Dynamické programovanie

zhora nadol – memoizácia

zdola nahor

FIBONACCIHO ČÍSLA

$$f_0 = 0$$
$$f_1 = 1$$
$$f_n = f_{n-1} + f_{n-2}$$

function $f(n)$;

if n=0 **then**
    return 0
else
    if n=1 **then**
        return 1
    else
        return ($f(n-1) + f(n-2)$)

Rozdeľuj a panuj

- delí problémy na nezávislé podproblémy,
- ak sú podproblémy závislé, vedie to k veľkej časovej výpočtovej zložitosti,
- riešenie – uloženie už vypočítaných hodnôt do pamäte – **memoizácia**

Výpočet pre *n*=5

Rozdeľuj a panuj

```
f(5)=5
 |_f(4)=3
 |   |_f(3)=2
 |   |    |_f(2)=1
 |   |    |    |_f(1)=1
 |   |    |    |_f(0)=0
 |   |    |_f(1)=1
 |   |_f(2)=1
 |        |_f(1)=1
 |        |_f(0)=0
 |_f(3)=2
     |_f(2)=1
     |    |_f(1)=1
     |    |_f(0)=0
     |_f(1)=1
```

Dynamické programovanie zhora nadol

```
f(5)=5
|_f(4)=3
|   |_f(3)=2
|   |    |_f(2)=1
|   |    |    |_f(1)=1
|   |    |    |_f(0)=0
|   |    |_f(1)=1
|   |_f(2)=1
|_f(3)=2
```

Memoizácia

Zavedieme pole $Fib[0..n]$ inicializované na -1

```
function f(n);

if Fib[n]<0 then
    if n=0 then Fib[n] ← 0
    else
        if n=1 then Fib[n] ← 1
        else
            Fib[n] ← f(n – 1) + f(n – 2);
return Fib[n];
```

```
function f(n);

if n=0 then return 0
else
    if n=1 then return 1
    else return (f(n – 1) + f(n – 2))
```

Dynamické programovanie zdola nahor

Hodnoty v tabuľke sa vypočítavajú od menšieho rozmeru k väčšiemu:

$Fib[0] \leftarrow 0;$
$Fib[1] \leftarrow 1;$
for $i \leftarrow 2$ **to** $n - 1$ **do**
    $Fib[i] \leftarrow Fib[i-1]+Fib[i-2];$

$T(n) = 2 + n - 2 = n$
$S(n) = n$

Zníženie počtu zapamätaných hodnôt:

$Fib0 \leftarrow 0;$
$Fib1 \leftarrow 1;$
for $i \leftarrow 2$ **to** $n - 1$ **do**
    begin
        $Fib2 \leftarrow Fib0+Fib1;$
        $Fib0 \leftarrow Fib1;$
        $Fib1 \leftarrow Fib2;$
    end;

$T(n) = 2 + 3(n-2) = 3n - 4$
$S(n) = 3$

Kombinačné číslo

$$\binom{n}{0} = 1$$

$$\binom{n}{n} = 1$$

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

function *C*(*n*,*k*);

if *k*=0 **then** return 1
else
    if *n*=*k* **then** return 1
    else
        return *C*(*n*-1,*k*)+*C*(*n*-1,*k*-1);

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1

Výpočet pre n=5, k=2

Rozdeľuj a panuj

```
C(5,2)=10
 |_C(4,2)=6
 |   |_C(3,2)=3
 |   |   |_C(2,2)=1
 |   |   |_C(2,1)=2
 |   |     |_C(1,1)=1
 |   |     |_C(1,0)=1
 |   |_C(3,1)=3
 |       |_C(2,1)=2
 |       | |_C(1,1)=1
 |       | |_C(1,0)=1
 |       |_C(2,0)=1
 |_C(4,1)=4
     |_C(3,1)=3
     |   |_C(2,1)=2
     |   |   |_C(1,1)=1
     |   |   |_C(1,0)=1
     |   |_C(2,0)=1
     |_C(3,0)=1
```

DP zhora nadol

```
C(5,2)=10
 |_C(4,2)=6
 |   |_C(3,2)=3
 |   |   |_C(2,2)=1
 |   |   |_C(2,1)=2
 |   |     |_C(1,1)=1
 |   |     |_C(1,0)=1
 |   |_C(3,1)=3
 |     |_C(2,1)=2
 |     |_C(2,0)=1
 |_C(4,1)=4
   |_C(3,1)=3
   |_C(3,0)=1
```

Dynamické programovanie zhora nadol

- pole $Comb[0..n,0..k]$ je inicializované na -1

function $C(n,k)$;

if $Comb[n,k]<0$ **then**
    if $k=0$ **then** $Comb[n,k] \leftarrow 1$
    else
        if $n=k$ **then** $Comb[n,k] \leftarrow 1$
        else
            $Comb[n,k] \leftarrow C(n-1,k)+C(n-1,k-1)$
return $Comb[n,k]$

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1

function $C(n,k)$;

if $k=0$ **then** return 1
else
    if $n=k$ **then** return 1
    else
        return $C(n-1,k)+C(n-1,k-1)$;

$n=5,\ k=2$

	0	1	2
0	-1	-1	-1
1	1	1	-1
2	1	2	1
3	1	3	3
4	-1	4	6
5	-1	-1	10

Dynamické programovanie zdola nahor

for $r \leftarrow 1$ **to** n-k **do**
  $Comb$[r,0]:=1;
for $s \leftarrow 1$ **to** k **do**
  begin
    $Comb$[s,s] $\leftarrow$ 1;
    for $r \leftarrow s$+1 **to** n-k+s **do**
      $Comb$[r,s] $\leftarrow$ $Comb$[r-1,s]+$Comb$[r-1,s-1];
  end;

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1

n=5, k=2

	0	1	2
0			
1	1	1	
2	1	2	1
3	1	3	3
4		4	6
5			10

n=5, k=3

	0	1	2	3
0				
1	1	1		
2	1	2	1	
3		3	3	1
4			6	4
5				10

$\mathrm{T}(n,k) = n - k + k\ (n - k + 1)$
$\mathrm{S}(n,k) = n\ (k + 1)$

Úspornejšie využitie poľa

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1

for $r \leftarrow 0$ **to** n-k **do**
    $Comb[r,0] \leftarrow 1;$
for $s \leftarrow 1$ **to** k **do**
    begin
        $Comb[0,s] \leftarrow 1;$
        for $r \leftarrow 1$ **to** n-k **do**
            $Comb[r,s] \leftarrow Comb[r-1,s]+Comb[r,s-1];$
    end;

$\mathrm{T}(n,k) = n - k + k\,(n - k + 1)$
$\mathrm{S}(n,k) = (n - k + 1)(k + 1)$

n=5, k=3

	0	1	2	3
0		1	1	1
1	1	2	3	4
2	1	3	6	10

Problém batoha

- Majme n predmetov s hmotnosťami $h_1, h_2, \ldots, h_n$ a cenami $c_1, c_2, \ldots, c_n$. Nájdime množinu predmetov s najväčším súčtom cien, ktoré sa dajú odniesť v batohu s nosnosťou $Hmax$.
- Príklad: $Hmax = 10$

i	h_i	c_i
1	6	30
2	3	14
3	2	16
4	4	9

- Riešenie: predmety 1, 3

Problém batoha

- Funkcia *batoh*(H,j) vyjadruje maximálnu hodnotu, ktorá sa dá odniesť v batohu s hmotnosťou H z predmetov 1, 2, ..., j.
- Hľadáme hodnotu *batoh*($Hmax$,n).
- rozložíme na podproblémy:
  - určíme maximálnu cenu obsahu batoha, ak vezmeme j-ty predmet (cenu zvýšime o c_j, nosnosť batoha znížime o h_j): *batoh*(H-h_j, j-1)+ c_j
  - ak nevezmeme j-ty predmet (cena zostáva, nosnosť zostáva, počet predmetov sa znižuje): *batoh*(H, j-1)
- a zložíme výsledok:
  - určíme maximum z týchto dvoch možností

PROBLÉM BATOHA

- ak sa j-ty predmet nezmestí do batoha, hľadáme riešenie pomocou predmetov 1, 2, ..., j-1: $batoh(H, j) = batoh(H, j-1)$
- ak sa j-ty predmet zmestí do batoha, rozložíme problém na dva podproblémy:
  - určíme maximálnu cenu obsahu batoha, ak vezmeme j-ty predmet (cenu zvýšime o c_j, nosnosť batoha znížime o h_j): $batoh(H-h_j, j-1) + c_j$
  - ak nevezmeme j-ty predmet (cena zostáva, nosnosť zostáva, počet predmetov sa znižuje): $batoh(H, j-1)$
- a zložíme výsledok:
  - určíme maximum z týchto dvoch možností
- triviálny prípad:
  - ak j=0, $batoh(H, j)=0$

Problém batoha

$batoh(H, j) = 0$ ak j=0

$batoh(H, j) = batoh(H, j\text{-}1)$, ak $h_j>H$

$batoh(H, j) = \max\{batoh(H\text{-}h_j, j\text{-}1)+ c_j, batoh(H, j\text{-}1)\}$ inak

function *batoh*(*H*, *j*);

if *j* = 0 **then** return 0
else
    if *h*[*j*] > *H* **then** return *batoh*(*H*,*j*-1)
    else
        begin
            b1 ← *batoh*(*H*-*h*[*j*],*j*-1)+*c*[*j*];
            b2 ← *batoh*(*H*,*j*-1);
            if *b1* > *b2* **then** return *b1*
            else return *b2*
        end;

Problém batoha – DP zhora nadol

- čiastkové riešenia si ukladáme do tabuľky $B[0..Hmax,0..n]$, ktorej hodnoty inicializujeme na -1

function *batoh*(H, j);

if $B[H,j]<0$ **then**
    begin
        if $j=0$ **then** $B[H,j] \leftarrow 0$
        else
            if $h[j]>H$ **then** $B[H,j] \leftarrow batoh(H,j\text{-}1)$
            else
                begin
                    $b1 \leftarrow batoh(H\text{-}h[j],j\text{-}1)+c[j]$;
                    $b2 \leftarrow batoh(H,j\text{-}1)$;
                    if $b1>b2$ **then** $B[H,j] \leftarrow b1$
                    else $B[H,j] \leftarrow b2$
                end
    end;
return $B[H,j]$;

VÝPOČET TABUĽKY ZDOLA NAHOR

Vstup:

i	h_i	c_i
1	6	30
2	3	18
3	2	17
4	4	14

Funkcia na výpočet ceny:

$batoh(H, j) = 0$ ak j=0

$batoh(H, j) = batoh(H, j\text{-}1)$, ak h_j>H

inak

$batoh(H, j) =$

$\max\{batoh(H\text{-}h_j, j\text{-}1)+ c_j, batoh(H, j\text{-}1)\}$

	0	*1*	*2*	*3*	*4*
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	17	17
3	0	0	18	18	18
4	0	0	18	18	18
5	0	0	18	35	35
6	0	30	30	35	35
7	0	30	30	35	35
8	0	30	30	47	47
9	0	30	48	48	49
10	0	30	48	48	49

Problém batoha – dynamické programovanie zdola nahor

- Tabuľku B vypočítame nerekurzívne zdola od menšieho rozmeru úlohy (počet predmetov) k väčšiemu, výsledok je $B[Hmax,n]$:

for $H \leftarrow 0$ **to** $Hmax$ **do** $B[H,0] \leftarrow 0$;

for $j \leftarrow 1$ **to** n **do**

  for $H \leftarrow 0$ **to** $Hmax$ **do**

    if $h[j]>H$ **then** $B[H,j] \leftarrow B[H,j\text{-}1]$

    else

      begin

        $b1 \leftarrow B[H\text{-}h[j],j\text{-}1]+c[j]$;

        $b2 \leftarrow B[H,j\text{-}1]$;

        if $b1 > b2$ **then** $B[H,j] \leftarrow b1$

        else $B[H,j] \leftarrow b2$;

      end;

Porovnanie výpočtu tabuľky zhora a zdola

zhora nadol

0	-1	-1	-1	-1
0	0	-1	-1	-1
0	-1	-1	-1	-1
0	0	-1	-1	-1
0	0	18	-1	-1
0	0	-1	-1	-1
0	30	30	35	-1
0	30	-1	-1	-1
0	30	30	-1	-1
-1	-1	-1	-1	-1
0	30	48	48	49

zdola nahor

0	0	0	0	0
0	0	0	0	0
0	0	0	17	17
0	0	18	18	18
0	0	18	18	18
0	0	18	35	35
0	30	30	35	35
0	30	30	35	35
0	30	30	47	47
0	30	48	48	49
0	30	48	48	49

Výpočet predmetov vložených do batoha

- z tabuľky B: ak nastala zmena medzi hodnotami v susedných stĺpcoch tabuľky j, j-1, do batoha sa vkladal predmet j

for $j \leftarrow n$ **downto** 1 **do**
  if $B[H,j] > B[H,j\text{-}1]$ **then**
    begin
      $H \leftarrow H$ - $h[j]$;
      write(j)
    end;

i	h_i	c_i
1	6	30
2	3	18
3	2	17
4	4	14

	0	*1*	*2*	*3*	*4*
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	17	17
3	0	0	18	18	18
4	0	0	18	18	18
5	0	0	18	35	35
6	0	30	30	35	35
7	0	30	30	35	35
8	0	30	30	47	47
9	0	30	48	48	49
10	0	30	48	48	49

Určme predmety v batohu

for $j \leftarrow n$ **downto** 1 **do**
  if $B[H,j] > B[H,j\text{-}1]$ **then**
    begin
      $H \leftarrow H$ - $h[j]$;
      write(j)
    end;

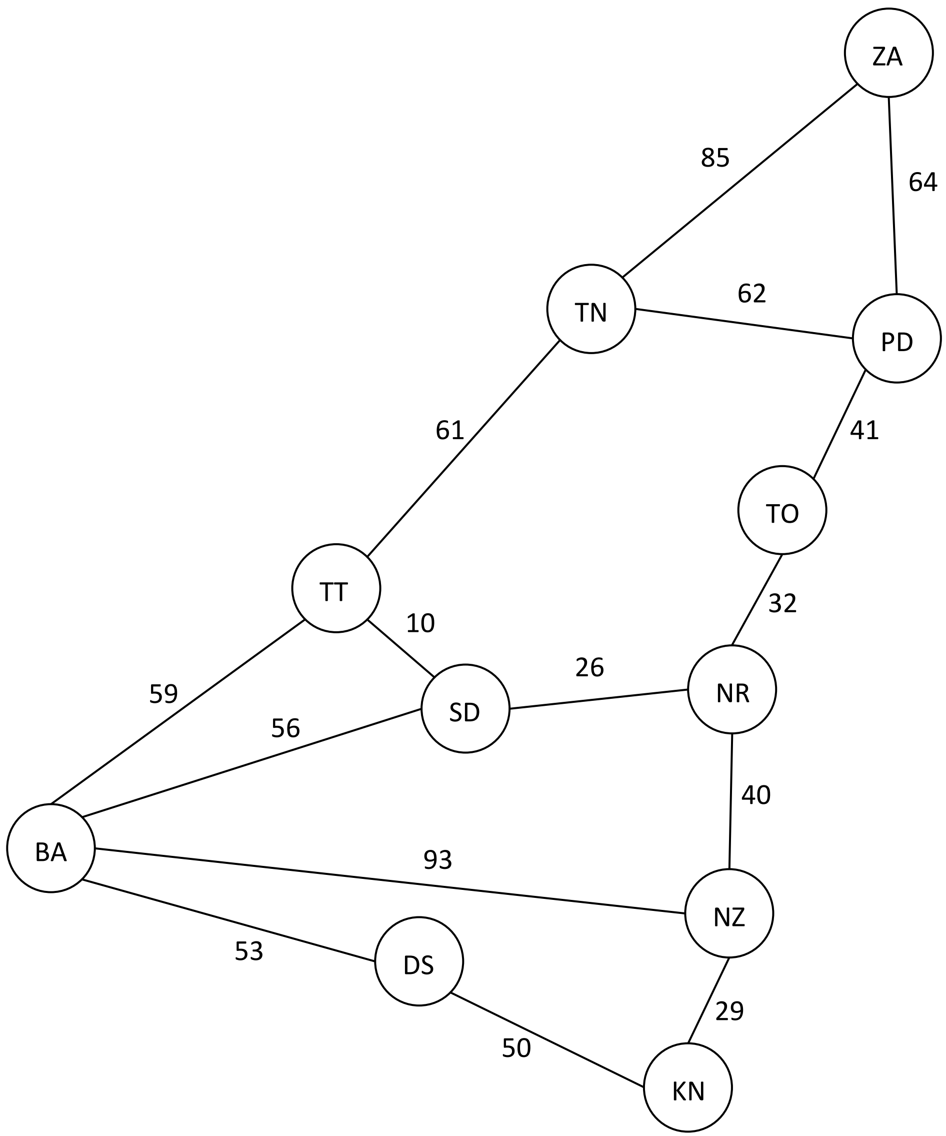
```
cislo:        1    2    3    4
hmotnost:     6    2    3    4
cena:        30   14   16    9

nosnost batoha: 10

Cena batoha je 46

    0    0    0    0    0
    0    0    0    0    0
    0    0   14   14   14
    0    0   14   16   16
    0    0   14   16   16
    0    0   14   30   30
    0   30   30   30   30
    0   30   30   30   30
    0   30   44   44   44
    0   30   44   46   46
    0   30   44   46   46
```

```
cislo:        1    2    3    4
hmotnost:     6    2    3    4
cena:        30   14   16    9

nosnost batoha: 10

Cena batoha je 46

    0   -1   -1   -1   -1
    0    0   -1   -1   -1
    0   -1   -1   -1   -1
    0    0   14   -1   -1
    0    0   -1   -1   -1
    0    0   -1   -1   -1
    0   30   30   30   -1
    0   30   30   -1   -1
    0   30   -1   -1   -1
   -1   -1   -1   -1   -1
    0   30   44   46   46
```

Problém batoha – výpočtová zložitosť

- n je počet predmetov
- časová výpočtová zložitosť:
  - vyplnenie tabuľky B na zistenie najvyššej ceny: $(n+1).(Hmax+1)$
  - zistenie predmetov vložených do batoha: n
  - $T(n) = O(n.Hmax)$
- pamäťová výpočtová zložitosť:
  - veľkosť tabuľky B: $(n+1).(Hmax+1)$
  - $S(n) = O(n.Hmax)$

Hľadanie najkratšej cesty v grafe



Hľadanie najkratšej cesty v grafe – dynamické programovanie zdola nahor

Majme graf cestnej siete daný incidenčnou maticou. Vypočítajme najkratšiu cestu z mesta *z* do mesta *k*.

	BA	DS	KN	NZ	NR	SD	TT	TO	TN	PD	ZA
BA	0	53	1000	93	1000	56	59	1000	1000	1000	1000
DS	53	0	50	1000	1000	1000	1000	1000	1000	1000	1000
KN	1000	50	0	29	1000	1000	1000	1000	1000	1000	1000
NZ	93	1000	29	0	40	1000	1000	1000	1000	1000	1000
NR	1000	1000	1000	40	0	26	1000	32	1000	1000	1000
SD	56	1000	1000	1000	26	0	10	1000	1000	1000	1000
TT	59	1000	1000	1000	1000	10	0	1000	61	1000	1000
TO	1000	1000	1000	1000	32	1000	1000	0	1000	41	1000
TN	1000	1000	1000	1000	1000	1000	61	1000	0	62	85
PD	1000	1000	1000	1000	1000	1000	1000	41	62	0	64
ZA	1000	1000	1000	1000	1000	1000	1000	1000	85	64	0

Hľadanie najkratšej cesty v grafe

BA	DS	KN	NZ	NR	SD	TT	TO	TN	PD	ZA
59	1000	1000	1000	1000	10	0	1000	61	1000	1000
59	1000	1000	1000	36	10	0	1000	61	1000	1000
59	1000	1000	76	36	10	0	68	61	1000	1000
59	112	1000	76	36	10	0	68	61	1000	1000
59	112	1000	76	36	10	0	68	61	123	146
59	112	1000	76	36	10	0	68	61	109	146
59	112	105	76	36	10	0	68	61	109	146
59	112	105	76	36	10	0	68	61	109	146
59	112	105	76	36	10	0	68	61	109	146
59	112	105	76	36	10	0	68	61	109	146
59	112	105	76	36	10	0	68	61	109	146

7	7	7	7	7	7	7	7	7	7	7
7	7	7	7	6	7	7	7	7	7	7
7	7	7	5	6	7	7	5	7	7	7
7	1	7	5	6	7	7	5	7	7	7
7	1	7	5	6	7	7	5	7	9	9
7	1	7	5	6	7	7	5	7	8	9
7	1	4	5	6	7	7	5	7	8	9
7	1	4	5	6	7	7	5	7	8	9
7	1	4	5	6	7	7	5	7	8	9
7	1	4	5	6	7	7	5	7	8	9
7	1	4	5	6	7	7	5	7	8	9
1-BA	2-DS	3-KN	4-NZ	5-NR	6-SD	7-TT	8-TO	9-TN	10-PD	11-ZA

Hľadanie najkratšej cesty v grafe

z – index začiatočného mesta
k – index koncového mesta
$c[1..n,1..n]$ – incidenčná matica grafu
$d[1..n]$ – pole najkratších vzdialeností zo začiatočného mesta
$p[1..n]$ – pole predchodcov miest na najkratšej ceste
S – množina miest, do ktorých je najkratšia cesta známa

Inicializácia premenných:

z, k, c – vstupy

	BA	DS	KN	NZ	NR	SD	TT	TO	TN	PD	ZA
BA	0	53	1000	93	1000	56	59	1000	1000	1000	1000
DS	53	0	50	1000	1000	1000	1000	1000	1000	1000	1000
KN	1000	50	0	29	1000	1000	1000	1000	1000	1000	1000
NZ	93	1000	29	0	40	1000	1000	1000	1000	1000	1000
NR	1000	1000	1000	40	0	26	1000	32	1000	1000	1000
SD	56	1000	1000	1000	26	0	10	1000	1000	1000	1000
TT	59	1000	1000	1000	1000	10	0	1000	61	1000	1000
TO	1000	1000	1000	1000	32	1000	1000	0	1000	41	1000
TN	1000	1000	1000	1000	1000	1000	61	1000	0	62	85
PD	1000	1000	1000	1000	1000	1000	1000	41	62	0	64
ZA	1000	1000	1000	1000	1000	1000	1000	1000	85	64	0

$d[i]$ – inicializuje sa na priame vzdialenosti z mesta z do i

for $i \leftarrow 1$ **to** n **do** $d[i] \leftarrow c[z,i]$;

59	1000	1000	1000	1000	10	0	1000	61	1000	1000

$p[i]$ – inicializuje sa na začiatočné mesto ako predchodcu koncového mesta na priamej ceste

for $i \leftarrow 1$ **to** n **do** $p[i] \leftarrow z$;

7	7	7	7	7	7	7	7	7	7	7

S – najkratšiu cestu poznáme do začiatočného mesta

$S \leftarrow [z]$;

Výpočet dĺžky najkratších ciest:

```
for i ← 2 to n-1 do begin
    m ← 0;                             //hľadanie mesta m s minimálnou
    for j ← 1 to n do                  //vzdialenosťou do mesta z
        if not (j in s) and (d[j]<d[m])
        then m ← j;
    S ← S+[m];                         //pridanie mesta m do množiny S
    for j ← 1 to n do begin
        if not (j in S) then
            if d[j]>d[m]+c[m,j] then begin
                p[j] ← m;                  //úprava trasy cez m
                d[j] ← d[m]+c[m,j]         //úprava najkratších vzd. cez m
            end;
    end;
end;
```

BA	DS	KN	NZ	NR	SD	TT	TO	TN	PD	ZA
59	1000	1000	1000	1000	10	0	1000	61	1000	1000
59	1000	1000	1000	36	10	0	1000	61	1000	1000
59	1000	1000	76	36	10	0	68	61	1000	1000
59	112	1000	76	36	10	0	68	61	1000	1000
59	112	1000	76	36	10	0	68	61	123	146
59	112	1000	76	36	10	0	68	61	109	146
59	112	105	76	36	10	0	68	61	109	146
59	112	105	76	36	10	0	68	61	109	146
59	112	105	76	36	10	0	68	61	109	146
59	112	105	76	36	10	0	68	61	109	146
59	112	105	76	36	10	0	68	61	109	146

Výpočet trasy najkratšej cesty (zoznam t)

$i \leftarrow k$;
$t \leftarrow (k)$;
while $i \neq z$ **do begin**
    $i \leftarrow p[i]$;
    $t \leftarrow$ vložprvý(i, t);
end;

7	1	4	5	6	7	7	5	7	8	9
1-BA	2-DS	3-KN	4-NZ	5-NR	6-SD	7-TT	8-TO	9-TN	10-PD	11-ZA

Výpočtová zložitosť hľadania najkratšej cesty

- hľadanie mesta *min*:

$$n-1+n-2+\cdots+2=\frac{(n+1)(n-2)}{2}$$

- úprava najkratších vzdialeností v poli *d*:

$$n-2+n-3+\cdots+1=\frac{(n-1)(n-2)}{2}$$

- spolu porovnaní

$$T(n)=\frac{(n+1)(n-2)+(n-1)(n-2)}{2}=\frac{2n(n-2)}{2}=n(n-2)=O(n^2)$$

- pamäť – matica *c*, polia *d*, *p*, množina *s*:

$$S(n)=n^2+n+n+n=O(n^2)$$