Backtracking

Problema da Partição
Polícia e Ladrão

Problema da Partição

O Problema da Partição é a tarefa de decidir quando um conjunto *S* de *n* números inteiros positivos a_1, a_2, a_3, …, a_n pode ser particionado em dois subconjuntos S_1 e S_2 de tal forma que a soma dos números em S_1 seja igual à soma dos números em S_2.

$$\sum_{i \in S_1} a_i = \sum_{j \in S_2} a_j$$

Exemplo

Dado o conjunto $S = \{3, 1, 1, 2, 2, 1\}$, temos como possíveis soluções válidas os subconjuntos:

- $S_1 = \{1, 1, 1, 2\}$ e $S_2 = \{2, 3\}$
- $S_1 = \{3, 1, 1\}$ e $S_2 = \{2, 2, 1\}$

Perceba, no entanto, que nem todo conjunto possui uma partição. Por exemplo, $S = \{2, 5\}$.

Algoritmo por Backtracking

- Gerar todos os 2^n subconjuntos do conjunto de entrada *S*, tal que *n* é o número de elementos em *S.*

Algoritmo por Backtracking

- Verificar se existe alguma partição de *S* tal que a soma dos elementos do subconjunto gerado e a soma dos elementos não selecionados de *S* sejam exatamente iguais.

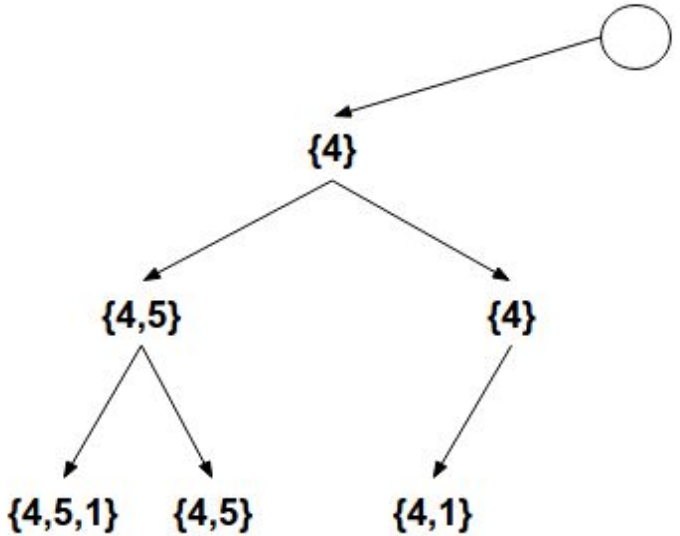
  - Caso verdadeiro, aceitar a partição como solução e parar o processamento.
  - Caso falso, prosseguir para a próxima partição do conjunto.

Algoritmo por Backtracking

- Se nenhuma partição do conjunto de entrada *S* satisfaz a declaração do problema, então o conjunto não possui solução.

Exemplo

Dado o conjunto *S = {4, 5, 1}*, a execução do algoritmo resulta na seguinte árvore de chamadas recursivas:



Código em C++

```
void backtracking(vector<int> S, int k, int partition_sum, int sum, bool found_solution){
    if(k == int(S.size())){
        if(partition_sum  == sum){
            found_solution = true;
            return;
        }
    }
    else{
        partition_sum += S[k];
        sum -= S[k];
        backtracking(S, k+1, partition_sum, sum, found_solution);
        if(found_solution) return;
        sum += S[k];
        partition_sum -= S[k];
        backtracking(S, k+1, partition_sum, sum, found_solution);
        if(found_solution) return;
    }
}
```

Algoritmo por Backtracking com Poda

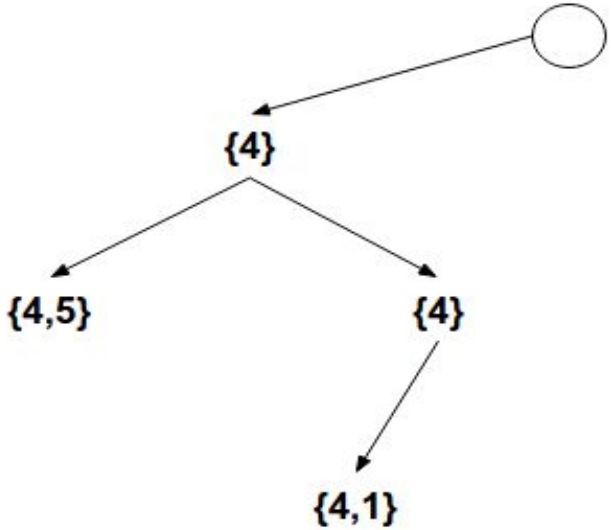
- Gerar apenas metade dos 2^n subconjuntos possíveis do conjunto de entrada *S*, tal que *n* é o número de elementos em *S*.

Algoritmo por Backtracking com Poda

- Se a inclusão de um elemento num subconjunto que está sendo gerado faz com que a soma desses elementos seja maior do que a soma dos elementos ainda não selecionados de *S*, a busca é interrompida e os próximos subconjuntos gerados a partir daquele ponto não irão considerar este elemento.

Exemplo

Dado o conjunto *S = {4, 5, 1}*, a execução do algoritmo otimizado resulta na seguinte árvore de chamadas recursivas:



Polícia e Ladrão

https://www.urionlinejudge.com.br/judge/pt/problems/view/1905