Lodičky s utečencami

```
// base cases
N[0]:=0; N[1]=a[1]
// filling the array N
for i:=2 to n do
  if a[i]+N[i-2] > N[i-1] then
    N[i]:=a[i]+N[i-2]
  else
    N[i]:=N[i-1]

return N[n]
```

Lodičky s utečencami—vypísanie riešenia

```
// base cases
N[0]:=0; N[1]=a[1]; * keep[1]:=true
// filling the array N
for i:=2 to n do
  if a[i]+N[i-2] > N[i-1] then
    N[i]:=a[i]+N[i-2]
*   keep[i]:=true;
  else
    N[i]:=N[i-1]
*   keep[i]:=false;
// recover solution
i:=n;
while (i>0) do
  if (keep[i]) write 'land boat ',i; i:=i-2
  else i:=i-1;
```

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];
```