## Optimális pézváltás

Adott $n$ darab pénz; $\{p_1, \ldots, p_n\}$ és egy kifizetendő $E$ összeg. A lehető legkevesebb pénz összegeként akarjuk kifizetni (felváltani) az $E$ összeget. A pénzek tetszőleges pozítív egész szám értékűek lehetnek, nem csak a szokásos 1,2,5,10, stb.

### Feladat

Ijunk olyan programot, amely kiszámítja, hogy legkevesebb hány darab pénzzel lehet kifizetni az $E$ összeget, és meg is ad egy felváltást!

### Bemenet

A standard bemenet első sora két egész számot tartalmaz, a pénzek $n$ $(1 \leq n \leq 300)$ számát és a felváltandó $E$ $(1 \leq E \leq 10\,000)$ összeget. A második sor pontosan $n$ pozitív egész számot tartalmat egy-egy szóközzel elválasztva, a felváltáshoz használható pénzek értékeit. Egy pénz csak egyszer szerepelhet a felváltásban.

### Kimenet

A standard kimenet első sora azt a legkisebb $m$ számot tartalmazza, ahány darab pénzzel kifizethető az $E$ összeg.
A második sor tartalmazza az optimális kifizetésben szereplő pénzek sorszámait egy-egy szóközzel elválasztva, tetszőleges sorrendben. Több megoldás esetén bárnelyik megadható.
Ha az $E$ összeget nem lehet kifizetni, akkor az első és egyetlen sor a -1 számot tartalmazza.

### Példa

| Bemenet | Kimenet |
|---|---|
| `6 8`<br>`1 1 4 4 5 1 1` | `2`<br>`3 4` |

### Korlátok

Időlimit: 0.1 mp.
Memórilimit: 32 MiB
Pontozás: a tesztesetek 40%-ában $n < 1000$

---

## Megoldás

### 1. lépés: Az optimális megoldás szerkezetének elemzése

Tegyük fel, hogy

$$E = p_{i_1} + \ldots + p_{i_k},\ i_1 < \ldots < i_k$$

egy optimális megoldása a feladatnak. Ekkor

$$E - p_{i_k} = p_{i_1} + \ldots + p_{i_{k-1}}$$

optimális megoldása lesz annak a feladatnak, amelynek bemenete a felváltandó $E - p_{i_k}$ érték, és a felváltáshoz legfeljebb a első $i_k - 1$ $(p_1, \ldots, p_{i_k - 1})$ pénzeket használhatjuk. Ugyanis, ha lenne kevesebb pénzből álló felváltása $E - p_{i_k}$-nak, akkor $E$-nek is lenne $k$-nál kevesebb pénzből álló felváltása.

## 2. lépés: Részproblémkra és összetevőkre bontás

Minden $(X, i)$ $(1 \leq X \leq E, 1 \leq i \leq n)$ számpárra vegyük azt a részproblémát, hogy legkevesebb hány pénz összegeként lehet az $X$ értéket előállítani legfeljebb az első $i$ $\{p_1, \ldots, p_i\}$ pénz felhasználásával. Ha nincs megoldás, akkor legyen ez az érték $n+1$. Jelölje az $(X, i)$ részprobléma optimális megoldásának értékét $Opt(X, i)$. Definiáljuk az optimális megoldás értékét $X = 0$-ra és $i = 0$-ra is, azaz legyen $Opt(X, 0) = n+1$ és $Opt(0, i) = 0$.

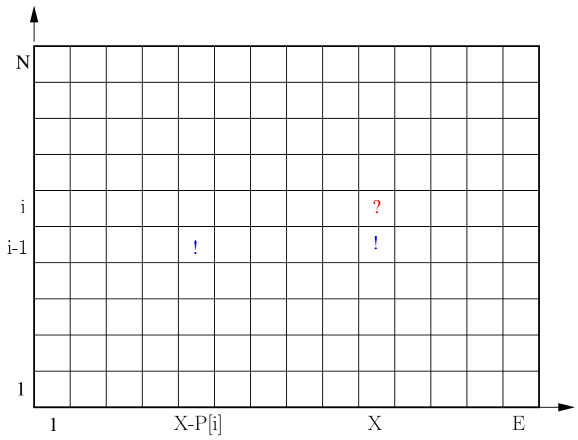
## 3. lépés: Részproblémák optimális megoldásának kifejezése (rekurzívan) az összetevők optimális megoldásaiból

$Opt(X, i)$-re az alábbi rekurzív összefüggés írható fel.

$$Opt(X,i) = \begin{cases} \infty & \text{ha} \quad i = 0 \wedge X > 0 \\ 0 & \text{ha} \quad X = 0 \\ Opt(X, i-1) & \text{ha} \quad X < p_i \\ \min(Opt(X, i-1), 1 + Opt(X - p_i, i-1)) & \text{ha} \quad X \geq p_i \end{cases} \tag{1}$$

A kindulási probléma megoldása $Opt(E, n)$.

## 4. lépés: Részproblémák optimális megoldásának kiszámítása alulról-felfelé haladva, táblázatkitöltéssel




1. ábra. Az optimális pénzválzás táblázata

Az $(X, i)$ részprobkéma összetevői, azaz azok a részproblémák, amelyektől függhet: $(X, i-1)$ és $X - p_i, i-1$. Tároljuk az $(X, i)$ részprobléma megoldását ($Opt(X, i)$ érékét) a táblázat $(X, i)$ koordinátájú elemében. Ekkor a részproblémák számítási sorrendje, azaz a táblázatkitöltés sorrendje soronként alulról felfelé, balról-jobbra.

## 5. lépés: Egy optimális megoldás előállítása a 4. lépésben kiszámított (és tárolt) információkból

Legyen $i$ a legkisebb olyan index, amelyre $Opt(E, i) = Opt(E, n)$. Tehát $E$ előállítható az első $i$ pénz felhasználásával $Opt(E, n)$ darab pénzzel, de az első $i-1$ pénzzel nem. Tehát az optimális megoldásban szerepel az $i$-edik pénz. Válaszzuk be a megoldásba, majd folytassuk az előállítást az $(E - p_i, i-1)$ részproblémára.

## Megvalósítás C++ nyelven

```cpp
#include <iostream>
using namespace std;

int main(){
   int n,E;
   cin>>n>>E;
   int P[n+1];
   for(int i=1;i<=n;i++)
      cin>>P[i];
   int Opt[E+1][n+1] ;
   for (int x=1;x<=E; x++) Opt[x][0]=n+1;   // 0. sor ki toltese
   for (int i=0;i<=n; i++) Opt[0][i]=0;     // 0. oszlop ki toltese
   for (int i=1; i<=n; i++){
      for (int x=1; x<=E; x++){
         Opt[x][i]=Opt[x][i-1];
         if (P[i]<=x && Opt[x][i]>Opt[x-P[i]][i-1]+1)
            Opt[x][i]=Opt[x-P[i]][i-1]+1;
      }
   }
//egy megoldas eloallitasa visszafejtessel
   if(Opt[E][n]<=n){
      int S[n+1];
      int m=0;
      int x=E; int i=n;
      do{
         while (i>1 && Opt[x][i]==Opt[x][i-1]) i--;
         S[++m]=i ;
         x-=P[i--];
      }while (x>0);
      cout<<m<<endl;
      for(int i=1;i<m;i++)
         cout<<S[i]<<" ";
      cout<<S[m]<<endl;
   }else{
      cout<<-1<<endl;
   }
 return 0;
 }
```