**5. Problema da Mochila (Branch and Bound)**

O objetivo é encontrar o valor máximo que pode ser colocado na mochila sem exceder sua capacidade. Usaremos uma árvore de decisão para explorar combinações de itens.

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

**Casos de Teste**

1. Entrada:
   1. Itens: {(2, 40), (3, 50), (4, 100), (5, 95)}
   2. Capacidade: 7
   3. Saída Esperada: 135
2. Entrada:
   1. Itens: {(1, 10), (2, 20), (3, 30)}
   2. Capacidade: 5
   3. Saída Esperada: 50

**Análise de Complexidade**

Problema da Mochila:

* Tempo: no pior caso, mas com poda significativa para instâncias práticas.
* Espaço: onde *n* é o número de itens.

**Operação Básica**

1. Ordenação dos Itens por Razão Valor/Peso

* Operação: A ordenação é realizada logo no início do algoritmo para preparar a lista de itens de acordo com a razão valor/peso.
* Complexidade: O (n log n), onde n é o número de itens.

2. Cálculo do Bound (Função calculateBound)

* Operação: Para cada nó da árvore de busca, é calculado um limite superior (bound) que estimará o lucro máximo que pode ser obtido a partir daquele ponto, ajudando a decidir se o nó deve ser explorado ou podado.
* Complexidade: O(n) por nó, já que a função percorre os itens após o nível atual.

3. Expansão dos Nós

* Operação: Para cada nó, o algoritmo gera dois nós filhos, representando as opções de incluir ou não o próximo item na mochila. Esses nós são explorados recursivamente na árvore de busca.
* Complexidade: O número de nós gerados pode ser exponencial, ou seja, até O(2n) no pior caso, mas é otimizado por meio da poda (usando o bound).

4. Atualização do Máximo Lucro (Variável maxProfit)

* Operação: Quando um nó válido é explorado (i.e., a solução parcial tem um lucro maior que o lucro máximo encontrado até o momento), o valor de maxProfit é atualizado.
* Complexidade: O (1) para cada nó, uma simples comparação.

**Expressão de Custo Completa:**

* Passo 1: Ordenação dos itens O (n log n).
* Passo 2: Expansão dos nós e cálculo do bound:
* O número de nós pode crescer exponencialmente no pior caso (O(2n)).
* Para cada nó, o cálculo do bound tem complexidade O(n).
* A atualização do lucro máximo é O (1).

Portanto, a complexidade total do algoritmo no pior caso é a soma do custo de ordenação e o custo de explorar todos os nós possíveis.

O (n log n +2n ⋅ n)

Onde:

* O (n log n) é o custo da ordenação dos itens.
* O (2n ⋅ n) é o custo de explorar todos os nós possíveis (com 2n nós no pior caso e cada nó exigindo O(n) operações para calcular o bound).

**Classe de Problemas**

O Problema da Mochila com a técnica de Branch and Bound pertence à classe de problemas NP-difíceis.

**Justificativa:**

1. **Problema de otimização**: O objetivo do problema é encontrar a solução ótima que maximiza o valor da mochila sem exceder a capacidade máxima, considerando uma série de itens com peso e valor específicos. A solução envolve otimizar uma função (maximizar o lucro) sob restrições (capacidade limitada da mochila).
2. **Complexidade**:

* O problema da mochila 0/1, no qual se decide se inclui ou não cada item, é um clássico exemplo de problema NP-difícil. Isto significa que não se conhece um algoritmo eficiente (polinomial) que resolva todos os casos do problema em tempo razoável.
* A técnica de Branch and Bound é uma forma de explorar a solução ótima de maneira mais eficiente, embora não altere a complexidade fundamental do problema. Ela permite cortar ramos da árvore de busca que não podem levar à solução ótima, mas no pior caso ainda há uma exploração exponencial (custo O(2n)).

Características:

* Classe de Problemas: NP-difícil (NP-hard).
* Tipo de problema: Problema de otimização combinatória.

Embora o Problema da Mochila com Branch and Bound seja uma técnica para reduzir a complexidade prática do algoritmo, ele ainda pertence à classe NP-difícil, pois mesmo com técnicas de poda como Branch and Bound, a complexidade no pior caso ainda cresce exponencialmente com o número de itens.