DEV JATIN BHAI PATEL

CE091

20CEUOS018

DAA LAB5

->Implement 0/1 Knapsack and Making change using Dynamic Programming

```cpp
#include <bits/stdc++.h>
using namespace std;


int main(){
    int p[5]={0,1,2,5,6};
    int w[5]={0,2,3,4,5};
    int capacity=8;
    int arr[sizeof(p)/sizeof(p[0])][capacity+1];
    for(int i=0;i<sizeof(p)/sizeof(p[0]);i++)
    {
        for(int j=0;j<=capacity;j++)
        {
            if(i==0||j==0)
                arr[i][j]=0;
            else if(j>=w[i])
            {
                arr[i][j]=max(arr[i-1][j] , p[i]+arr[i-1][j-w[i]]);
            }
            else
                arr[i][j]=arr[i-1][j];
            cout<<arr[i][j]<<" ";
        }cout<<"\n";
    }
    int sol[5];
    int i=4,j=8;
```

```
do
    {
        if (arr[i][j] != arr[i - 1][j])
        {
            sol[i] = 1;
            j = j - w[i];
            i--;
        }
        else
        {
            sol[i] = 0;
            i--;
        }
    } while (i > 0 && j > 0);
    if (arr[i][j] != arr[i - 1][j])
    {
        sol[i] = 1;
    }
    else
    {
        sol[i] = 0;
    }
    cout << "Final Solution Vector\n";
    for (int i = 1; i <= 4; i++)
        cout << sol[i] << " ";
    cout << endl;
    cout << "Profit = " << arr[4][8];
    return 0;
}
```

```
0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1
0 0 1 2 2 3 3 3 3
0 0 1 2 5 5 6 7 7
0 0 1 2 5 6 6 7 8
Final Solution Vector
0 1 0 1
Profit = 8
```

->Implement Making change using Dynamic Programming

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int d[4] = {0, 1, 4, 6};
    int n = sizeof(d) / sizeof(d[0]);
    int amount = 8;
    int arr[n-1+1][amount + 1];
    for (int i = 1; i <= n-1; i++)
    {
        arr[i][0] = 0;
    }
    for (int i = 0; i <= amount; i++)
        arr[0][i] = INT_MAX;

    for (int i = 1; i <= n-1; i++)
    {
        for (int j = 1; j <= amount; j++)
        {
            if (i == 1)
            {
                if (j < d[i])
                    arr[i][j] = 0;
                else
                    arr[i][j] = 1 + arr[i][j - d[1]];
            }
            else
            {
                if (j < d[i])
                    arr[i][j] = arr[i - 1][j];
                else
                    arr[i][j] = min(arr[i - 1][j], 1 + arr[i][j - d[i]]);
            }
        }
    }
```

```
for (int i = 1; i <= n-1; i++)
    {
        for (int j = 1; j <= amount; j++)
        {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
    cout << "coins = " << arr[n-1][amount] << endl;
    int sol[n-1 + 1] = {0};
    int i = n-1, j = amount;
    do
    {
        if (arr[i][j] == arr[i - 1][j])
        {
            if (sol[i] != 1)
                sol[i] = 0;
            i--;
        }
        else if (arr[i][j] == 1 + arr[i][j - d[i]])
        {
            sol[i] = 1;
            j -= d[i];
        }
    } while (i > 0 && j > 0);
    cout << "solution vector:" << endl;
    for (int i = 1; i <= n-1; i++)
        cout << d[i] << " ";
    cout << endl;
    for (int i = 1; i <= n-1; i++)
    {
        cout << sol[i] << " ";
    }
    cout << endl;
    return 0;
```

```
1 2 3 4 5 6 7 8
1 2 3 1 2 3 4 2
1 2 3 1 2 1 2 2
coins = 2
solution vector:
1 4 6
0 1 0
```