Coin change

total -> 11

coins-> 1,2,3,4,5 (infinite no. of each)

11-> (5,5,1), (5,4,2),.....

find the total number of ways?

base case->
ways to form 0 -> 1

-> we have formed amount 4 in 5 ways
-> we have one more coin '2'
-> how many ways will be there to form 6?

way 1+ 2-> 6
way 2+ 2-> 6
way 3+ 2-> 6
way 4+ 2-> 6
way 5+ 2-> 6

5 ways to form 6

//without permutation
n=4 and coins -> 3,2,1

0-> 1
1-> 0+1
2-> 1+1
3-> 1+0+2
4-> 0+1+3

```
//with permutation
n=4 and coins -> 1,2,3

0-> 1=1
1-> 0+1=1
2-> 0+1+1=2
3-> 0+1+1+2=4
4-> 0+1+2+4=7
```

Code for 'without permutation'

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

typedef long long ll;

int main(){
    ll k,n; cin>>k>>n;
    ll coins[n];
    for(ll i=0;i<n;i++){
        cin>>coins[i];
    }
    ll ways[k+1]={1};
    //1,3,2 only one way to iterate
    for(ll i=0;i<n;i++){
        for(ll j=0;j<=k-coins[i];j++){
            ways[j+coins[i]]+=ways[j];
        }
    }
    cout<<ways[k];
}

// coins array size * amount (n *k)
```

Code for with permutation

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

typedef long long ll;

int main(){
    ll k,n; cin>>k>>n;
    ll coins[n];
    for(ll i=0;i<n;i++){
        cin>>coins[i];
    }
    ll ways[k+1]={1};
    for(ll j=0;j<=k;j++){
        for(ll i=0;i<n;i++){
            if(j+coins[i]<=k){
                ways[j+coins[i]]+=ways[j];
            }
        }
    }
    cout<<ways[k];
}

// complexity -> coins array size * amount (n * k)
```

https://www.spoj.com/problems/TRT/

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

//dp[i][j] -> from start and end we are at ith and jth index
// indexes -> 1,2,3.....i........j,j+1,j+2.....n

int dp[2005][2005];
int func(int i, int j, int a[],int n)
{
    if(i==0 && j==n+1)
    {
        return 0;
    }
    if(dp[i][j]!=-1)
    return dp[i][j];
    int ans1=0;
    int ans2 = 0;
    if(i) // When we have come from i-1,j and we are taking the ith
element
    ans1 = func(i-1,j,a,n) + a[i] * (i+n-j+1);
    if(j<=n) // When we have come from i,j+1 and we are taking the jth
element
    ans2 = func(i,j+1,a,n) + a[j] * (i+n-j+1);
    dp[i][j] = max(ans1,ans2);
    return dp[i][j];
}


void solve()
{
   int n;
   cin>>n;
   int a[n+1];
   for(int i=1;i<=n;i++)
   cin>>a[i];
```

```
    int ans = 0;
    memset(dp,-1,sizeof(dp));
    // For the ending condition we must cover the cases when we have
covered all the elements.
    // i.e. i + (n-j+1) = n   or j - i = 1 or  j = i+1
    for(int i=0;i<=n;i++)
    ans = max(ans,func(i,i+1,a,n));
    cout<<ans<<"\n";
}
```

Matrix Chain Multiplication

A(n,m)

B(m,k)

C(k,p)

A*B -> n*m*k operations to multiply

Matrix Chain Multiplication
ABC -> (AB)C or A(BC)

(AB)C -> n*m*k+n*k*p
(n*k)

A(BC)-> m*k*p+n*m*p
m*p

AB -> n*m*k

ABCDE ->

0 1 2 3 4
p[ ]={1,2,3,4,5}

dp[i][j] = Minimum cost of multiplying matrices from i to j

dp[0][1]=0
dp[i][i+1]=0 for every possible i

i.....j
  k
M1(i,k)*M2(k,j)

ll dp[n][m];
if(dp[i][j]!=-1) // memoization

dp[i][j] = min (dp[i][k]+dp[k][j]+p[i]*p[k]*p[j]) for k>i && k<j

Code:

```
#include<bits/stdc++.h>
#define INF INT_MAX
using namespace std;

int dp[1001][1001];

int find(int l,int r,int p[]){
   if(l+1==r)
   return 0;

   if(dp[l][r]!=-1)
   return dp[l][r];

   dp[l][r]=INF;

   for(int k=l+1;k<r;k++){
       dp[l][r]=min(dp[l][r],find(l,k,p)+find(k,r,p)+p[l]*p[k]*p[r]);
   }
   return dp[l][r];
}

int main(){
```

```
    memset(dp,-1,sizeof(dp));
    // input p[]
    int n;
    cin>>n;

    int p[n];

    for(int i=0;i<n;i++)
    cin>>p[i];

    cout<<find(0,n-1,p);

}
```

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