HW) You are given n objects the ith object has weight Wi and a target weight K. You have to tell the maximum value possible for any weight <=K
n<=1e5, k<=1e5
Sum of weights <=1e5

Q) Longest Increasing Subsequence
You are given an array of n integers find the longest strictly increasing subsequence of the array?
n<=2000
A[i]<=1e9

Input:
1 2 6 1 3 4
Output:
4

1	1
2	dp[2]=max(1, dp[1]+1)=2
6	3
1	1
3	3
4	4

```
Int dp[n+1], ans=0;
for(int i=1;i<=n;i++)
{
        dp[i]=1;
        for(int j=1;j<i;j++)
        {
                if(a[j]<a[i])
                dp[i]=max(dp[i], dp[j]+1);
        }
        ans=max(ans, dp[i]);
}
Return ans;
```

Q) Longest common subsequence
Given 2 strings a and b find the longest common subsequence between the 2.
|A| <= 2000
|B| <= 2000

Input

abcdef   dp[4][4] abcd ahyc
                              Abc ahyc   abcd ahy   abc ahy
Ahyczzyerf   i-1, j   i, j-1   i-1, j-1

Output:
4 (acef)

Dp[i][j] longest common subsequence of prefix of a consisting of i characters and prefix of b consisting of j characters.

dp[3][4]=2
abc   ahyc
ab   ahyc
abc ahy
dp[2][3]=1
Ab   ahy
dp[3][4]=max(dp[3][4], dp[2][3]+1)=2

dp[i][j]=max(dp[i-1][j], dp[i][j-1]);
abc ahyc
abcd ahyc
Dp[i-1][j-1]   a[i] == b[j]
dp[i][j]=max(dp[i][j], dp[i-1][j-1]+1);
dp[i][j]=dp[i-1][j-1]+1;

```
Int dp[n+1][m+1];
memset(dp, 0, sizeof(dp));
a=’#’+a;
b=’$’+b;
for(int i=1;i<=n;i++)
{
        for(int j=1;j<=m;j++)
        {
                dp[i][j]=max(dp[i-1][j], dp[i][j-1]);
                if(a[i]==b[j])
                dp[i][j]=max(dp[i][j], dp[i-1][j-1]+1);
        }
}
Int ans
```

Coin Change Problem

You have infinite coins of certain values available. Find the minimum number of coins required to get a sum of N. If it is impossible to get a sum of N, print -1.

1, 2, 5
9 = 5 + 2 + 2 = 3 coins

2, 4, 5
8 = 4 + 4
8 = 5 + 2 + 1
1 = -1
2 = 1
3 = -1
4 = 1
5 = 1
6 = 2
7 = 2
8 = 2

```
States - dp[i] = Minimum number of coins required to get a sum equal to i.
Transitions - for(int j=0;j<n;j++)
                dp[i] = min(dp[i],dp[i-a[j]]+1)
Base Case - dp[0] = 0
Goal - dp[N]
```

```
int dp[N+1];
for(int i=0;i<=N;i++)
{
        dp[i] = INF;
        for(int j=0;j<n;j++)
        {
                if(i-a[j]>=0)
                        dp[i] = min(dp[i],dp[i-a[j]]+1);
        }
}
if(dp[N]>=INF)
        cout << -1;
else
        cout << dp[N];
```

https://atcoder.jp/contests/dp/tasks/dp_n

```
States - dp[i][j] = Minimum cost of merging all slimes from a[i] to a[j] into a single slime
Transitions - for(int k=i;k<j;k++)
                dp[i][j] = min(dp[i][j],dp[i][k]+dp[k+1][j]+sum[i][j]);
Base Case - if(i==j) dp[i][j]=0
Goal - dp[1][n]
```

7 6 8 6 1 1
6 8 6 1
(6 8 6) + (1)
(6 8) + (6 1)
(6) + (8 6 1)

```
int mincost(int i,int j)
{
        if(i==j)
                return 0;
        if(dp[i][j]!=-1)
                return dp[i][j];
        dp[i][j]=INF;
        for(int k=i;k<j;k++)
                dp[i][j] = min(dp[i][j],mincost(i,k)+mincost(k+1,j)+sum[i][j]));
        return dp[i][j];
}
```

```
int main()
{
    int a[n];
    //Input
    memset(dp,-1,sizeof(dp))
    for(int i=0;i<n;i++)
    {
        sum[i][i]=a[i];
        for(int j=i+1;j<n;j++)
            sum[i][j] = sum[i][j-1]+a[j];
    }
    cout << mincost(0,n-1);
}
```

Time Complexity - $O(N^3)$