Q. You are given N weights (1<=N<=1e5) with weights w[i] (1<=w[i]<=1e5). You have to determine if a given sum W is possible (1<=W<=1e5).

(sum of all weights is less than equal to1e5)

1, 2, 3, 4, .. n

n(n+1)/2 <= 1e5

N = approx sqrt(1e5) 500

```
map<int,int> count;
set<int> s;
for(int i=0;i<n;i++){
      count [w[i]]++;
      s.insert(w[i]);
}

w.clear();
for(auto i:s) w.push_back(s);

Dp[500][w+1];

// dp[i][j] denote minimum number of ith element required to
produce sum j, if it is not possible -INF;

int solve(int i, int j){
      if(j==0) return 0;
      if(i<0){
            return -INF;
      }
      if(dp[i][j]!=-1) return dp[i][j];
      if(solve(i-1, j)!=-INF) return dp[i][j] = 0;
      Int cnt = solve(i, j-w[i]);
      if(cnt==-INF) return dp[i][j] = -INF;
      if(cnt<count[w[i]]) return dp[i][j] = cnt+1;
      return dp[i][j] = -INF;
}
```

Bitmask DP/ Subset DP

Representation of sets using bits

{0, 1, 2, … 10}
{1, 3, 4, 8}

00100011010 => 2^8 + 2^4 + 2^3 + 2^1 => 282

Bitmask DP =>

- When the problem itself involves sets and subsets
- When the problem requires us to iterate over all the permutations but can be reducible to sets
  Lets say n = 20
  N! - 10^18
  2^n - 10^6

Spoj ASSIGN

```
int dp[N][1<<N];

dp[i][msk] -> denotes if I am on student i and the set bits of msk are the subjects that have already been assigned, then the number of ways for further assignment

int  solve(int i, int mask){

      if(i==n) return 1;

      if(dp[i][mask]!=-1) return dp[i][mask];

      dp[i][mask]=0;
      for(int j=0;j<n;j++){
            if(likes[i][j]==0||(mask&(1LL<<j))) continue;
            dp[i][mask] = dp[i][mask] + solve(i+1, mask|(1LL<<j));
      }
      return dp[i][mask];
}

Time Complexity : O((n^2) 2^n)
```

4*10^8
Answer: dp[0][0]

// Iterative Implementation

Dp[mask] denotes the number of ways to assign first setbits(mask) students using mask subjects

```
int dp[1<<N] = {0};

dp[0] = 1;
for(int mask=1;mask<(1<<N);mask++){
     int i = __builtin_popcount(mask);
     for(int j=0;j<n;j++){
          if(likes[i][j]==0||(mask&(1<<j))==0) continue;
          dp[mask] = (dp[mask] + dp[mask^(1<<j)]);
     }
}

cout<<dp[(1<<n)-1];


1<<3 => 8
1000

0000
0001
0010
0011
0100
0101
0110
0111
```

CSES Elevator Rides

People who are left
Sum of weights in the last ride

```
pair<int,int> dp[1<<N];

dp[0] = {0, inf};
for(int s = 1; s < (1<<N); s++ ){
      dp[s] = {N+1, 0};
      for(int p = 0; p < N; p++ ){
            if((s&(1<<p))==0) continue;
            auto prev = dp[s^(1<<p)];
            if(prev.second + w[p] <=x){
            // add p to the last ride only
                  dp[s] = min(dp[s], {prev.first, prev.second +
w[p]});
            }else{
            // create a new ride for p
                  dp[s] = min(dp[s], {prev.first+1, w[p]});
            }
      }
}
```

Time Complexity: O(n*2^n)

cout<<dp[(1<<n)-1].first;

Submask Enumeration

Mask => 100101101
Submask => 000100001

Submask is any number which satisfies
Mask|submask = mask

M = mask

```
for (int s=m; ; s=(s-1)&m) {
 ... you can use s ...
 if (s==0)  break;
}
```

For a mask with n setbits there will be 2^n submasks
0011010110
S = m 0011010000 => 0011001111 => 0011000110
S = s-1

Iterating over all masks with their submasks

```
for (int m=0; m<(1<<n); ++m)
    for (int s=m; s; s=(s-1)&m)
     ... s and m …
```

Number of masks with K setbits
= n C k
Complexity of the second loop for mask with K setbits
= 2^k

Total complexity for the code for masks with K setbits
= n C k 2^k

Total complexity
= Summation from k = 0 to k = n (n C k 2^k)

(1 + 2)^n

= O(3^n)

Atcoder DP - U Grouping

dp[mask] = if all the rabbits of this mask are grouped in some way then what is the maximum total score.

TRANSITION
dp[mask] = max(dp[mask], dp[mask^submask] + profit[submask]);

```
for(int mask=0;mask<(1<<n);mask++){
      profit[mask]=0;
      for(int i=0;i<n;i++){
            if((mask&(1<<i))==0) continue;
            for(int j=i+1;j<n;j++){
                  if(mask&(1<<j)) profit[mask]+=a[i][j];
            }
      }
}
```

```
for(int mask=0;mask<(1<<n);mask++){
      for(int s = m; ; s = (s-1)&mask){
            dp[mask] = max(dp[mask], dp[mask^s] + profit[s]);
            if(s==0) break;
      }
}
```

Answer dp[mask]

Time complexity O(n^2 * 2^n + 3^n)