Príklad: Lodičky s utečencami

Úloha: Lodičky s utečencami prichádzajú do prístavu

i-ta lodička má na palube a_i utečencov

Z dvoch po sebe idúcich lodičiek môžeme akceptovať nanajvýš jednu

Ktoré lodičky vybrať aby sme zachránili najviac utečencov?

Príklady:

1 100 1 100

1 100 100 1

1 1 100 1 1 100 1 100

1 60 100 60

Idea: Budeme skladať riešenie "väčších" problémov z "menších" problémov.

Označenie: N_i: najväčší možný počet zachránených utečencov pre postupnosť lodičiek $a_1, a_2, \ldots, a_i$

Rozbor prípadov: Pozrime sa na i-tu lodičku:

- i-tu lodičku necháme pristáť: $N_i = a_i + N_{i-2}$
- i-tu lodičku vrátime späť: $N_i = N_{i-1}$

Platí teda: $N_i = \max\{a_i + N_{i-2}, N_{i-1}\}$
$(N_0 = 0, N_1 = a_1)$

```
// 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]
```

Časová zložitosť: $\Theta(n)$

Príklad:

```
a:        1    1   100   1   1  100   1   1000
N:  0
```

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?

Príklad: Problém batohu

Úloha: n objektov, batoh veľkosti W

i-ty objekt: váha w_i, hodnota v_i

Vezmi objekty **s najvyššou celkovo cenou**, tak aby si **neprekročil celkovú váhu** W

Príklad:

Celková váha: $W = 10$

Objekty (váha, hodnota): $(1, 4)$, $(5, 5)$, $(5, 5)$

Celková váha: $W = 9$

Objekty (váha, hodnota): $(1, 4)$, $(5, 5)$, $(5, 5)$

Riešenie: prvý pokus

Podproblém: V_j: maximálna hodnota objektov, ktoré sa zmestia do batohu veľkosti j

Riešenie bude hodnota V_W

Rekurencia: (vyskúšaj všetky možnosti prvého zobraného objektu)

$V_j = \max_{i:w_i \leq j}\{v_i + V_{j-w_i}\}$

Riešenie: druhý pokus

Podproblém: $V_{i,j}$: maximálna hodnota objektov **spomedzi objektov** $1, \ldots, i$, ktoré sa zmestia do batohu veľkosti j

Riešenie bude hodnota $V_{n,W}$

Rekurencia: Podľa toho, či sa rozhodneme použiť i-ty objekt

- použijeme i-ty objekt: $V_{i,j} = v_i + V_{i-1,j-w_i}$
- nepoužijeme i-ty objekt: $V_{i,j} = V_{i-1,j}$

Teda: $V_{i,j} = \max\{V_{i-1,j}, v_i + V_{i-1,j-w_i}\}$

Bázové podproblémy: $V_{0,j} = 0$

Poradie vypĺňanie: po riadkoch z ľavého horného rohu do pravého dolného rohu

Príklad: Celková váha: $W = 10$

Objekty (váha, hodnota): $(1, 4)$, $(5, 5)$, $(5, 5)$

$V_{i,j} = \max\{V_{i-1,j}, v_i + V_{i-1,j-w_i}\}$

	0	1	2	3	4	5	6	7	8	9	10
	0	0	0	0	0	0	0	0	0	0	0
(1,4)	0	4	4	4	4	4	4	4	4	4	4
(5,5)	0	4	4	4	4	5	9	9	9	9	9
(5,5)	0	4	4	4	4	5	9	9	9	9	10

```
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];
```

Časová zložitosť: $\Theta(nW)$

Pamäťová zložitosť: $\Theta(nW)$

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;
```

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
```

Príklad: Celková váha: $W = 10$

Objekty (váha, hodnota): $(1, 4)$, $(5, 5)$, $(5, 5)$

$V_{i,j} = \max\{V_{i-1,j}, v_i + V_{i-1,j-w_i}\}$

	0	1	2	3	4	5	6	7	8	9	10
	0	0	0	0	0	0	0	0	0	0	0
(1,4)	0	4	4	4	4	4	4	4	4	4	4
(5,5)	0	4	4	4	4	5	9	9	9	9	9
(5,5)	0	4	4	4	4	5	9	9	9	9	10

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];
```

Časová zložitosť: $\Theta(nW)$

Pamäťová zložitosť: $\Theta(W)$

Zhrnutie prednášky

- Technika dynamického programovania: riešenie problémov pomocou rozkladu na dobre definované podproblémy a "vypĺňanie matice"
- Príklad lodičiek s utečencami: jednorozmerná matica
- Problém batohu: dvojrozmerná matica