Kanishk Singh- SE21UCAM005 (CAM)

Lab 5- Greedy Algorithm Implementation

Functions used:-

1. One function to generate random problems for Knapsack

```
def generate_random_knapsack_problem(n):
    weights = []
    values = []

    for _ in range(n):
        weight = random.randint(1, 200)
        value = random.randint(1, 500)
        weights.append(weight)
        values.append(value)

    capacity = random.randint(1, 1000)

    return weights, values, capacity
```

2. Then we have the Knapsack solver using greedy approach

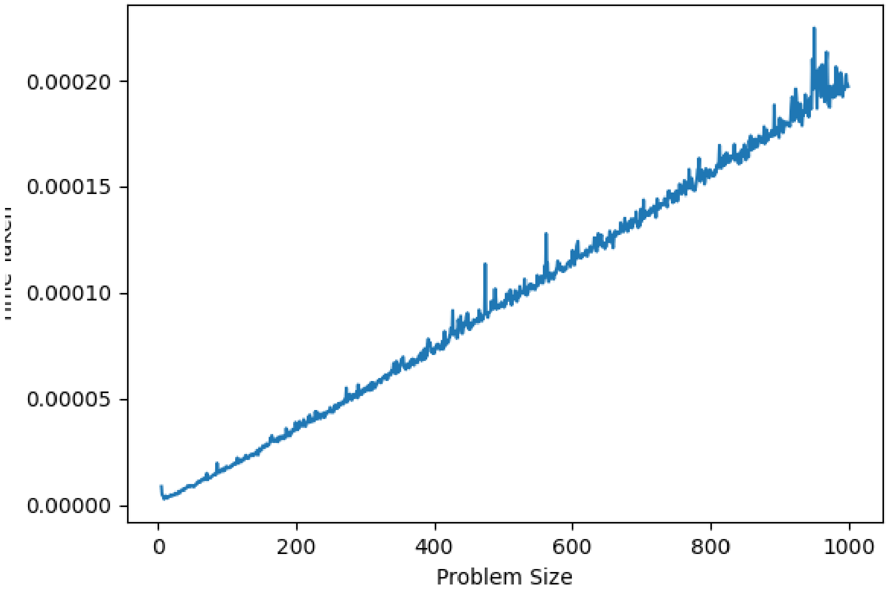
```
def knapsack_greedy(weights, values, capacity):
    n = len(weights)
    value_per_weight = [(values[i] / weights[i], weights[i], values[i]) for i in range(n)]
    value_per_weight.sort(reverse=True)

    max_value = 0
    knapsack = []

    for _, weight, value in value_per_weight:
        if capacity == 0:
            break
        if weight <= capacity:
            max_value += value
            knapsack.append((weight, value, 1.0))
            capacity -= weight
        else:
            fraction = capacity / weight
            max_value += fraction * value
            knapsack.append((weight, value, fraction))
            capacity = 0

    return max_value, knapsack
```

Graph:-



Conclusion: As we can see that the graph looks very linear and looks like O(n) but in theory it should be O(nlogn). This could be because of our limited data set and computation power.