PRACTICAL 5

Topic: A\* star algorithm for solving 0/1 knapsack problem

Introduction:

The **A\* (A-star) algorithm** is commonly used for pathfinding and graph traversal, but it can also be adapted to solve optimization problems like the **0/1 Knapsack Problem**. Here's how it works in the context of the knapsack problem:

**Problem Description**

* **Input**:
  + nnn: Number of items.
  + wiw\_iwi​: Weight of item iii.
  + viv\_ivi​: Value of item iii.
  + WWW: Maximum capacity of the knapsack.
* **Output**:
  + Subset of items maximizing the total value without exceeding the knapsack capacity.

Code:

import heapq

class KnapsackAStar:

    def \_\_init\_\_(self, weights, values, capacity):

        self.weights = weights  # List of weights

        self.values = values    # List of values

        self.capacity = capacity

        self.n = len(weights)   # Number of items

    def heuristic(self, idx, current\_weight, current\_value):

        """Calculate the heuristic value."""

        remaining\_capacity = self.capacity - current\_weight

        remaining\_value = 0

        for i in range(idx, self.n):

            if self.weights[i] <= remaining\_capacity:

                remaining\_value += self.values[i]

                remaining\_capacity -= self.weights[i]

        return current\_value + remaining\_value

    def a\_star(self):

        """Solve the knapsack problem using A\*."""

        # Priority queue: (priority, current\_value, current\_weight, index, items\_included)

        pq = []

        heapq.heappush(pq, (-0, 0, 0, 0, []))  # Start with index 0

        best\_value = 0

        best\_items = []

        while pq:

            \_, current\_value, current\_weight, idx, items\_included = heapq.heappop(pq)

            # Check if we've reached the end of the items

            if idx == self.n:

                if current\_value > best\_value:

                    best\_value = current\_value

                    best\_items = items\_included

                continue

            # Exclude current item

            heapq.heappush(pq, (-self.heuristic(idx + 1, current\_weight, current\_value),

                                current\_value, current\_weight, idx + 1, items\_included))

            # Include current item (if it fits)

            if current\_weight + self.weights[idx] <= self.capacity:

                heapq.heappush(pq, (-self.heuristic(idx + 1,

                                                     current\_weight + self.weights[idx],

                                                     current\_value + self.values[idx]),

                                    current\_value + self.values[idx],

                                    current\_weight + self.weights[idx],

                                    idx + 1,

                                    items\_included + [idx]))

        return best\_value, best\_items

# Example

weights = [2, 3, 4, 5]

values = [3, 4, 5, 6]

capacity = 5

solver = KnapsackAStar(weights, values, capacity)

best\_value, best\_items = solver.a\_star()

print("Maximum Value:", best\_value)

print("Items Included:", best\_items)

Output:

