|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Competitive Programming Laboratory (216U01L401)** | **Semester:** | **IV** |
| **Date of Performance:** | **21 / 02 / 25** | **DIV/ Batch No:** | **C3** |
| **Student Name:** | **Om Satish Thanage** | **Roll No:** | **16010123217** |

**Experiment No: 2**

|  |
| --- |
| **Title: To implement a competitive programming problem on a platform (e.g., Leetcode) optimized using arrays, hash tables, and a divide-and-conquer approach.** |

|  |
| --- |
| **Aim and Objective of the Experiment:** |
| * Understand the concepts of array, hash tables, divide and conquer approach * Implement the code on leetcode or similar platform |

|  |
| --- |
| **COs to be achieved:** |
| **CO2:** **Analyse and optimize algorithms using amortized analysis and bit manipulation, equipping them to tackle complex computational problems.** |

|  |
| --- |
| **Books/ Journals/ Websites referred:** |
| 1. GeeksforGeeks 2. <https://www.youtube.com/watch?v=x-QkFoh-i5Y> 3. AOA lab (Amol Muley sir) learnt how to apply dynamic programming from sir |

|  |
| --- |
| **Theory:** |
| The **greedy approach** is an algorithmic paradigm that makes locally optimal choices at each step with the hope of finding a global optimum. It does not consider future consequences and never reconsiders past decisions. This method works well for problems where a **greedy choice property** (local optimum leads to a global optimum) and **optimal substructure** (optimal solution of a problem contains optimal solutions of subproblems) hold.  Common examples of greedy algorithms include:   1. **Activity Selection Problem** – Choosing the maximum number of non-overlapping activities. 2. **Huffman Coding** – Constructing an optimal prefix code for data compression. 3. **Kruskal’s and Prim’s Algorithm** – Finding the Minimum Spanning Tree (MST) in a graph. 4. **Dijkstra’s Algorithm** – Finding the shortest path from a single source in a weighted graph. 5. **Fractional Knapsack Problem** – Maximizing value by taking fractions of items with the highest value-to-weight ratio.   Greedy algorithms are efficient and often run in **O(n log n) or O(n)** time, but they do not always provide the optimal solution for every problem. When a problem does not satisfy the greedy choice property, dynamic programming or backtracking may be better alternatives. |

|  |
| --- |
| **Problem statement** |
| * 1. Use greedy algorithm and solve coin problem.   Examples for coin problems are : {9,6,5,1} make count of 11. |

|  |
| --- |
| **Code :** |
| Greedy Approach  #include <bits/stdc++.h>  using namespace std;  #define int long long  #define endl "\n"  const int MOD = 1e9 + 7;  const int INF = LLONG\_MAX >> 1;  signed main(){    cout<<"Enter the number of coins: \t";  int n; cin>>n;  vector<int> coins;  for(int i = 0; i<n; i++){  int x; cin>>x;  coins.push\_back(x);  }  cout<<"Enter the target amount: \t";  int amt; cin>>amt;  sort(coins.rbegin(), coins.rend());  int count = 0;  for(int i = 0; i < coins.size(); i++){  while (amt >= coins[i]) {  amt -= coins[i];  count++;  }  }  if(amt != 0){  cout<<"Solution not possible"<<endl;  return 0;  }  cout<<"Coins required:\t"<<count<<endl;  return 0;  }  Dynamic Programming Appraoch  #include <bits/stdc++.h>  using namespace std;  #define int long long  #define endl "\n"  const int MOD = 1e9 + 7;  const int INF = LLONG\_MAX >> 1;  signed main(){    cout<<"Enter the number of coins:\t";  int n; cin>>n;  vector<int> coins;  for(int i = 0; i<n; i++){  int x; cin>>x;  coins.push\_back(x);  }  cout<<"Enter the target amount:\t";  int amt; cin>>amt;  vector<int> min\_coins(amt+1, MOD);  min\_coins[0] = 0;  for(int i = 1; i<=amt; i++){  for(int j = 0; j<coins.size(); j++){  if(i - coins[j] >= 0){  min\_coins[i] = min(min\_coins[i], min\_coins[i - coins[j]] + 1);  }  }  }  cout<<"Minimum coins required to make "<<amt<<" is: "<<min\_coins[amt]<<endl;    return 0;  } |

|  |
| --- |
| **Output:** |
| Greedy Approach:    Dynamic Programming approach: |

|  |
| --- |
| **Post Lab Subjective/Objective type Questions:** |
|  |

|  |
| --- |
| **Conclusion:** |
| From this experiment we learnt how to solve the coins problem using greedy approach and to overcome its drawback we used dynamic programming to solve that issue. |