

Deccan Education Society's  
Fergusson College (Autonomous), Pune  
Department of Computer Science

**A**

**Report  
on**

***“Visual Representation of Fractional Knapsack Using  
Greedy Strategy”***

In partial fulfillment of Post Graduate course

in

M.Sc. Computer Science – I

(Semester -I)

CSC-520 Practical I

SUBMITTED BY

*Prajakta Prabhakar Pare (246249)*

*Vaibhavi Sanjay Pasalkar (246260)*

## Index

<b>Sr. No</b>	<b>Table of Content</b>	<b>Page No</b>
<b>1</b>	<b>Case Study description</b> <b>(Algorithm details and UI design details)</b>	<b>3</b>
<b>2</b>	<b>Screenshots</b>	<b>4</b>
<b>3</b>	<b>References (if you have used any)</b>	<b>11</b>

## **Algorithm Details:**

The Fractional Knapsack Problem is a classic optimization problem that can be solved efficiently using a greedy algorithm. In this problem, you are given a set of items, each characterized by a specific weight and value. The objective is to determine the optimal combination of items to include in a knapsack that has a limited weight capacity. The goal is to maximize the total value of the items in the knapsack while ensuring that the total weight does not exceed the knapsack's capacity. Unlike the 0/1 Knapsack Problem, where you can only take whole items, this problem allows you to take parts of items.

### **Steps: -**

1. **Calculate Ratios:** For each item, compute the value-to-weight ratio.
2. **Sort Items:** Sort items by their ratio in descending order.
3. **Initialize:** Set total value to 0 and remaining capacity to the knapsack's max weight.
4. **Select Items:**
  - If the item fits, add its value to total value and reduce remaining capacity.
  - If it doesn't fit, take the fraction that fits and break the loop.
5. **Return:** The total value is the maximum value in the knapsack.

### **❖ Time Complexity:**

- Best Case: ( $O(n \log n)$ )
- Average Case: ( $O(n \log n)$ )
- Worst Case: ( $O(n \log n)$ )

### **❖ Space Complexity:**

- Best Case: ( $O(n)$ )
- Average Case: ( $O(n)$ )
- Worst Case: ( $O(n)$ )

## UI Design Details:

The user interface for the Fractional Knapsack Problem is designed to be user-friendly and interactive, allowing users to input their data and visualize the results effectively.

Below are the key components of the UI:

- ❖ **Single Page Web Application:** It is a single-page web application. It is built as one continuous page, allowing users to move between sections smoothly. This design enhances the user experience by avoiding page reloads.
- ❖ **Color Palette:**
  - Primary Colors: Dark blue, white, light gray, and azure.
  - Accent Colors: Red for emphasis on important headings and links.
- ❖ **Typography:**
  - **Font Family:** Uses 'Lato' for a modern and clean appearance.
  - **Font Weights:** Different weights for headings and body text to establish hierarchy.
- ❖ **Title Section:** A prominent title section that clearly states "**Fractional Knapsack Problem**" with a visually appealing background. Dark blue background with white text for the title.
- ❖ **Navigation Bar:** A responsive navigation bar that allows users to navigate between different sections of the application, including Home, Calculator, Learn, and Code.
- ❖ **Footer:** A footer that includes links to the main sections of the application and copyright information.
  - **Background:** Dark blue similar to the title section.
  - **Text Colour:** White for contrast.
  - **Footer Links:** Centered and styled with hover effects.
- ❖ **Responsive Design:** The UI is designed to be responsive, ensuring that it works well on various devices, including desktops, tablets, and smartphones.

## HOME PAGE

- ❖ **Hero Section:** A prominent section with a motivational heading and a brief description.
  - Two call-to-action buttons ("Use Calculator" and "Start Learning") that change colour and slightly elevate on hover, encouraging user interaction.
  - A two-column layout with an overview of the Fractional Knapsack Problem on the left and a visual (image) on the right.
  - The info container has a shadow effect and rounded corners for a card-like appearance

#### ❖ **FAQ Container:**

- White background with padding and rounded corners.
- Each FAQ item is collapsible using the **<details>** and **<summary>** elements, enhancing interactivity.

### **CALCULATOR PAGE**

#### ❖ **Calculator Section:**

##### • **Input Fields:**

- Three labelled input fields for users to enter profits, weights, and the capacity of the sack. Each input field has a placeholder to guide users on the expected format (e.g., "e.g. 60,100,120").
- The input fields are styled to be user-friendly, with sufficient padding and margins for ease of use.
- Calculate Button: A button labelled "Calculate" that triggers the calculation process. The button is likely styled to stand out, encouraging user interaction.

#### ❖ **Result Display:**

- A paragraph element (**<p id="result">**) to display the calculated maximum profit or other results after the calculation is performed. This area is initially empty and will be populated dynamically based on user input.
  1. **Item Count:** A paragraph that shows the number of items processed.
  2. **Result Table:** A table that displays profits, weights, and profit-to-weight ratios. This table is structured with headers for clarity.
  3. **Detailed Table:** Another table that provides detailed calculations for each item, including profit calculations and remaining weight calculations. This table is designed to be populated dynamically based on the calculation results.

#### ❖ **Sack Container:**

- A visual representation of the knapsack, which may include an image (e.g., "ss.jpg") that visually represents the concept of a knapsack. This area is likely intended to show items that are being added to the knapsack as part of the calculation process.

#### ❖ **Interactive Elements:**

- **Dynamic Tables:** The tables for results and detailed calculations are designed to be populated dynamically, allowing for real-time updates based on user input. This interactivity enhances user engagement and provides immediate feedback.
- **Scroll Functionality:** The overflow: scroll style on the split section indicates that it can handle overflow content, allowing users to scroll through results if they exceed the visible area.

## LEARN PAGE

### ❖ **Example-Application Section:**

- **Flex Layout:** Displays the example and real-world applications side by side.
- **Knapsack-Example:**
  - Background Colour: White with rounded corners and a box shadow.
  - Text Styling: Headings are bold and coloured for emphasis.
  - Lists are formatted with bullet points for clarity.
- **Real-World Applications:** Each application is presented in a card format with a white background, border, and padding for separation. Headings are styled with a blue colour for consistency and emphasis.

### ❖ **Video Container:**

- Contains an embedded video for visual learning.
- Styled similarly to the intro section for consistency.

### ❖ **YouTube Videos Section:**

- **Container:** Centered with a title that is bold and underlined.
- **Video Thumbnails:** Responsive layout that adjusts based on screen size, ensuring videos are displayed neatly.

## CODE PAGE

### ❖ **Code Language Selection Section:**

- Four buttons for selecting programming languages (C, JavaScript, C++, Python).
- Styled with a white background, black text, and a border.
- Hover effects include a blue underline animation.
- Code is displayed in a `<pre>` tag for formatting, with a black background and white text for contrast.
- Each code section includes a "Copy Code" button that allows users to copy the code to the clipboard using JavaScript.
- Positioned to the right of the code title for easy access.

Fractional Knapsack Problem

HomeCalculatorLearnCode

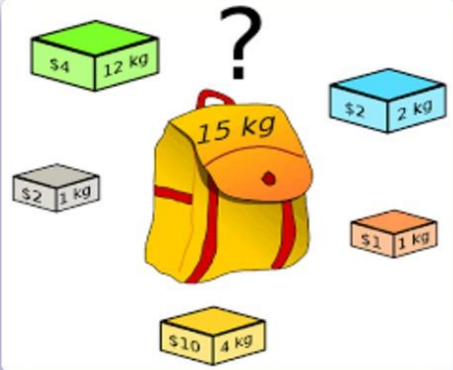
Solve optimization Problem with Precision and Ease!

"Effortlessly solve knapsack problems with user friendly interactive learning and a simple calculator!"

Use CalculatorStart Learning

Overview of the Fractional Knapsack Problem

The fractional knapsack problem is a well-known optimization problem where the goal is to maximize the total value of items you can carry in a knapsack with a weight limit. Unlike the 0/1 knapsack problem, you can take parts of an item, making it more flexible for real-world situations. This method helps you make better choices by considering the value of items relative to their weight. Additionally, the greedy algorithm used to solve this problem makes finding the best solution quick and efficient. It is commonly used in scenarios like budgeting, resource allocation, and investment strategies, where maximizing returns is crucial.



Frequently Asked Questions

► How does the fractional knapsack problem differ from the 0/1 knapsack problem?

► What is the significance of the value-to-weight ratio in this problem?

► Why is the fractional knapsack problem solved using a greedy algorithm?

► What are some real-world applications of the fractional knapsack problem?

► What is the time complexity of solving the fractional knapsack problem?

HomeCalculatorLearnCode

© 2024 Fractional Knapsack Problem. All rights reserved.

## CALCULATOR PAGE

KnapSack Calculator

C:/Users/lenovo/Desktop/Fractional\_Knapsack-/calculator.html

Fractional Knapsack Problem

[Home](#)[Calculator](#)[Learn](#)[Code](#)

### KnapSack Calculator

Profits (comma-separated):

Weights (comma-separated):

Capacity of Sack:

Calculate

**Maximum Profit is: 24.00**

### Let's solve the problem with the calculator


Number of items: 2

Profits	Weights	P-W Ratio
15	15	1.00
22	20	1.10
20	21	0.95

Item	Profit Calculation	Remaining Weight Calculation
I2	$0.00 + 22.00 = 22.00$	$22.00 - 20.00 = 2.00$
I1	$22.00 + 2.00 = 24.00$	$2.00 - 2.00 = 0.00$

Which Items Should I Choose ?

I2 I1



[Home](#)[Calculator](#)[Learn](#)[Code](#)

© 2024 Fractional Knapsack Problem. All rights reserved.



# LEARN PAGE

Fractional Knapsack Problem

C:/Users/lenovo/Desktop/Fractional\_Knapsack-/learn.html

Fractional Knapsack Problem

HomeCalculatorLearnCode


### Fractional Knapsack Problem.

The Fractional Knapsack Problem is a problem in combinatorial optimization where you can take fractions of items to maximize the total value in a knapsack of limited capacity.

#### Steps to Solve the Fractional Knapsack Problem

1. Input the Items
2. Calculate Value-to-Weight Ratios
3. Sort Items by Value-to-Weight Ratio
4. Initialize Variables
5. Select Items
6. Output the Result

### Watch and Learn



### EXAMPLE

Consider items with the following values and weights:

- Item A: Value = 80, Weight = 15
- Item B: Value = 120, Weight = 30
- Item C: Value = 60, Weight = 10

With a knapsack capacity of 40, the optimal solution would involve taking:

#### Step-by-Step Breakdown

1. **Input the Items:**
  - Item A: Value = 80, Weight = 15
  - Item B: Value = 120, Weight = 30
  - Item C: Value = 60, Weight = 10
2. **Calculate Value-to-Weight Ratios:**
  - Item A:  $80 / 15 = 5.33$
  - Item B:  $120 / 30 = 4.00$
  - Item C:  $60 / 10 = 6.00$

### REAL-WORLD APPLICATIONS

#### Finance

Used in portfolio optimization to maximize returns. This involves selecting the best combination of assets to achieve the highest possible return for a given level of risk.

#### Logistics

Helps in optimizing cargo loading in transportation. Efficient loading strategies reduce costs and improve delivery times by maximizing space utilization.

#### Manufacturing

Optimizes the selection of materials to maximize production efficiency. This involves choosing the best materials and processes to reduce waste and increase output.

## 2. Calculate Value-to-Weight Ratios:

- Item A:  $80 / 15 = 5.33$
- Item B:  $120 / 30 = 4.00$
- Item C:  $60 / 10 = 6.00$

## 3. Sort Items by Value-to-Weight Ratio:

- Item C: 6.00
- Item A: 5.33
- Item B: 4.00

## 4. Initialize Variables: Set total value to 0 and remaining capacity to 40.

## 5. Select Items:

- Take Item C: Full (Weight = 10, Value = 60)
- Take Item A: Full (Weight = 15, Value = 80)
- Take Item B: Quarter (Weight = 10, Value = 40)

## 6. Output the Result: Total Value = $60 + 80 + 40 = 180$

## Manufacturing

Optimizes the selection of materials to maximize production efficiency. This involves choosing the best materials and processes to reduce waste and increase output.

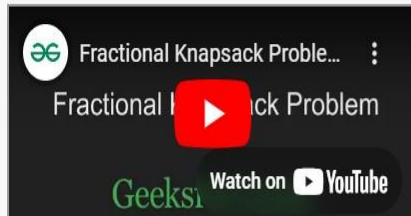
## Telecommunications

Used in bandwidth allocation to maximize data transmission efficiency. Proper allocation ensures that network resources are utilized effectively, improving service quality.

## Healthcare

Assists in optimizing the allocation of medical resources and treatments to maximize patient outcomes. This involves prioritizing care based on patient needs and resource availability.

## MORE VIDEOS ON FRACTIONAL KNAPSACK



[Home](#) [Calculator](#) [Learn](#) [Code](#)

© 2024 Fractional Knapsack Problem. All rights reserved.

# CODE PAGE

Fractional Knapsack Problem

HomeCalculatorLearnCode

## Code for Fractional Knapsack Problem

Explore solutions for the fractional knapsack problem across different programming languages and learn the implementation details.

CJavaScriptC++Python

JavaScript Code

Copy Code

```
function knapsack(weights, values, capacity) {
  let n = weights.length;
  let dp = Array(n+1).fill().map(() => Array(capacity+1).fill(0));

  for (let i = 0; i <= n; i++) {
    for (let w = 0; w <= capacity; w++) {
      if (i == 0 || w == 0) {
        dp[i][w] = 0;
      } else if (weights[i-1] <= w) {
        dp[i][w] = Math.max(values[i-1] + dp[i-1][w - weights[i-1]], dp[i-1][w]);
      } else {
        dp[i][w] = dp[i-1][w];
      }
    }
  }

  return dp[n][capacity];
}

let weights = [1, 2, 3, 8, 7, 4];
let values = [20, 5, 10, 40, 15, 25];
let capacity = 10;
console.log(knapsack(weights, values, capacity));
```

HomeCalculatorLearnCode

© 2024 Fractional Knapsack Problem. All rights reserved.