



CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

Expt. 10 PROJECT

Student Name: Tanay Manish Nesari

UID: 23BCS13761

Branch: BE-CSE

Section/Group: KRG 1-B

Semester: 5th

Date of Performance: 07/11/2025

Subject Name: DAA

Subject Code: 23CSH-301

Title: Project 3- Classic Challenges Solved: Dynamic Programming for Fibonacci, Coin Change, Knapsack, and Matrix Multiplication

Description: Implement dynamic programming solutions for classic problems like the Fibonacci sequence, coin change, knapsack problem, and matrix chain multiplication

Objective: Solve optimization problems efficiently by breaking them into smaller subproblems

Code:

```
import React, { useState, useMemo } from 'react';
import { ChevronDown, RefreshCcw, Zap, TrendingUp, Grid, Maximize } from 'lucide-react';
const solveFibonacci = (n) => {
  n = parseInt(n);
  if (isNaN(n) || n < 0 || n > 90) return "Please enter a non-negative number (max 90)";
  if (n <= 1) return n;
  let a = 0;
  let b = 1;
  let result = 1;
  for (let i = 2; i <= n; i++) {
    result = a + b;
    a = b;
    b = result;
  }
  return result;
};
const solveCoinChange = (coins, amount) => {
  amount = parseInt(amount);
  if (isNaN(amount) || amount < 0) return "Please enter a non-negative amount";
  if (amount === 0) return 0;

  const dp = new Array(amount + 1).fill(Infinity);
  dp[0] = 0;
  for (const coin of coins) {
    for (let j = coin; j <= amount; j++) {
```

```

        dp[j] = Math.min(dp[j], dp[j - coin] + 1);
    }
}

return dp[amount] === Infinity ? -1 : dp[amount];
};

const solveKnapsack = (W, weights, values, n) => {
    const dp = Array(n + 1).fill(0).map(() => Array(W + 1).fill(0));
    for (let i = 1; i <= n; i++) {
        for (let w = 1; w <= W; w++) {
            const currentWeight = weights[i - 1];
            const currentValue = values[i - 1];
            if (currentWeight <= w) {
                dp[i][w] = Math.max(currentValue + dp[i - 1][w - currentWeight], dp[i - 1][w]);
            } else {
                dp[i][w] = dp[i - 1][w];
            }
        }
    }
    return dp[n][W];
};

const solveMatrixChainMultiplication = (p) => {
    const n = p.length - 1;
    if (n < 1) return 0;
    const dp = Array(n).fill(0).map(() => Array(n).fill(0));
    for (let len = 2; len <= n; len++) {
        for (let i = 0; i <= n - len; i++) {
            const j = i + len - 1;
            dp[i][j] = Infinity;
            for (let k = i; k < j; k++) {
                const cost = dp[i][k] + dp[k + 1][j] + p[i] * p[k + 1] * p[j + 1];
                if (cost < dp[i][j]) {
                    dp[i][j] = cost;
                }
            }
        }
    }
    return dp[0][n - 1];
};

const problemOptions = [
    { id: 0, title: 'Select a Classic DP Challenge', icon: <ChevronDown size={20} /> },
    { id: 1, title: '1. Fibonacci Sequence (F(n))', icon: <TrendingUp size={20} />, placeholder: "Enter n (e.g., 15)" },
    { id: 2, title: '2. Coin Change (Min Coins)', icon: <Zap size={20} />, placeholder: "Enter target amount (e.g., 18)" },
    { id: 3, title: '3. 0/1 Knapsack Problem', icon: <Maximize size={20} /> },
    { id: 4, title: '4. Matrix Chain Multiplication', icon: <Grid size={20} /> },
];
const KNAPSACK_DATA = {
    W: 50,
    items: [
        { name: "Laptop", value: 60, weight: 10 },
        { name: "Book", value: 100, weight: 20 },
        { name: "Coffee Maker", value: 120, weight: 30 },
    ],
}

```

```

getValues: (items) => items.map(i => i.value),
getWeights: (items) => items.map(i => i.weight)
};

const COIN_SET = [1, 5, 10, 25];
const MATRIX_CHAIN_DIMS = [30, 35, 15, 5, 10, 20, 25];
const ProblemCard = ({ problem }) => (
  <div className="bg-gray-100 p-3 rounded-lg flex items-center shadow-inner">
    <div className="text-blue-600 mr-3">{problem.icon}</div>
    <p className="text-sm font-medium text-gray-700">{problem.title}</p>
  </div>
);

const ResultCard = ({ result, problemTitle, input }) => {
  let description = "";
  let detailedInput = null;
  if (problemTitle.includes('Fibonacci')) {
    description = `The ${input}-th number in the sequence is:`;
  } else if (problemTitle.includes('Coin Change')) {
    description = `The minimum number of coins needed for amount ${input} (using ${COIN_SET.join(',')}) is:`;
  } else if (problemTitle.includes('Knapsack')) {
    description = `The maximum total value that can be carried in a knapsack with capacity ${KNAPSACK_DATA.W} is:`;
    detailedInput = (
      <>
        <p className="text-xs font-semibold mt-2">Items Available:</p>
        <ul className="list-disc list-inside text-xs">
          {KNAPSACK_DATA.items.map((item, index) => (
            <li key={index}>{item.name} (Value: ${item.value}, Weight: {item.weight})</li>
          )));
        </ul>
      </>
    );
  } else if (problemTitle.includes('Matrix Chain')) {
    description = `The minimum number of scalar multiplications required for the chain (dimensions: ${MATRIX_CHAIN_DIMS.join(' x ')}) is:`;
  }
  return (
    <div className="w-full mt-6 p-6 bg-white border border-green-200 rounded-xl shadow-lg transition-transform duration-300 transform hover:shadow-xl">
      <h3 className="text-xl font-bold text-green-700 flex items-center mb-3">
        <RefreshCcw size={24} className="mr-2 animate-spin-slow" /> Solution Found
      </h3>
      <p className="text-sm text-gray-600 mb-4">{description}</p>
      {detailedInput}
      <div className="mt-4 p-3 bg-green-50 rounded-lg border-l-4 border-green-500">
        <p className="text-3xl font-extrabold text-green-800 break-all">
          {result}
        </p>
      </div>
    </div>
  );
};

export default function App() {
  const [selectedProblemId, setSelectedProblemId] = useState(0);

```

```

const [inputValue, setInputValue] = useState("");
const [result, setResult] = useState(null);
const [loading, setLoading] = useState(false);
const currentProblem = useMemo(() =>
  problemOptions.find(p => p.id === selectedProblemId) || problemOptions[0],
  [selectedProblemId]
);
const handleSolve = () => {
  if (selectedProblemId === 0) return;
  setLoading(true);
  setResult(null);
  setTimeout(() => {
    let calculatedResult;
    try {
      switch (selectedProblemId) {
        case 1:
          calculatedResult = solveFibonacci(inputValue);
          break;
        case 2:
          calculatedResult = solveCoinChange(COIN_SET, inputValue);
          break;
        case 3:
          calculatedResult = solveKnapsack(
            KNAPSACK_DATA.W,
            KNAPSACK_DATA.getWeights(KNAPSACK_DATA.items),
            KNAPSACK_DATA.getValues(KNAPSACK_DATA.items),
            KNAPSACK_DATA.items.length
          );
          break;
        case 4:
          calculatedResult = solveMatrixChainMultiplication(MATRIX_CHAIN_DIMS);
          break;
        default:
          calculatedResult = "Please select a valid problem.";
      }
    } catch (error) {
      calculatedResult = "An error occurred during calculation.";
      console.error(error);
    }
    setResult(calculatedResult);
    setLoading(false);
  }, 300);
};
const needsInput = selectedProblemId === 1 || selectedProblemId === 2;
return (
  <div className="min-h-screen bg-gray-50 p-4 sm:p-8 font-sans flex justify-center">
    <div className="w-full max-w-xl">
      <h1 className="text-3xl font-extrabold text-gray-900 mb-6 text-center">
        Classic DP Challenges Solver
      </h1>
      <p className="text-center text-sm text-gray-500 mb-8">
        Select one of the four core dynamic programming problems to solve instantly.
      </p>
      <div className="p-6 bg-white shadow-2xl rounded-2xl border border-blue-100">

```

```

<div className="mb-6">
  <label htmlFor="problem-select" className="block text-sm font-semibold text-gray-700 mb-2">
    1. Select Problem
  </label>
  <select
    id="problem-select"
    value={selectedProblemId}
    onChange={(e) => {
      setSelectedProblemId(parseInt(e.target.value));
      setInputValue("");
      setResult(null);
    }}
    className="w-full p-3 border border-gray-300 rounded-xl bg-white text-gray-800 focus:ring-blue-500 focus:border-blue-500 appearance-none transition duration-150 ease-in-out"
  >
    {problemOptions.map(opt => (
      <option key={opt.id} value={opt.id} disabled={opt.id === 0}>
        {opt.title.replace(/<[^>]*>?/gm, "")}
      </option>
    ))}
  </select>
</div>
{needsInput && (
  <div className="mb-6">
    <label htmlFor="input-value" className="block text-sm font-semibold text-gray-700 mb-2">
      2. Enter Input Value
    </label>
    <input
      id="input-value"
      type="number"
      value={inputValue}
      onChange={(e) => setInputValue(e.target.value)}
      placeholder={currentProblem.placeholder}
      className="w-full p-3 border border-gray-300 rounded-xl text-gray-800 focus:ring-blue-500 focus:border-blue-500 transition duration-150 ease-in-out"
    >
  </div>
)
  {!needsInput && selectedProblemId !== 0 && (
    <div className="mb-6">
      <label className="block text-sm font-semibold text-gray-700 mb-2">
        2. Fixed Data Input
      </label>
      <ProblemCard problem={currentProblem} />
    </div>
  )
  <button
    onClick={handleSolve}
    disabled={loading || selectedProblemId === 0 || (needsInput && !inputValue)}
    className={`w-full py-3 rounded-xl font-bold text-lg text-white transition duration-300 ease-in-out shadow-md
      ${loading || selectedProblemId === 0 || (needsInput && !inputValue)
      ? 'bg-blue-300 cursor-not-allowed'
    `}
  >
    Solve
  </button>
)
}

```

```

        : 'bg-blue-600 hover:bg-blue-700 hover:shadow-lg focus:outline-none focus:ring-4
focus:ring-blue-500 focus:ring-opacity-50'
      }
    >
      {loading ? 'Solving...' : 'Solve DP Problem'}
    </button>
  </div>
  {result !== null && (
    <ResultCard
      result={result.toLocaleString()}
      problemTitle={currentProblem.title}
      input={inputValue}
    />
  )}
  </div>
</div>
);
}

```

Classic DP Challenges Solver

Select one of the four core dynamic programming problems to solve instantly.

1. Select Problem
2. Enter Input Value

Solve DP Problem

 **Solution Found**

The 18-th number in the sequence is:

2,584