```
coins[0]:=0;
for i:=1 to S do
  min:=infinity;
  for j:=1 to m do
    if d[j]<=i and coins[i-d[j]]<min then
      min:=coins[i-d[j]];
  coins[i]:=1+min;

return coins[S];
```

Time: $\Theta(mS)$

```
coins[0]:=0;
for i:=1 to S do
  min:=infinity;
  for j:=1 to m do
    if d[j]<=i and coins[i-d[j]]<min then
      min:=coins[i-d[j]];
*     minchoice:=j;
  coins[i]:=1+min;
* choice[i]:=minchoice;
// ----------- recover solution --------
if coins[S]=infinity then write 'No solution!';
else
  while S>0 do
    write d[choice[S]];
    S:=S-d[choice[S]];
```

Dynamické programovanie—zhrnutie

1. **Určíme podproblém.**
   - aké sú rozmery matice, ktorú budeme vypĺňať?
   - aký je presný význam každého políčka matice?
   - kde v matici nájdeme riešenie pôvodnej úlohy?
2. **Vyriešime podproblém za pomoci iných podproblémov.**
   Ako vypočítame jedno políčko matice z iných políčiek matice?
3. **Bázové podproblémy.** Ktoré políčka nemožno vypočítať pomocou vzťahov z predchádzajúceho kroku? Aké hodnoty by mali obsahovať?
4. **Vyberieme poradie vypĺňania.** V akom poradí musíme maticu vypĺňať tak, aby sme v každom kroku mali vypočítané všetky políčka, ktoré potrebujeme na výpočet daného políčka?

Problém batohu

```
for j:=0 to W do
  V[0,j]:=0;
for i:=1 to n do
  for j:=1 to W do
    sol:=V[i-1,j];
    if (w[i]<=j) then
      othersol:=V[i-1,j-w[i]]+v[i];
      if (othersol>sol) then
        sol:=othersol;
    V[i,j]:=sol;
return V[n,W];
```

Problém batohu—zapamätanie riešenia

```
for j:=0 to W do
  V[0,j]:=0;
for i:=1 to n do
  for j:=1 to W do
    sol:=V[i-1,j];
*   take[i,j]:=false;
    if (w[i]<=j) then
      othersol:=V[i-1,j-w[i]]+v[i];
      if (othersol>sol) then
        sol:=othersol;
*       take[i,j]:=true;
    V[i,j]:=sol;
```

Problém batohu—vypísanie riešenia

```
// write the list of items to take
i:=n; j:=W;
while i>0 do
  if take[i,j] then
    write i
    j:=j-w[i]
  i:=i-1
```

Problém batohu—zjednodušený kód

```
for j:=0 to W do
* V[j]:=0;
for i:=1 to n do
* for j:=W downto 1 do
    if (w[i]<=j) then
*     othersol:=V[j-w[i]]+v[i];
*     if (othersol>V[j]) then
*       V[j]:=othersol;
return V[W];
```