73d10001c1

http://www.cs.berkeley.edu/~daw/teaching/cs170-s03/Notes/lecture23.pdf

http://www.ecst.csuchico.edu/~amk/foo/csci356/notes/ch11/NP5.html

http://www.cs.mun.ca/~kol/courses/6743-f07/lec10.pdf