Knapsack Visualizer (Fractional & 0/1)

A MINI PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT FOR THE AWARD OF THE DEGREE OF

MASTERS IN COMPUTER APPLICATIONS (AI &ML)

Submitted by

Vivek Gaur (25MCI10130)

Supervisor

Dr. Gurpreet Kaur (E5742)

ASSISTANT PROFESSOR, UIC

CHANDIGARH UNIVERSITY

UNIVERSITY INSTITUTE OF COMPUTING

CHANDIGARH UNIVERSITY

MOHALI, PUNJAB-140301

October,2025

ACKNOWLEDGEMENT

We deem it a pleasure to acknowledge our sense of gratitude to our project guide Dr. Gurpreet Kaur (Assistant Professor, UIC) under whom we have carried out the project work. Her incisive and objective guidance and timely advice encouraged us with constant flow of energy to continue the work.

We wish to reciprocate in full measure the kindness shown by Dr. Krishna Tuli (H.O.D, University Institute of Computing) who inspired us with his valuable suggestions in successfully completing the project work.

We shall remain grateful to Dr. Manisha Malhotra, Additional Director, University Institute of Computing, for providing us a strong academic atmosphere by enforcing strict discipline to do the project work with utmost concentration and dedication.

Finally, we must say that no height is ever achieved without some sacrifices made at some end and it is here where we owe our special debt to our parents and our friends for showing their generous love and care throughout the entire period of time.

Date: 7.11.2024

Place: Chandigarh University, Mohali, Punjab.

By: Vivek Gau, UID- 25MCI10130.

TABLE OF CONTENTS

ABSTRACT

The **Knapsack Visualizer** is an interactive web-based application designed to demonstrate the working of the **Fractional Knapsack** and **0/1 Knapsack** algorithms using dynamic animations. The project uses **HTML, CSS, and JavaScript** to visualize how items are selected and packed to achieve maximum profit within a given capacity.

The application provides two modes — *Fractional (Greedy)* and *0/1 (Dynamic Programming)* — allowing users to enter profits, weights, and capacity, then observe the algorithm's execution in real-time.
The main goal is to help students understand algorithmic behavior, logic, and optimization techniques in an intuitive and engaging manner.

CHAPTER 1: INTRODUCTION

1.1 Project Overview

The Knapsack Visualizer project is developed as part of the Design and Analysis of Algorithms (DAA) curriculum to showcase how optimization algorithms can be made interactive. It visually represents the process of selecting items with given profits and weights to maximize total profit under a fixed capacity constraint.
It supports both:

- Fractional Knapsack (Greedy approach)
- 0/1 Knapsack (Dynamic Programming)

The interface dynamically animates each step, helping users understand core DAA concepts interactively.

1.2 Objective

- To visualize both Fractional and 0/1 Knapsack problems.
- To demonstrate the working of Greedy and DP approaches.
- To allow user input for profits, weights, and capacity.
- To provide animated, step-by-step visualization of the algorithm’s execution.
- To enhance conceptual understanding of optimization in algorithms.

1.3 Motivation

Algorithm visualization helps bridge the gap between theory and understanding. Many students struggle to grasp dynamic programming and greedy logic by reading only pseudocode. This project aims to make learning more engaging by visually representing how algorithms make decisions step by step.

CHAPTER 2: SYSTEM ANALYSIS

2.1 Existing System

• Traditional methods involve static pseudocode and dry-run tables.
• Students find it hard to visualize the dynamic process of item selection.
• Lacks real-time interaction and engagement.

2.2 Proposed System

The proposed system provides:
• A web-based interface for user interaction.
• Real-time visual feedback of algorithm execution.
• Separate tabs for **Fractional Knapsack** and **0/1 Knapsack**.
• Algorithm logs, notes, and live capacity tracking.

2.3 Feasibility Study

Technical Feasibility: Implemented using simple web technologies (HTML, CSS, JavaScript).
Operational Feasibility: Interactive and easy-to-use interface for students.
Economic Feasibility: Fully open-source, with no external costs or frameworks required.

CHAPTER 3: SYSTEM DESIGN AND IMPLEMENTATION

3.1 System Architecture

The system consists of three major layers:

- **Input Layer:** Accepts profits, weights, and capacity from the user.
- **Processing Layer:** Executes the algorithm logic (Greedy / DP).
- **Visualization Layer:** Animates item placement into the knapsack bag and logs algorithm steps.

3.2 Module Description

• **Input Module:** Takes user-entered profits, weights, and capacity.
• **Visualization Module:** Displays real-time item movement and logs.
• **Algorithm Module:** Handles both Fractional (Greedy) and 0/1 (DP) computations.
• **UI Module:** Manages tab switching and resetting the simulation.

3.3 Technology Stack

Component	Technology Used
Frontend	**HTML5, CSS3**
Programming	**JavaScript (Vanilla)**
Algorithm	**Greedy & Dynamic Programming**
Animation	**CSS Transitions, JavaScript DOM**
Compatibility	**Works on all modern browsers**

CHAPTER 4: SYSTEM TESTING AND RESULTS

4.1 Testing Approach

Testing included validation of inputs, UI functionality, and correctness of results for both algorithm types.

- Functional Testing: Verified create, reset, and run buttons.
- Algorithm Testing: Checked profit and weight calculations for both approaches.
- Visual Testing: Ensured smooth and accurate animations during execution.

4.2 Test Cases

Test Case	**Input**	**Expected Output**	**Result**
TC1	P=60,100,120 W=10,20,30 C=50 (Fractional)	Max profit = 240	Pass
TC2	P=60,100,120 W=10,20,30 C=50 (0/1)	Max profit = 220	Pass
TC3	Empty inputs	Alert “Enter profits and weights”	Pass
TC4	Mismatched list lengths	Error “Profits and weights counts must match”	Pass

4.3 Results

The Knapsack Visualizer successfully demonstrates both Greedy and DP approaches, producing accurate results with clear animations.
Users can observe the filling process, profit accumulation, and remaining capacity in real time.

CHAPTER 5. CODE

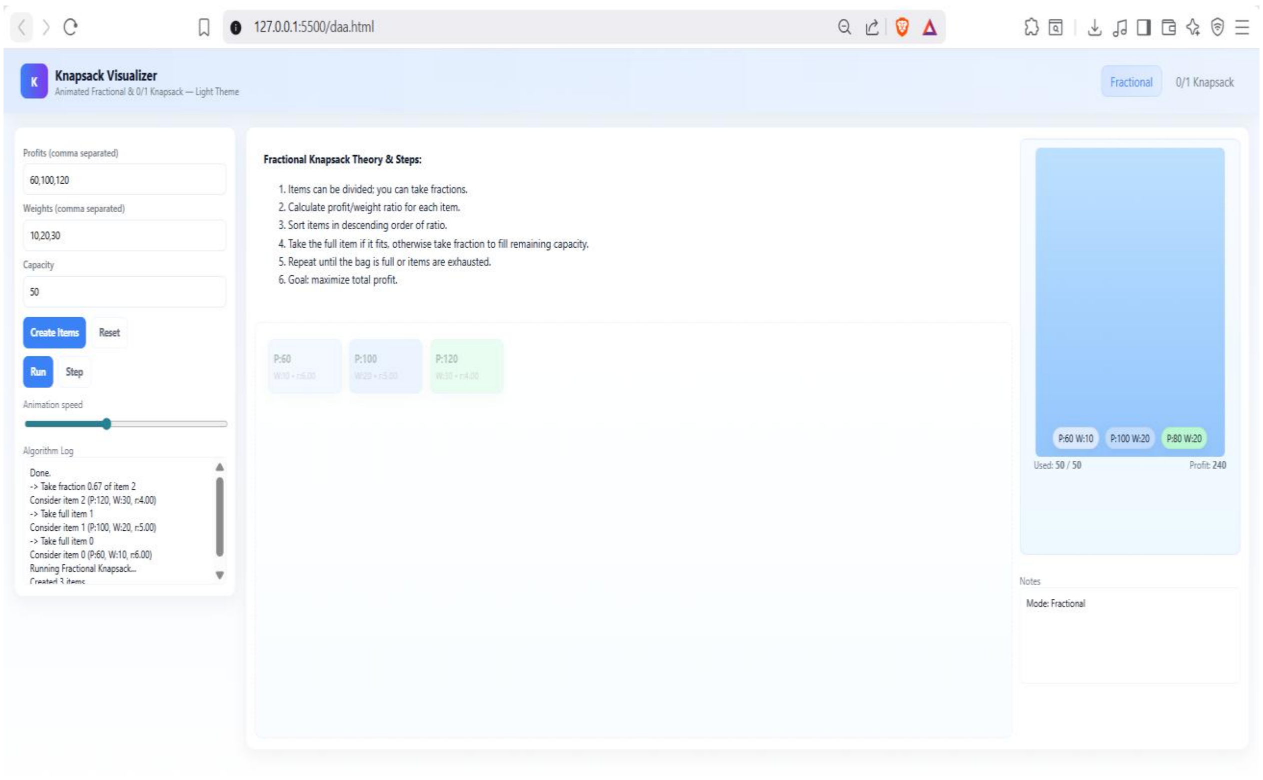
```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1" />
<title>Knapsack Visualizer — Light Theme</title>
<link rel="preconnect" href="https://fonts.gstatic.com">
<style>
  :root{
    --bg: #f7fbff;
    --panel: #ffffff;
    --muted: #6b7280;
    --accent: #3b82f6;
    --accent-2: #60a5fa;
    --card-shadow: 0 6px 24px rgba(15,23,42,0.06);
  }
  *{box-sizing:border-box;font-family:Inter,ui-sans-serif,system-ui,-apple-system,"Segoe UI",Roboto,"Helvetica Neue",Arial;}
  html,body{height:100%;margin:0;background:linear-gradient(180deg,#f3f8ff 0%, #ffffff 100%);color:#0f172a;}
  .app{min-height:100vh;display:flex;flex-direction:column;}
  header{background:linear-gradient(90deg, rgba(59,130,246,0.10), rgba(96,165,250,0.06));padding:18px 28px;display:flex;align-items:center;justify-content:space-between;gap:12px;box-shadow:var(--card-shadow);}
```

```css
  header .title{display:flex;gap:12px;align-items:center}
  .logo{width:44px;height:44px;border-radius:10px;background:linear-gradient(90deg,var(--accent),#7c3aed);display:flex;align-items:center;justify-content:center;color:white;font-weight:700}
  h1{margin:0;font-size:18px}
  .subtitle{color:var(--muted);font-size:13px}
  .tabs{display:flex;gap:8px}
  .tab{padding:8px 14px;border-radius:10px;border:1px solid transparent;background:transparent;color:var(--muted);cursor:pointer}
  .tab.active{background:linear-gradient(90deg, rgba(59,130,246,0.12), rgba(96,165,250,0.08));color:var(--accent);border-color:rgba(59,130,246,0.12)}
  .container{display:grid;grid-template-columns:360px 1fr;gap:18px;padding:18px;align-items:start;}
  .panel{background:var(--panel);border-radius:12px;padding:14px;box-shadow:var(--card-shadow);}
  .controls label{display:block;color:var(--muted);font-weight:600;margin-top:8px;font-size:13px}
  input[type="text"], input[type="number"]{width:100%;padding:10px;border-radius:8px;border:1px solid #e6eef8;margin-top:6px}
  .row{display:flex;gap:8px;margin-top:12px}
  .btn{padding:10px 12px;border-radius:8px;border:0;background:var(--accent);color:white;cursor:pointer;font-weight:700}
  .btn.ghost{background:transparent;border:1px solid #eef6ff;color:var(--muted)}
  .muted{color:var(--muted);font-size:13px}
  .visual-wrap{display:flex;gap:12px;align-items:flex-start}
  .items-area{flex:1; min-height:520px;border-radius:10px;padding:12px;border:1px dashed #e6f0ff; position:relative;background:linear-gradient(180deg,#ffffff,#f9fdff); overflow:auto;}
  .items-grid{display:flex;flex-wrap:wrap;gap:12px;padding:8px}
  .item{width:120px;height:68px;border-radius:10px;padding:8px;display:flex;flex-direction:column;justify-content:center;box-shadow:0 6px 18px rgba(15,23,42,0.06);font-weight:700;color:#032;position:relative;background:#dbeafe;border:2px solid #bfdbfe;}
  .item .p{font-size:14px}
  .item .w{font-size:12px;color:#0f172a80;margin-top:4px;font-weight:600}
  .bag-column{width:360px;display:flex;flex-direction:column;gap:12px;align-items:stretch}
  .bag{height:520px;border-radius:10px;padding:10px;border:2px solid rgba(59,130,246,0.12);background:linear-gradient(180deg,#f8fbff,#eef9ff);position:relative;overflow:hidden;display:flex;flex-direction:column;align-items:center;}
  .bag-viewport{width:92%;height:78%;border-radius:8px;border:1px dashed
rgba(59,130,246,0.08);position:relative;background:linear-
gradient(180deg,#ffffff,#f4fbff);overflow:hidden}

  .bag-fill{position:absolute;left:0;bottom:0;height:0%;width:100%;background:linear-
gradient(180deg,#bde0fe,#93c5fd);display:flex;align-items:flex-end;justify-
content:center;gap:6px;padding-bottom:8px;transition:height 500ms ease;}

  .placed-item{height:28px;border-radius:14px;padding:6px 8px;font-
size:13px;background:white;border:1px solid rgba(15,23,42,0.06);display:inline-flex;align-
items:center;gap:8px;margin:2px;box-shadow:0 6px 14px rgba(2,6,23,0.04);}

  .bag-stats{display:flex;justify-content:space-between;align-items:center;padding-top:8px}

  .small-muted{color:var(--muted);font-size:13px}

  .log{height:160px;overflow:auto;border-
radius:8px;padding:10px;background:#ffffff;border:1px solid #eef6ff;font-
size:13px;color:#0b1220}

  .theory{background:var(--panel);padding:14px;border-radius:10px;margin-
bottom:14px;font-size:15px;line-height:1.5;color:#0f172a;}

  @media (max-width:1000px){.container{grid-template-columns:1fr; padding:12px}.bag-
column{width:100%}}

</style>

</head>

<body>

<div class="app">

  <header>

    <div class="title">

      <div class="logo">K</div>

      <div>

        <h1>Knapsack Visualizer</h1>

        <div class="subtitle">Animated Fractional & 0/1 Knapsack — Light Theme</div>

      </div>

    </div>

    <div class="tabs">

      <div class="tab active" id="tabFraction">Fractional</div>

      <div class="tab" id="tabZero">0/1 Knapsack</div>

    </div>

  </header>
  <div class="container">
    <aside class="panel controls" aria-label="controls">
      <label>Profits (comma separated)</label>
      <input id="profits" type="text" placeholder="e.g., 60,100,120" value="60,100,120">

      <label>Weights (comma separated)</label>
      <input id="weights" type="text" placeholder="e.g., 10,20,30" value="10,20,30">

      <label>Capacity</label>
      <input id="capacity" type="number" placeholder="e.g., 50" value="50">

      <div class="row">
        <button class="btn" id="createBtn">Create Items</button>
        <button class="btn ghost" id="resetBtn">Reset</button>
      </div>

      <div class="row" style="margin-top:10px">
        <button class="btn" id="runBtn">Run</button>
        <button class="btn ghost" id="stepBtn">Step</button>
      </div>

      <div style="margin-top:12px">
        <div class="muted">Animation speed</div>
        <input id="speed" type="range" min="200" max="1200" value="600"
style="width:100%;margin-top:8px">
      </div>

      <div style="margin-top:12px" class="muted">Algorithm Log</div>
      <div class="log" id="log"></div>
    </aside>

    <section class="panel visual-wrap" aria-label="visual">
      <div style="flex:1">
  <div class="theory" id="theoryFraction">
    <strong>Fractional Knapsack Theory & Steps:</strong>
    <ol>
      <li>Items can be divided; you can take fractions.</li>
      <li>Calculate profit/weight ratio for each item.</li>
      <li>Sort items in descending order of ratio.</li>
      <li>Take the full item if it fits, otherwise take fraction to fill remaining capacity.</li>
      <li>Repeat until the bag is full or items are exhausted.</li>
      <li>Goal: maximize total profit.</li>
    </ol>
  </div>

  <div class="theory" id="theoryZero" style="display:none">
    <strong>0/1 Knapsack Theory & Steps:</strong>
    <ol>
      <li>Items cannot be divided; either take whole item or not.</li>
      <li>Use Dynamic Programming (DP) to find max profit.</li>
      <li>Build DP table with rows = items, columns = capacities.</li>
      <li>dp[i][w] = max(profit[i-1]+dp[i-1][w-weight[i-1]], dp[i-1][w])</li>
      <li>Backtrack DP table to find chosen items.</li>
      <li>Animate items into the bag to visualize packing.</li>
    </ol>
  </div>

  <div class="items-area" id="itemsArea">
    <div class="items-grid" id="itemsGrid" aria-live="polite"></div>
  </div>
</div>

<div class="bag-column">
  <div class="bag" id="bag">
    <div class="bag-viewport" id="bagViewport">
```

```
          <div class="bag-fill" id="bagFill"></div>
        </div>
        <div class="bag-stats" style="width:100%;padding:0 12px;">
          <div class="small-muted">Used: <strong id="used">0</strong> / <strong
id="total">0</strong></div>
          <div class="small-muted">Profit: <strong id="profit">0</strong></div>
        </div>
      </div>
      <div style="margin-top:12px">
        <div class="muted">Notes</div>
        <div class="log" id="notes" style="height:120px"></div>
      </div>
    </div>
  </section>
 </div>
</div>

<script>
const tabFraction=document.getElementById('tabFraction');
const tabZero=document.getElementById('tabZero');
let mode='fractional';
const theoryFraction=document.getElementById('theoryFraction');
const theoryZero=document.getElementById('theoryZero');

const createBtn=document.getElementById('createBtn');
const resetBtn=document.getElementById('resetBtn');
const runBtn=document.getElementById('runBtn');
const stepBtn=document.getElementById('stepBtn');
const itemsGrid=document.getElementById('itemsGrid');
const bagViewport=document.getElementById('bagViewport');
const bagFill=document.getElementById('bagFill');
const usedSpan=document.getElementById('used');
const totalSpan=document.getElementById('total');
const profitSpan=document.getElementById('profit');

const logDiv=document.getElementById('log');

const notesDiv=document.getElementById('notes');

const speedEl=document.getElementById('speed');


let items=[],capacity=0,running=false,stepMode=false;


const
colors=["#dbeafe","#bfdbfe","#bbf7d0","#fff7ed","#fde68a","#fecaca","#fef3c7","#e9d5ff","
#c7d2fe"];


function clearLogs(){logDiv.innerHTML='';notesDiv.innerHTML='';}

function log(msg){const
p=document.createElement('div');p.textContent=msg;logDiv.prepend(p);}

function note(msg){const
p=document.createElement('div');p.textContent=msg;notesDiv.prepend(p);}


function parseInputs(){

  clearLogs();

  const pstr=document.getElementById('profits').value.trim();

  const wstr=document.getElementById('weights').value.trim();

  const cap=Number(document.getElementById('capacity').value);

  if(!pstr||!wstr){alert('Enter profits and weights');return null;}

  const pArr=pstr.split(',').map(s=>Number(s.trim())).filter(s=>!isNaN(s));

  const wArr=wstr.split(',').map(s=>Number(s.trim())).filter(s=>!isNaN(s));

  if(pArr.length!==wArr.length){alert('Profits and weights counts must match');return null;}

  if(!Number.isFinite(cap)||cap<=0){alert('Capacity must be a positive number');return null;}

  return {profits:pArr,weights:wArr,capacity:cap};

}


function createItems(){

  const parsed=parseInputs();if(!parsed)return;

  resetAll();capacity=parsed.capacity;totalSpan.textContent=capacity;

  const {profits,weights}=parsed;items=[];
  for(let i=0;i<profits.length;i++){
    const p=profits[i],w=weights[i];const ratio=p/w;
    const el=document.createElement('div');el.className='item';
    el.style.background=colors[i%colors.length];
    el.innerHTML=`<div class="p">P:${p}</div><div class="w">W:${w} •
r:${ratio.toFixed(2)}</div>`;
    itemsGrid.appendChild(el);items.push({idx:i,profit:p,weight:w,ratio,elem:el});
  }
  log(`Created ${items.length} items.`);
  note(`Mode: ${mode==='fractional'?'Fractional':'0/1 DP'}`);
}

function resetAll(){
  running=false;stepMode=false;itemsGrid.innerHTML='';bagFill.style.height='0%';bagFill.inn
erHTML='';
  usedSpan.textContent='0';profitSpan.textContent='0';totalSpan.textContent='0';items=[];
}

// Tabs
tabFraction.addEventListener('click',()=>{
  mode='fractional';tabFraction.classList.add('active');tabZero.classList.remove('active');
  theoryFraction.style.display='block';theoryZero.style.display='none';resetAll();createItems();
});
tabZero.addEventListener('click',()=>{
  mode='zeroone';tabZero.classList.add('active');tabFraction.classList.remove('active');
  theoryZero.style.display='block';theoryFraction.style.display='none';resetAll();createItems();
});

createBtn.addEventListener('click',createItems);
resetBtn.addEventListener('click',()=>{resetAll();clearLogs();});

// Animate helper
function animateItemIntoBag(itemObj,fraction=1){
  return new Promise(resolve=>{
    const duration=Math.max(300,Number(speedEl.value));
    const start=itemObj.elem.getBoundingClientRect();
    const bagRect=bagViewport.getBoundingClientRect();
    const clone=itemObj.elem.cloneNode(true);
    clone.style.position='fixed';clone.style.left=start.left+'px';clone.style.top=start.top+'px';
    clone.style.width=start.width+'px';clone.style.height=start.height+'px';
    clone.style.margin='0';clone.style.zIndex=9999;
    clone.style.transition=`transform ${duration}ms cubic-bezier(.2,.9,.2,1), opacity
${duration/2}ms`;
    document.body.appendChild(clone);
    const placedCount=bagFill.querySelectorAll('.placed-item').length;
    const pillWidth=Math.min(100,start.width*0.9);
    const paddingLeft=8+(placedCount%6)*(pillWidth/5+6);
    const targetX=bagRect.left+10+paddingLeft;
    const targetY=bagRect.bottom-28-Math.floor(placedCount/6)*32;
    const scale=0.75;
    const translateX=targetX-start.left;
    const translateY=targetY-start.top;
    requestAnimationFrame(()=>{clone.style.transform=`translate(${translateX}px,
${translateY}px) scale(${scale})`;clone.style.opacity='0.95';});
    setTimeout(()=>{
      clone.remove();
      const pill=document.createElement('div');pill.className='placed-item';
      pill.style.background=itemObj.elem.style.background||'#fff';
      const profitMoved=(itemObj.profit*fraction);
      const weightMoved=(itemObj.weight*fraction);
      pill.textContent=`P:${Math.round(profitMoved)} W:${Math.round(weightMoved)}`;
      bagFill.appendChild(pill);
      resolve({profitMoved,weightMoved});
    },duration+40);
  });
}
```

```
function updateBagUI(used,total,profit){
 const percent=Math.min(100,Math.round((used/total)*10000)/100||0);
 bagFill.style.height=percent+'%';
 usedSpan.textContent=Math.round(used);
 totalSpan.textContent=total;
 profitSpan.textContent=Math.round(profit);
}

// Fractional
async function runFractional(){
 log('Running Fractional Knapsack...');
 const sorted=[...items].sort((a,b)=>b.ratio-a.ratio);
 let remaining=capacity,totalProfit=0,used=0;
 updateBagUI(used,capacity,totalProfit);
 for(let it of sorted){
  if(remaining<=0)break;
  log(`Consider item ${it.idx} (P:${it.profit}, W:${it.weight}, r:${it.ratio.toFixed(2)})`);
  await new Promise(r=>setTimeout(r,200));
  if(it.weight<=remaining){
   log(` -> Take full item ${it.idx}`);
   const res=await animateItemIntoBag(it,1);
   used+=it.weight;totalProfit+=it.profit;it.elem.style.opacity='0.3';
  }else{
   const fraction=remaining/it.weight;
   log(` -> Take fraction ${fraction.toFixed(2)} of item ${it.idx}`);
   const res=await animateItemIntoBag(it,fraction);
   used+=it.weight*fraction;totalProfit+=it.profit*fraction;it.elem.style.opacity='0.3';
  }
  remaining=capacity-used;updateBagUI(used,capacity,totalProfit);
  await new Promise(r=>setTimeout(r,Number(speedEl.value)));
 }
 log('Done.');
}

// 0/1 Knapsack using DP
async function runZeroOne(){
 log('Running 0/1 Knapsack...');
 const n=items.length;
 const W=capacity;
 const dp=Array.from({length:n+1},()=>Array(W+1).fill(0));
 for(let i=1;i<=n;i++){
  for(let w=0;w<=W;w++){
   if(items[i-1].weight<=w){
    dp[i][w]=Math.max(items[i-1].profit+dp[i-1][w-items[i-1].weight],dp[i-1][w]);
   }else dp[i][w]=dp[i-1][w];
  }
 }
 let w=W,totalProfit=dp[n][W];let used=0;
 const taken=[];
 for(let i=n;i>0;i--){
  if(dp[i][w]!=dp[i-1][w]){
   taken.push(items[i-1]);
   w-=items[i-1].weight;
  }
 }
 taken.reverse();
 for(let it of taken){
  log(`Take item ${it.idx} (P:${it.profit}, W:${it.weight})`);
  const res=await animateItemIntoBag(it,1);
  used+=it.weight;
  updateBagUI(used,capacity,totalProfit);
  it.elem.style.opacity='0.3';
  await new Promise(r=>setTimeout(r,Number(speedEl.value)));
  }
  log('Done.');
}

runBtn.addEventListener('click',()=>{
  if(mode==='fractional') runFractional();
  else runZeroOne();
});

</script>
</body>
</html>
```

CHAPTER 6. OUTPUT



Profits (comma separated): 60,100,120

Weights (comma separated): 10,20,30

Capacity: 50

Fractional Knapsack Theory & Steps:

1. Items can be divided; you can take fractions.
2. Calculate profit/weight ratio for each item.
3. Sort items in descending order of ratio.
4. Take the full item if it fits, otherwise take fraction to fill remaining capacity.
5. Repeat until the bag is full or items are exhausted.
6. Goal: maximize total profit.

Algorithm Log

```
Done.
-> Take fraction 0.67 of item 2
Consider item 2 (P:120, W:30, r:4.00)
-> Take full item 1
Consider item 1 (P:100, W:20, r:5.00)
-> Take full item 0
Consider item 0 (P:60, W:10, r:6.00)
Running Fractional Knapsack...
Created 3 items
```

Used: **50 / 50** Profit: **240**

Notes: Mode: Fractional

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Conclusion

The Knapsack Visualizer provides an effective platform to understand and compare Greedy and Dynamic Programming approaches. The real-time animations and logs improve comprehension and retention of DAA concepts.

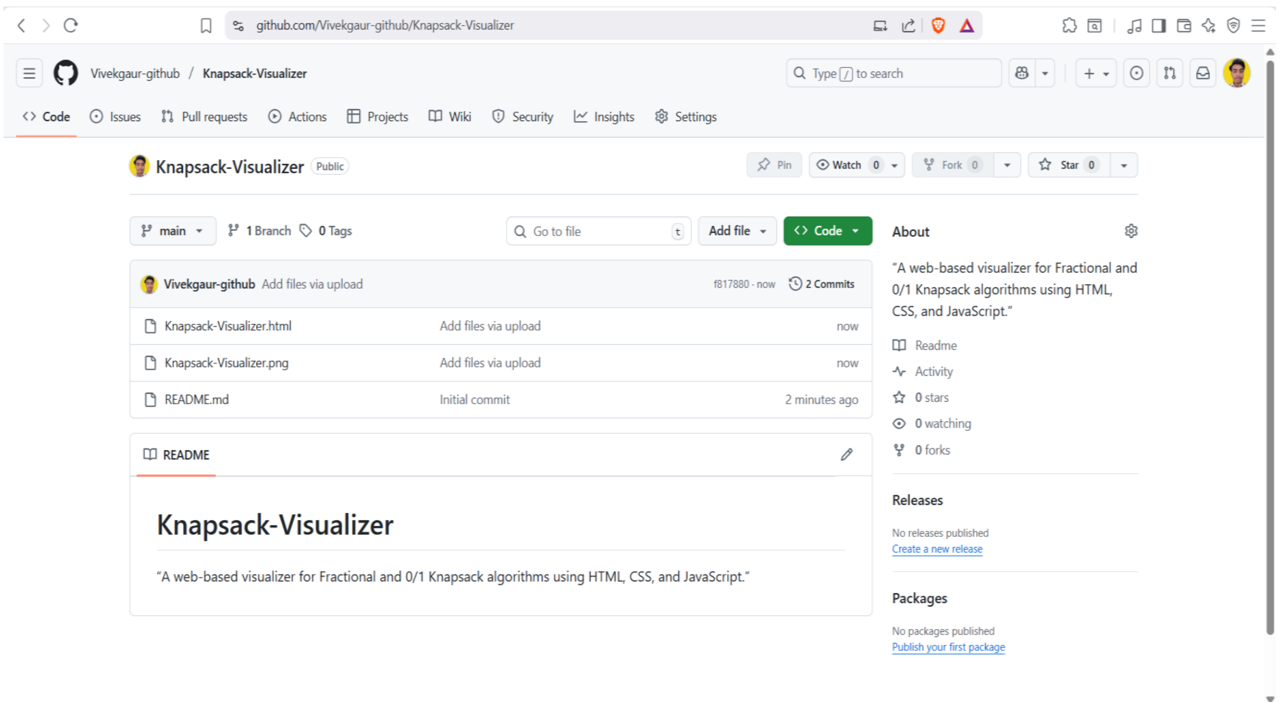
7.2 Future Enhancement

• Add visualization for **DP table formation**.
• Include **time complexity and comparison charts**.
• Implement **interactive step-by-step mode** for learning.
• Add **export-to-PDF and screenshot features**.
• Extend to other problems like **Subset Sum, Traveling Salesman, and Graph algorithms**.

8. REFERENCES

- **GeeksforGeeks – Explanation of Fractional and 0/1 Knapsack algorithms.**
- **W3Schools – HTML, CSS, and JavaScript tutorials for web development.**
- **MDN Web Docs (Mozilla Developer Network) – JavaScript DOM and event handling reference.**
- **YouTube Tutorials – Visualization of Knapsack problems and DAA concepts.**
- **Wikipedia – Knapsack problem and algorithm theory overview.**

DEPLOYMENT: GITHUB



Vivekgaur-github Add files via upload f817880 · now 2 Commits

Knapsack-Visualizer.html	Add files via upload	now
Knapsack-Visualizer.png	Add files via upload	now
README.md	Initial commit	2 minutes ago

README

Knapsack-Visualizer

"A web-based visualizer for Fractional and 0/1 Knapsack algorithms using HTML, CSS, and JavaScript."

About

"A web-based visualizer for Fractional and 0/1 Knapsack algorithms using HTML, CSS, and JavaScript."

Readme
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package