**Моделювання задачі «Рюкзак» методом динамічного програмування**

**Мета**: Навчитися формалізувати задачу оптимізації ресурсів та розв’язувати її за допомогою методів динамічного програмування. Побудувати оптимальне рішення для задачі «Рюкзак», проаналізувати таблицю переходів станів та візуалізувати вибраний набір предметів.

**Короткі теоретичні відомості**

**Використовуючи наявні відкриті джерела інформації, на кшталт *ChatGPT, Wikipedia*… представте короткі теоретичні відомості** (як Ви це бачите, щоб студент, який робитиме роботу, чітко зрозумів теорію і як потрібно зробити роботу)

**Завдання до лабораторної роботи**

1. **Побудувати таблицю** dp[i][w] за допомогою методу динамічного програмування, заповнити таблицю згідно рекурентного співвідношення; (*варіант потрібно отримати у викладача*)
2. **Знайти максимальну можливу цінність**, яку можна розмістити в рюкзаку для заданого набору предметів і максимальної ваги
3. **Відновити перелік предметів**, що входять до оптимального рішення
4. Реалізувати програмне забезпечення:

* засобами **JavaScript** (використовуючи вбудовані HTML таги для таблиць або іньші) побудувати таблицю завдання 1;
* відобразити проміжні етапи побудови\*. *Фраза "відобразити проміжні етапи побудови " в контексті візуалізації алгоритму означає, що можна додавати анімації або оновлення, щоб показувати, як змінюється таблиця динамічного програмування****на кожному кроці****алгоритму (відновлення переліку предметів в т.ч.). Це дозволяє зрозуміти, як працює сам алгоритм, і бачити, як поетапно обчислюються значення у таблиці*

1. Оформити звіт виконаної роботи, згідно взірця
2. Отримані результати (**лістинг коду** завантажити на **репозиторій**). У звіті потрібно вказати посилання на репозиторій.

**Таблиця вхідних даних для лабораторної роботи**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ Варіанта** | **Кількість предметів** | **Вага рюкзака W** | **Вектори ваг w[i]** | **Вектори цінностей v[i]** |
| 1 | 4 | 10 | [2, 3, 4, 5] | [3, 4, 5, 8] |
| 2 | 5 | 15 | [1, 3, 4, 5, 9] | [2, 4, 6, 7, 13] |
| 3 | 6 | 12 | [1, 2, 3, 2, 2, 1] | [2, 4, 5, 3, 7, 6] |
| 4 | 5 | 9 | [4, 3, 1, 2, 5] | [5, 4, 2, 2, 6] |
| 5 | 7 | 20 | [2, 4, 6, 3, 5, 7, 8] | [6, 10, 12, 7, 9, 14, 15] |
| 6 | 6 | 14 | [3, 5, 6, 4, 2, 3] | [6, 10, 12, 9, 4, 5] |
| 7 | 8 | 25 | [3, 2, 5, 7, 6, 4, 3, 1] | [5, 3, 8, 13, 11, 6, 7, 2] |
| 8 | 4 | 8 | [2, 3, 5, 2] | [3, 6, 9, 4] |
| 9 | 5 | 10 | [1, 2, 4, 6, 5] | [2, 5, 8, 13, 9] |
| 10 | 6 | 18 | [4, 3, 5, 6, 2, 4] | [8, 7, 10, 13, 5, 6] |
| 11 | 10 | 24 | [9, 3, 3, 3, 5, 8, 9] | [11, 10, 6, 4, 15, 3, 12] |
| 12 | 7 | 24 | [9, 3, 3, 3, 5, 8, 9] | [11, 10, 6, 4, 15, 3, 12] |
| 13 | 10 | 27 | [7, 4, 5, 10, 8, 5, 1, 7, 10, 7] | [11, 14, 11, 4, 2, 13, 7, 3, 13, 3] |
| 14 | 5 | 25 | [8, 1, 4, 7, 8] | [9, 9, 15, 9, 11] |
| 15 | 4 | 17 | [6, 5, 9, 3] | [8, 7, 14, 15] |
| 16 | 7 | 19 | [2, 10, 3, 8, 1, 2, 10] | [8, 5, 8, 7, 12, 5, 5] |
| 17 | 7 | 29 | [5, 6, 4, 6, 5, 6, 4] | [4, 8, 8, 13, 12, 11, 14] |
| 18 | 6 | 13 | [2, 7, 9, 6, 9, 7] | [5, 12, 12, 10, 15, 12] |
| 19 | 6 | 23 | [9, 9, 9, 7, 8, 9] | [9, 3, 15, 14, 8, 5] |
| 20 | 6 | 18 | [3, 10, 1, 4, 9, 7] | [7, 15, 15, 11, 12, 12] |
| 21 | 10 | 17 | [2, 3, 4, 7, 6, 7, 2, 5, 5, 6] | [14, 3, 14, 11, 4, 12, 9, 11, 9, 15] |
| 22 | 9 | 19 | [6, 8, 7, 1, 10, 9, 5, 3, 2] | [3, 7, 13, 4, 11, 13, 8, 10, 9] |
| 23 | 9 | 14 | [3, 5, 4, 2, 3, 9, 5, 6, 6] | [15, 15, 7, 7, 9, 9, 15, 6, 11] |
| 24 | 5 | 12 | [4, 10, 5, 4, 5] | [13, 7, 5, 9, 12] |
| 25 | 9 | 20 | [10, 7, 1, 6, 8, 1, 8, 6, 8] | [13, 13, 7, 7, 2, 14, 7, 2, 8] |
| 26 | 8 | 21 | [7, 3, 5, 2, 4, 4, 2, 4] | [12, 14, 3, 11, 9, 12, 11, 11] |
| 27 | 9 | 30 | [2, 2, 3, 5, 4, 10, 4, 4, 6] | [13, 3, 15, 6, 2, 12, 6, 5, 11] |
| 28 | 5 | 30 | [3, 9, 3, 6, 2] | [2, 10, 14, 8, 7] и |
| 29 | 4 | 16 | [2, 7, 9, 7] | [10, 4, 11, 9] |
| 30 | 8 | 15 | [10, 2, 9, 6, 3, 4, 8, 4] | [9, 4, 12, 2, 7, 15, 5, 5] |
| 31 | 6 | 29 | [2, 2, 7, 4, 9, 9] | [4, 7, 2, 8, 11, 13] |

**Хід роботи**

**Варіант-21**

**Приклад коду:**

import React, { useEffect, useState } from 'react';

const weights = [2, 3, 4, 7, 6, 7, 2, 5, 5, 6];

const values = [14, 3, 14, 11, 4, 12, 9, 11, 9, 15];

const capacity = 17;

const n = weights.length;

const App: React.FC = () => {

const [dpTable, setDpTable] = useState<number[][]>([]);

const [selectedItems, setSelectedItems] = useState<number[]>([]);

useEffect(() => {

const dp: number[][] = Array(n + 1)

.fill(0)

.map(() => Array(capacity + 1).fill(0));

for (let i = 1; i <= n; i++) {

for (let w = 0; w <= capacity; w++) {

if (weights[i - 1] > w) {

dp[i][w] = dp[i - 1][w];

} else {

dp[i][w] = Math.max(

dp[i - 1][w],

dp[i - 1][w - weights[i - 1]] + values[i - 1]

);

}

}

}

setDpTable(dp);

// Відновлення вибраних предметів

let w = capacity;

const items: number[] = [];

for (let i = n; i > 0 && w > 0; i--) {

if (dp[i][w] !== dp[i - 1][w]) {

items.push(i - 1);

w -= weights[i - 1];

}

}

setSelectedItems(items.reverse());

}, []);

return (

<div style={{ padding: '20px' }}>

<h1>Задача «Рюкзак» (Варіант 21)</h1>

<h2>Максимальна цінність: {dpTable[n]?.[capacity]}</h2>

<h3>Вибрані предмети (індекси з 0):</h3>

<ul>

{selectedItems.map((index) => (

<li key={index}>

Предмет {index + 1} — Вага: {weights[index]}, Цінність: {values[index]}

</li>

))}

</ul>

<h3>Таблиця dp[i][w]:</h3>

<div style={{ overflowX: 'auto' }}>

<table border={1} cellPadding={5}>

<thead>

<tr>

<th>i \ w</th>

{Array.from({ length: capacity + 1 }, (\_, w) => (

<th key={w}>{w}</th>

))}

</tr>

</thead>

<tbody>

{dpTable.map((row, i) => (

<tr key={i}>

<td>{i}</td>

{row.map((cell, w) => (

<td key={w} style={{ backgroundColor: selectedItems.includes(i - 1) ? '#d0f0c0' : '' }}>

{cell}

</td>

))}

</tr>

))}

</tbody>

</table>

</div>

</div>

);

};

export default App;

**Приклад роботи:**

****

**Висновки:**

У ході виконання лабораторної роботи було вивчено та реалізовано метод динамічного програмування для розв’язання задачі «Рюкзак». Було побудовано таблицю значень dp[i][w], яка дозволила знайти оптимальну цінність предметів, що можуть бути розміщені в рюкзаку з урахуванням заданих обмежень по вазі. Реалізація задачі на JavaScript з використанням React надала змогу наочно відобразити процес заповнення таблиці та етапи відновлення вибраного набору предметів. Візуалізація проміжних етапів дозволила краще зрозуміти принцип роботи алгоритму. Лабораторна робота сприяла набуттю навичок розв’язання задач оптимізації та практичному застосуванню теоретичних знань.