Design Analysis and Algorithem, Lab Assignment 5

Bhuvana Kanakam - SE21UCSE035

September 29, 2023

Question

Assignment implement and analyze a greedy algorithm for solving a classic problem.

Approach

- Choose one of the following classic problems for your assignment (or propose a similar problem to your instructor for approval):

  a. Fractional Knapsack Problem

  b. Activity Selection Problem

- Implement a Java/C/C++/Python program that solves the selected problem using a greedy algorithm. Your program should include the following components:

  a. Implement the greedy algorithm to solve the problem efficiently.

  b. Test your program with various problem instances to demonstrate its correctness and efficiency. You can generate random instances or use real data if applicable.

- Analyze the performance of your greedy algorithm in terms of time complexity and solution quality (e.g., how close it is to the optimal solution). Include this analysis in your documentation.

Answer

Below is the Python code that solves the Fractional Knapsack Problem using a greedy algorithm:

```
from matplotlib import pyplot
import numpy as np
import timeit
from functools import partial
import random

def fractionalknapsack(values, weights, Total_capacity):
    n = len(values)

    def score(i): return values[i] / weights[i]

    items = sorted(range(n), key=score, reverse=True)
    sel, value, weight = [], 0, 0
    for i in items:
        if weight + weights[i] <= Total_capacity:
            sel += [i]
            weight += weights[i]
            value += values[i]
    return value

def plotTC(fn, nMin, nMax, nInc, nTests):
    x = []
    y = []
    for i in range(nMin, nMax, nInc):
        N = i
        values = [random.randint(1, 100) for _ in range(N)]
        weights = [random.randint(1, 100) for _ in range(N)]
        Total_capacity = random.randint(1, 1000)
        testNTimer = timeit.Timer(partial(fn, values, weights, Total_capacity))
        t = testNTimer.timeit(number=nTests)
        x.append(i)
        y.append(t)
    p1 = pyplot.plot(x, y, 'o')

def main():
    plotTC(fractionalknapsack, 10, 1000, 10, 1000)
    pyplot.show()

if __name__ == '__main__':
    main()
```

Explanation

This program defines a function **fractional knapsack** that takes a list of items (each represented as a tuple of value and weight) and the knapsack's capacity as input. It then uses a greedy approach to select items with the highest value-to-weight ratio until the knapsack is full. The function returns the maximum value and a list of selected items with their respective values, weights, and fractions.

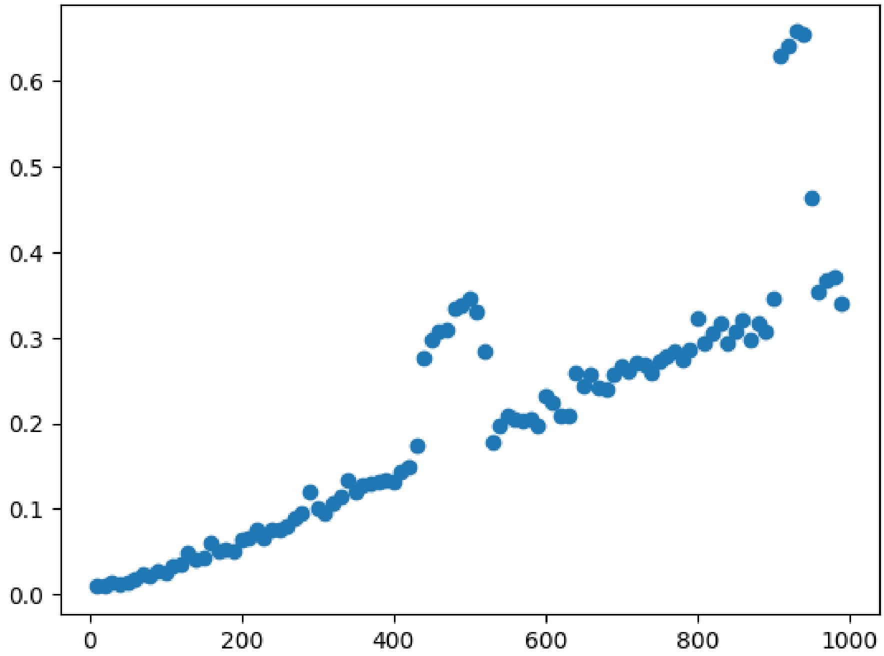

Figure 1: Time Complexity Analysis