Seminário 2 – Projeto e Análise de

Algoritmos

Tema: Problema da Mochila (Knapsack Problem)

Composição da Equipe:

- Bruno Trindade

- Magda Tainy Nunes Amaral

Escolha do Problema:

Problema da Mochila (Knapsack Problem) – Classe NP-Completo.

Descrição: Dado um conjunto de itens, cada um com um valor e um peso, determinar quais itens devem ser colocados na mochila para maximizar o valor total sem exceder a capacidade máxima.

Técnica de Construção do Algoritmo:

1. Programação Dinâmica (PD);

2. Heurística Gulosa;

3. Comparação experimental – análise de desempenho entre ambas as abordagens.

Algoritmo Implementado (Python):

```
def knapsack_dp(values, weights, capacity):
 n = len(values)
 dp = [[0] * (capacity + 1) for _ in range(n + 1)]
 for i in range(1, n + 1):
 for w in range(1, capacity + 1):
 if weights[i - 1] <= w:
 dp[i][w] = max(dp[i - 1][w], values[i - 1] + dp[i - 1][w - weights[i - 1]])  else:
 dp[i][w] = dp[i - 1][w]
 return dp[n][capacity]

def knapsack_greedy(values, weights, capacity):
 ratio = [(v / w, v, w) for v, w in zip(values, weights)]
 ratio.sort(reverse=True)
 total_value = total_weight = 0
 for r, v, w in ratio:
 if total_weight + w <= capacity:
 total_weight += w
 total_value += v
 return total_value

values = [60, 100, 120, 90, 30]
weights = [10, 20, 30, 25, 15]
capacity = 50

print("Solução ótima (PD):", knapsack_dp(values, weights, capacity))
print("Solução gulosa:", knapsack_greedy(values, weights, capacity))
```

Dataset Simples:

Valores: [60, 100, 120, 90, 30]
Pesos: [10, 20, 30, 25, 15]
Capacidade: 50

Conclusão:

O Problema da Mochila 0/1 é NP-Completo. A Programação Dinâmica encontra a solução ótima, enquanto a abordagem Gulosa fornece uma aproximação rápida e eficiente para grandes instâncias. Ambas ilustram bem o equilíbrio entre custo computacional e precisão em problemas NP.