Home

LEARNING OBJECTIVES

By the end of this lab session, you should be able to:

- Use Greedy Algorithm to solve 2 classic academic problems.
- Appreciate the intuitive simplicity and effectiveness of Greedy Algorithm.
- Understand that Greedy algorithms do not always yield optimal solutions, but for many problems they do.

Home

OVERVIEW OF LAB ACTIVITIES

Here are the activities for which you will upload solutions to xSITe or Gradescope:

1. Multiple Choice Quiz: Submit using xSite. Total 30 marks
2. Color Map: Submit using Gradescope. Total 40 marks
3. Knapsack: Submit using Gradescope. Total 30 marks

Home

MULTIPLE-CHOICE QUESTIONS

Select the best answer for each question. Some questions may have more than one correct option. In those cases, select all that apply.

Home

MULTIPLE-CHOICE QUESTION 1 OF 5

Which of the following conditions is **MOST specific** to proving that a greedy algorithm —rather than dynamic programming—will always produce an optimal solution?

a. The problem possesses optimal substructure

b. The greedy choice property holds

c. The solution space is finite and discrete

d. The subproblems are overlapping

Home

MULTIPLE-CHOICE QUESTION 2 OF 5

Which of the following problems can be correctly solved using a greedy algorithm? There may be more than one correct option.

a. Activity Selection Problem
b. 0/1 Knapsack Problem
c. Huffman Coding
d. Minimum Spanning Tree (e.g., Kruskal's or Prim's)
e. Single-source shortest path with non-negative edge weights (e.g., Dijkstra's algorithm)

Home

MULTIPLE-CHOICE QUESTION 3 OF 5

A greedy algorithm produces a suboptimal solution for a particular problem instance. Which conclusion is the **MOST accurate**?

a. The problem is likely NP-Hard

b. The greedy choice was implemented using the wrong sorting order

c. The problem does not satisfy the greedy choice property

d. The algorithm requires a memoization table to be correct

Home

MULTIPLE-CHOICE QUESTION 4 OF 5

Why is the Fractional Knapsack problem solvable using a greedy algorithm, while the 0/1 Knapsack problem is not?

a. Fractional Knapsack has fewer total combinations to check
b. 0/1 Knapsack has a higher theoretical time complexity
c. The ability to take fractions of items ensures that selecting by value/weight ratio never prevents achieving the optimal solution
d. 0/1 Knapsack requires sorting items by weight rather than value

Home

MULTIPLE-CHOICE QUESTION 5 OF 5

Which of the following are valid advantages **or** limitations of greedy algorithms? There may be more than one correct option.

a. They are generally easier to implement and more efficient than dynamic programming
b. They guarantee the global optimum for all optimization problems
c. They often have a time complexity of $O(n \log n)$ due to initial sorting of input
d. They may fail if a locally optimal choice prevents reaching the global optimum
e. They require maintaining a state table to solve overlapping subproblems

Home

EXERCISE 2: COLOR GRAPH

You are given an undirected graph represented by n vertices and m edges. Write a program to determine the minimum number of colors required to color the graph such that no two adjacent vertices have the same color.

PRACTICAL APPLICATION

Map Coloring & Geographic Boundaries. Coloring countries on a map so that no two neighboring countries share the same color. This is essentially the same problem, just visualized geographically.

Home

INPUT

- Number of edges
- Each edge as two space-separated integers

EXAMPLE INPUT

- Enter the number of edges: 6
- Enter each edge as two space-separated integers (e.g., ‘0 1’):

```
0 1
0 2
1 2
1 3
2 4
3 4
```

Home

OUTPUT

- Print the color assigned to each vertex and the total number of colors used.

EXAMPLE OUTPUT

```
Vertex  Color
0   0
1   1
2   2
3   0
4   1
Minimum number of colors required: 3
```

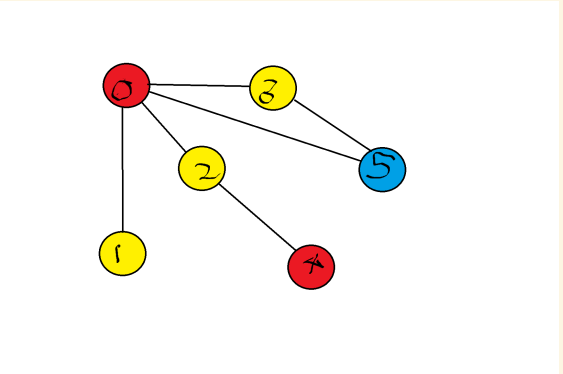
Home

HINTS: COLOR GRAPH

This problem is NP-complete, meaning that finding an exact solution efficiently for large graphs is computationally infeasible.

However, heuristic approaches like the greedy coloring algorithm can provide approximate solutions. The greedy algorithm doesn't always yield the optimal (minimum) number of colors. Its effectiveness heavily depends on the vertex ordering.

Ref TextBook Section 34-3 for more graph coloring problems.



Home

EXERCISE 3: KNAPSACK

You are a backpacker and you have a knapsack with a weight capacity of W. You are given n items, where each item has a weight and a value. Unlike the 0/1 knapsack problem, you are allowed to take fractions of items (i.e., you can take a part of an item, not necessarily the entire item).

Your goal is to maximize the total value of the items in the knapsack without exceeding the weight limit.

PRACTICAL APPLICATION

Maximizing value under a capacity constraint, especially in resource management. - Cargo Loading & Shipping. E.g. Airlines, shipping companies, or truck logistics need to decide which goods to load given limited weight or volume capacity.

Home

INPUT

- n (1 ≤ n ≤ 1000): The number of items.
- W (1 ≤ W ≤ 10^4): The weight capacity of the knapsack.
- An array of n elements where each element represents an item, containing:
  - weight[i] (1 ≤ weight[i] ≤ 10^4): The weight of the i-th item.
  - value[i] (1 ≤ value[i] ≤ 10^4): The value of the i-th item.

EXAMPLE INPUT

- Enter n the number of items: 3
- Enter W the weight capacity of the knapsack: 50
- Enter the elements of the array where each element represents an item's weight: 10, 20, 30
- Enter the elements of the array where each element represents an item's value: 60, 100, 120

Home

OUTPUT

- The maximum value that can be carried in the knapsack, up to two decimal places.

EXAMPLE OUTPUT

- Maximum value that can be carried: 240.00

Home

HINTS: KNAPSACK

1. Calculate Value-to-Weight Ratio: For each item, compute the ratio of its value to its weight (value/weight).
2. Sort Items by Ratio: Sort all items in descending order based on their value-to-weight ratio.

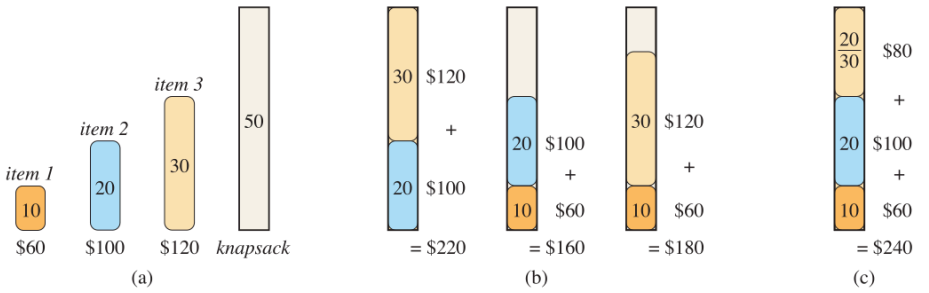


Figure 15.3 An example showing that the greedy strategy does not work for the 0-1 knapsack problem. **(a)** The thief must select a subset of the three items shown whose weight must not exceed 50 pounds. **(b)** The optimal subset includes items 2 and 3. Any solution with item 1 is suboptimal, even though item 1 has the greatest value per pound. **(c)** For the fractional knapsack problem, taking the items in order of greatest value per pound yields an optimal solution.

Home

HINTS: KNAPSACK

3. Select Items Greedily:

```
Initialize the total value to 0 and the remaining capacity of the knapsack to W.

Iterate through the sorted list of items:
  If the item's weight is less than or equal to the remaining capacity, add the entire item's value to
the total value and decrease the remaining capacity accordingly.
  If the item's weight exceeds the remaining capacity, take the fraction of the item that fits, add the
corresponding fraction of its value to the total value, and fill the knapsack to its capacity.

Terminate When Full: Stop the process once the knapsack reaches its weight capacity or all items have
been considered.
```

Home

REFERENCE