АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ЕКСПЕДИЦИЯ

За да намерим минималния брой контейнери, те трябва да са с възможно по-голяма вместимост, затова първо е добре да сортираме в намаляващ ред на стойностите им наличните контейнери. След това започваме да опитваме по колко броя от всеки контейнер можем да използваме, за да получим оставащото тегло на багажа. Ако се получи равенство, процесът спира и това е минималния брой контейнери, който се търси. Ако не е достигнато точното тегло, процесът продължава. Задачата винаги има решение, защото имаме контейнер с вместимост 1.

Задачата може да се реши като се използва метода „backtracking" или стандартната задача за образуване на суми от динамичното оптимиране.

Решение чрез динамично оптимиране

Автор: Емил Келеведжиев

```
#include<iostream>
#include<algorithm>
using namespace std;
int s,n;
int a[101];
int t[100001][101];
int f(int s, int n)
{
  if(t[s][n]>0) return t[s][n];
  if(s==0) return 0;
  if(n==1) return s;
  int v=f(s,n-1);
  int r=s-a[n];
  int c=1;
  while(r>=0)
   {int vv=c+f(r,n-1);
    if(vv<v)v=vv;
    r -= a[n];
    c++;
   }
  t[s][n]=v;
  return v;
}
int main()
{
  cin >> s >> n;
  for(int i=1;i<=n;i++) cin >> a[i];
  sort(a+1,a+n+1);
  int r=f(s,n);
  cout << r << endl;
}
```

Решение чрез „backtracking".

Автор: Пламенка Христова

```
#include <iostream>
#include <stdlib.h>
using namespace std;
 int cmp(const void * a, const void * b)
    {
     return ( *(int*)b - *(int*)a );
```

```
    }
 int  s,n,i,p,w,k=1000000000,j;
 int A[100] ;
 void trs(int p,int T, int m)
 {
  int br;
  br=(s-T)/A[p];
  if (m+br<k) {
   if (T+br*A[p]==s)k=m+br;
             else if (p<n) while (br>=0)
             {
             trs(p+1, T+br*A[p],m+br);
             br--;
             }
   }
 }
 int main()
{
  cin >> s>>n;
  for(i=0;i<n;i++) cin >>A[i];
  qsort (A, n, sizeof( int), cmp);
  trs(0,0,0);
  cout <<k;
}
```