Algoritmické strategie

Dynamické programování

Úvod do optimalizace

- Určete tři čísla z uvedené množiny, jejichž součin je největší
  [-5, -1, -0,5, 10, 2]

Úvod do optimalizace

- Najděte "nejlepší" řešení nějakého problému.
- Existuje mnoho platných - zde jakákoliv trojice čísel je řešením

- Najděte řešení, které má maximální nebo minimální hodnotu – **efektivně**
- Budeme vyhledávat v prostoru (částečných, parciálních) řešení.

Dynamické programování

- Řešení tohoto problému se skládá z řešení dílčích problémů
- Podobně jako rozděl a panuj
- Podproblémy jsou často využity opakovaně.
- **Ukládá řešení podproblémů**

Fibonacciho čísla

Připomeňme si Fibonacciho poslopnost

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . .

Rekurzivně:

$Fib(1) = 1$

$Fib(2) = 1$

$Fib(n) = Fib(n - 1) + Fib(n - 2)$

Rekurzivní výpočet členu posloupnosti Fib(i)

Rekurzivní výpočet členu posloupnosti

Fib(i):

if i = 1 **or** i = 2 **then**

  return 1

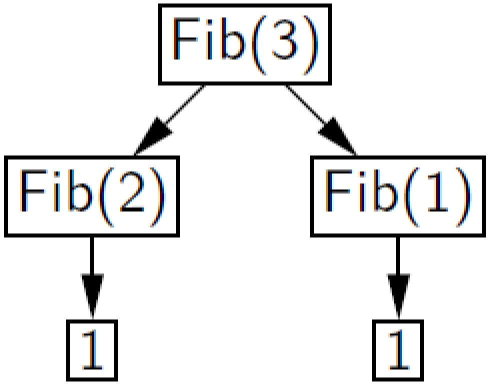
end if

return Fib(n-1) + Fib(n-2)

Běh programu

- Kolikrát je Fib(n) volán?

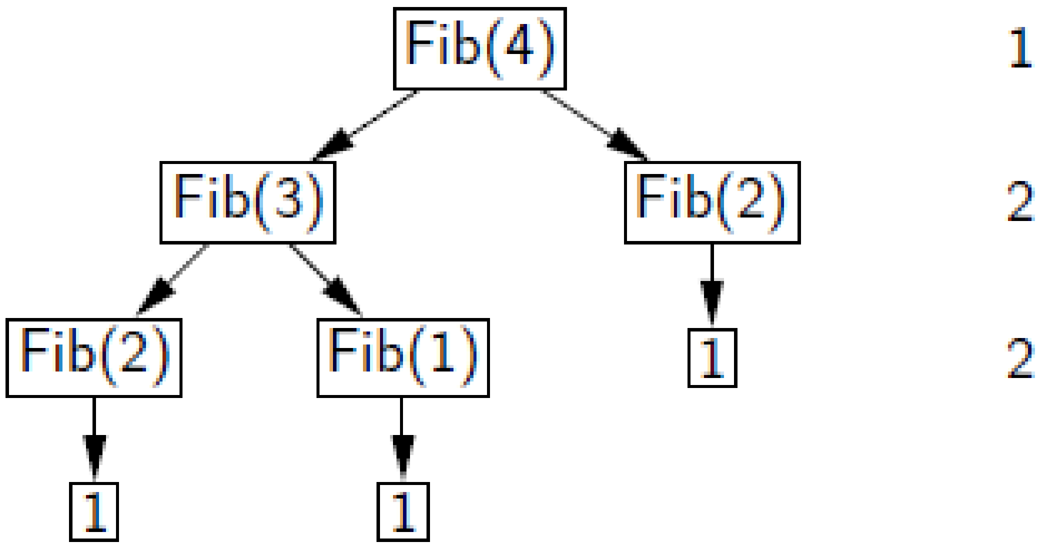
Fib(3)



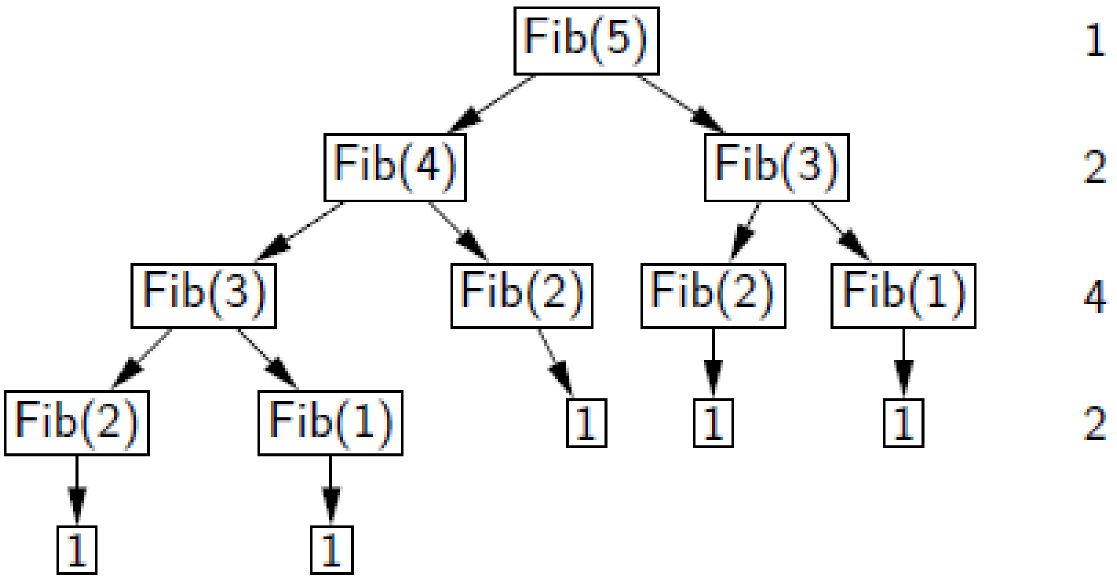
1

2

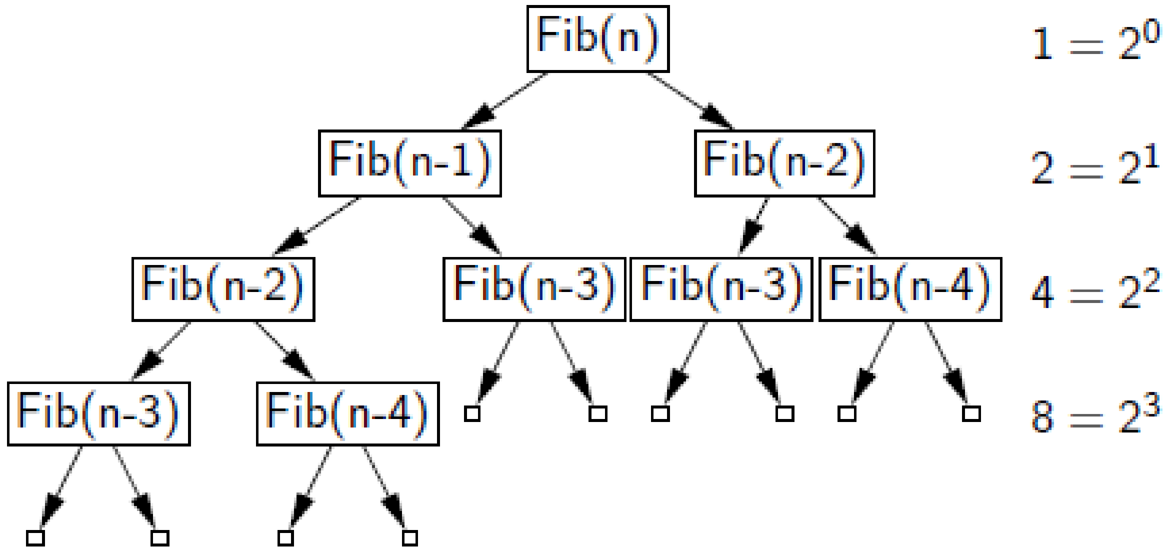
Fib(4)



Fib(5)



Fib(n)


Fib(n)
Fib(n-1)
Fib(n-2)
Fib(n-2)
Fib(n-3)
Fib(n-3)
Fib(n-4)
Fib(n-3)
Fib(n-4)
$1 = 2^0$
$2 = 2^1$
$4 = 2^2$
$8 = 2^3$

Fib(n)

- Fib(n) = $O(2^n)$
- Vždy znovu počítáváme Fib(n-2), když počítáme Fib(n) a Fib(n-1)

Výpočet Fibonacciho čísel pomocí dynamického programování

Fib-DP(n)

c[1] = 1

c[2] = 1

for *i* ← 1..*n* **do**

c[i] ← *c*[*i* − 1] + c[*i* − 2]

- **end for**
- **return** *c*[*n*]

- O(n)

Směna

- Vstup: máme n mincí různých hodnot $1 = v_1 < v_2 < . . . < v_n$
- Problém: Vyplatit částku C a přitom použít co nejmenší počet mincí
- Poznámka: všechny v_i a C jsou kladná celá čísla.

Směna – dynamické programování

- $M(j)$ = minimální počet mincí nutných pro provedení směny pro částku peněz j
- $M(j) = \min_i \{M(j - v_i)\} + 1$
- Nejmenší počet mincí potřebných pro výplatu j je nejmenší počet k tomu, aby platilo $j - v_i$, **plus jedna**
- Při vypočtu, začněme od nejmenšího množství, 1, a budeme vytvářet tabulku M

Směna - příklad

$v_1 = 1$

$v_2 = 3$

$v_3 = 4$

$C = 10$

Směna - příklad

$v_1 = 1$

$v_2 = 3$

$v_3 = 4$

$C = 10$

$M[1] = 1$

$M[i] \quad \boxed{1}$

Směna - příklad

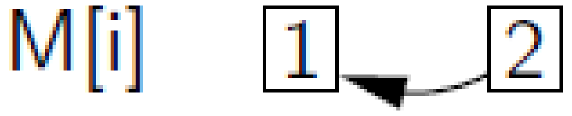
$v_1 = 1$

$v_2 = 3$

$v_3 = 4$

$C = 10$

$M[2] = M[1] + 1$



Směna - příklad

$v_1 = 1$

$v_2 = 3$

$v_3 = 4$

$C = 10$

$M[3] = 1$

M[i] $\boxed{1}$ $\boxed{2}$ $\boxed{1}$

Směna - příklad

$v_1 = 1$

$v_2 = 3$

$v_3 = 4$

$C = 10$

$M[4] = 1$

M[i]	1	2	1	1

Směna - příklad

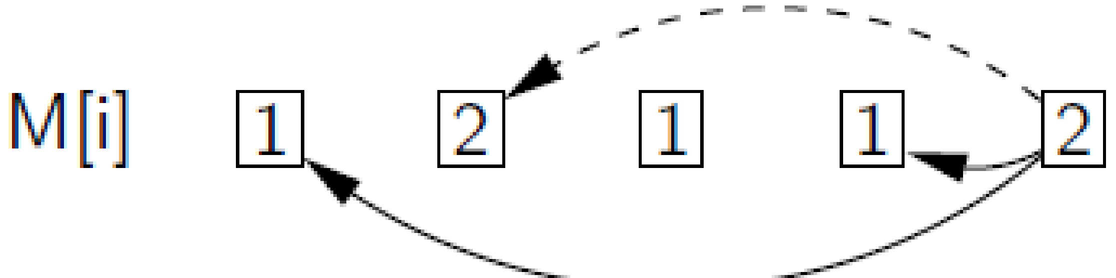
$v_1 = 1$

$v_2 = 3$

$v_3 = 4$

$C = 10$

$M[5] = M[4] + 1$



Směna - příklad

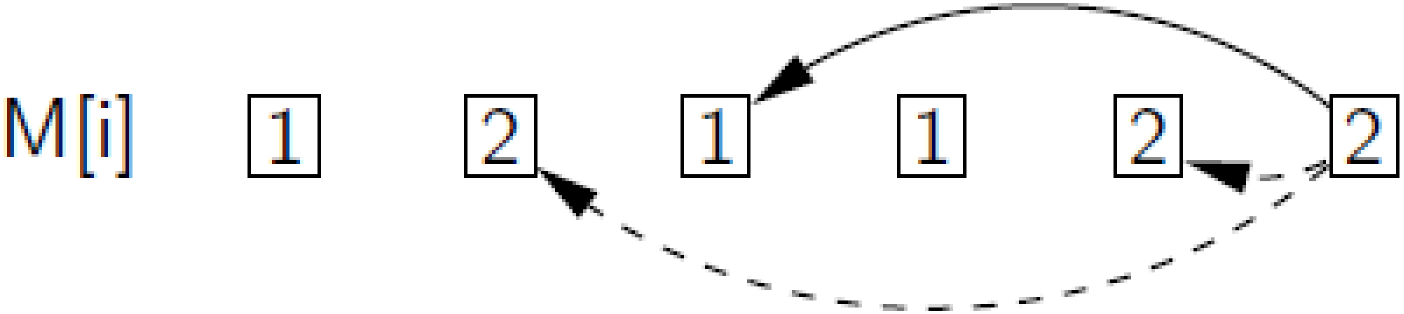
$v_1 = 1$

$v_2 = 3$

$v_3 = 4$

$C = 10$

$M[6] = M[3] + 1$



Směna - příklad

$v_1 = 1$

$v_2 = 3$

$v_3 = 4$

$C = 10$

$M[7] = M[3] + 1$

$M[i]$ $\boxed{1}$ $\boxed{2}$ $\boxed{1}$ $\boxed{1}$ $\boxed{2}$ $\boxed{2}$ $\boxed{2}$

Směna - příklad

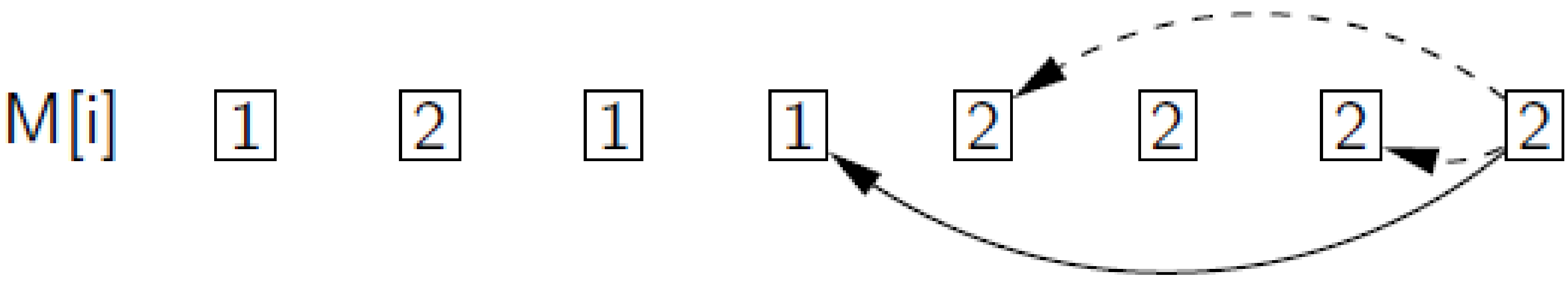
$v_1 = 1$

$v_2 = 3$

$v_3 = 4$

$C = 10$

$M[8] = M[4] + 1$



Směna - příklad

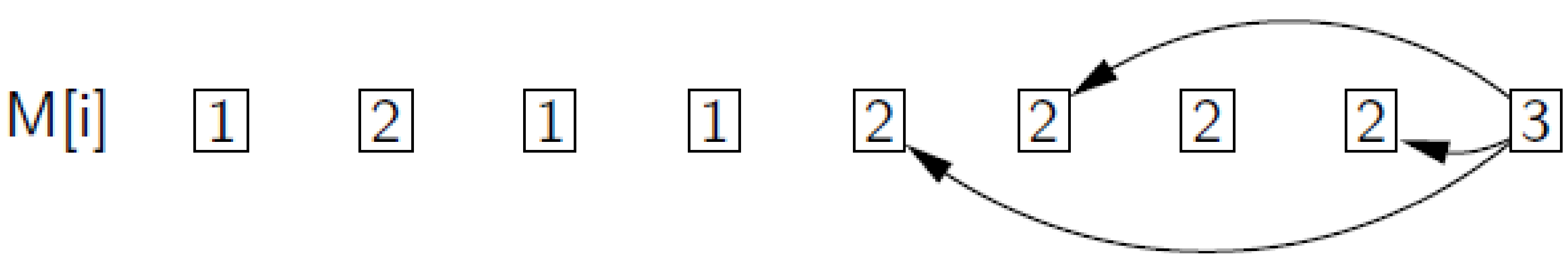
$v_1 = 1$

$v_2 = 3$

$v_3 = 4$

$C = 10$

$M[9] = M[6] + 1$ nebo $M[9] = M[8] + 1$ nebo $M[9] = M[5] + 1$



Směna - příklad

$v_1 = 1$

$v_2 = 3$

$v_3 = 4$

$C = 10$

$M[10] = M[6] + 1$ nebo $M[10] = M[7] + 1$

M[i] $\boxed{1}$ $\boxed{2}$ $\boxed{1}$ $\boxed{1}$ $\boxed{2}$ $\boxed{2}$ $\boxed{2}$ $\boxed{2}$ $\boxed{3}$ $\boxed{3}$

Směna – dynamické programování

```
Směna-min(C, v)
M[0] ← 0 {M[j] = ∞ where i < 0}
for j ← 1..C do
    min ← ∞
    for i ← 1..|v| do
        if M[j − v[i]] + 1 < min then
            min ← M[j − v[i]] + 1
        end if
    end for
    return M[C]
end for
```

Prvky dynamického programování

- Optimální substruktura
- Překrývající se podproblémy
- Rekonstrukce optimálního řešení

Optimální substruktura

Problém má **optimální substrukturu**, pokud optimální řešení je složeno z optimálních řešení dílčích podproblémů

Určení "optimální substruktury“

- Ukázat, že řešení zahrnuje provádění výběru, následně je nutné řešit podproblém nebo soubor podproblémů
- Předpokládat, že máme možnost volby, která vede k optimálnímu řešení (prozatím neřešíme, jak tuto volbu provést)
- Určit, které podproblémy zůstanou po optimální volbě a jak charakterizovat prostor podproblémů
- Ukázat, že řešení podproblémů používaných v optimálním řešení musí být také optimální

Optimální substruktura

Určení "optimální substruktury“

- Ukázat, že řešení podproblémů používaných v optimálním řešení musí být také optimální
  - Důkaz sporem: Předpokládejme, že řešení každého podproblému není optimální a odvodíme spor
  - Konkrétně, že pokud, že řešení podproblému není optimální, pak celkové řešení není optimální
  - "Cut-and-paste" technika: tvrdíme, že můžeme „vyříznout" suboptimální řešení a „vložit" optimální řešení podproblému vedoucí k celkově lepšímu řešení. Protože předpokládáme, že celkové řešení je optimální, pak je to rozpor

Překrývající se podproblémy

- Dynamické programování vede k efektivnímu řešení problémů v případech, kdy jsou podproblémy užity opakovaně
- Dynamické programování je podobné metodě *rozděl a panuj*
  - Identifikace podproblémů
  - Řešení podproblémů
  - Kombinace řešení
- Při metodě *rozděl a panuj* jsou ale podproblémy unikátní
- V dynamickém programování podproblémy se často opakují. Účinnost je provedena opětovným použitím řešení. Při řešení jsou podproblémy ukládány do tabulky

Rekonstrukce optimálního řešení

- Poslední složkou dynamického programování algoritmu je rekonstrukce řešení
- Tabulka pro dynamické programování obsahuje hodnotu, která je optimalizována. Rekonstrukce množiny voleb provedená pro hledání optimálního řešení však není vždy triviální
- Obvykle to zahrnuje ukládání provedené volby včetně hodnot buď v tabulce pro dynamické programování nebo v jiné tabulce stejných rozměrů

Minimální editační vzdálenost

Jak se liší jsou řetězce LEAD a LAST?

Minimální editační vzdálenost

Jak se liší jsou řetězce LEAD a LAST?

Přístup: Spočítáme počet editačních operací potřebných pro transformaci jednoho řetězce do druhého

Minimální editační vzdálenost

Jak se liší jsou řetězce LEAD a LAST?

Přístup: Spočítáme počet editačních operací potřebných pro transformaci jednoho řetězce do druhého

- Definujeme tři (nebo dvě) operace pro úpravu: Vložit, Odstranit a Nahradit
- Označujeme to jako Minimum Edit Distance (MED) nebo jako Levenshteinova vzdálenost

Minimální editační vzdálenost

Jak se liší jsou řetězce LEAD a LAST?

Tři

Např:

Jedno vymazání: LEAD $\rightarrow$ LAD

Jedno nahrazení: LAD $\rightarrow$ LAT

Jedno vložení: LAT $\rightarrow$ LAST

MED spočítaná pomocí dynamického programování

Optimální substruktura

- Dva řetězce: s_1 a s_2
- $s_1[1..i]$ a $s_2[1..j]$ mají minimální editační vzdálenost c
- Označujeme to jako: $d(s_1[1..i], s_2[1..j]) = c$
- Platí tvrzení: c = je nejmenší ze 4 hodnot:

1. $d(s_1[1..i-1], s_2[1..j-1])$ pokud $s_1[i] == s_2[j]$ - equivalence
2. $d(s_1[1..i-1], s_2[1..j-1]) + 1$ pokud $s_1[i] != s_2[j]$ - nahrazení
3. $d(s_1[1..i], s_2[1..j-1]) + 1$ - vložení
4. $d(s_1[1..i-1], s_2[1..j]) + 1$ - mazání

Rekurzivní MED

- Tato minimalizace vede potenciálnímu řešení

d(s_1, s_2)
if s_1.size = 0 **and** s_2.size = 0 **then**
    return 0
end if
if s_1.size = 0 **then**
    return d(s_1, s_2[1..s_2.size − 1]) + 1 **delete**
end if
if s_2.size = 0 **then**
    return d(s_1[1..s_1.size − 1], s_2) + 1 **insert**
end if
return Min(d(s_1, s_2[1..s_2.size − 1]) + 1,
        d(s_1[1..s_1.size − 1], s_2) + 1,
        d(s_1[1..s_1.size − 1], s_2[1..s_2.size − 1]) + 1))

Výpočet minimální editační vzdálenosti (Levenshteinovy)

Initialization

```
D(i,0) = i
D(0,j) = j
```

Recurrence Relation:

For each $i = 1 \ldots M$

For each $j = 1 \ldots N$

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } X(i) \neq Y(j) \\ 0; & \text{if } X(i) = Y(j) \end{cases} \end{cases}$$

Termination:

```
D(N,M) is distance
```

Dynamické programování pro MED

- Místo rekurze s celými řetězci a s podmínkou pro zastavení, zkonstruujeme řešení pomocí řešení menších podproblémů
- **Konstrukce tabulky**:
- Nechť $k = s_1$.size a $l = s_2$.size
- Cíl: zkonstruujeme tabulku M o rozměrech $k + 1$ na $l + 1$ tak, aby hodnota $M[i,j]$ určovala minimální počet editací k převedení $s_1[1...i\text{-}1]$ na $s_2[1... j\text{-}1]$.
- Pokud tento cíl je splněn, pak $d(s_1, s_2) = M[k+1, l+1]$

Tabulka MED

Příklad tabulky MED

Inicializace: nastavení horního řádku a levého sloupce

	Ø	L	A	S	T
Ø					
L					
E					
A					
D					

Tabulka MED

Příklad tabulky MED

Inicializace: nastavení horního řádku a levého sloupce

	∅	L	A	S	T
∅	0	1	2	3	4
L	1				
E	2				
A	3				
D	4				

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i-1,j-1]+1$, $M[i-1,j]+1$, $M[i,j-1]+1$

	Ø	L	A	S	T
Ø	0	1	2	3	4
L	1				
E	2				
A	3				
D	4				

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i-1,j-1]+1$, $M[i-1,j]+1$, $M[i,j-1]+1$

	∅	L	A	S	T
∅	0	1	2	3	4
L	1	0			
E	2				
A	3				
D	4				

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i\text{-}1,j\text{-}1]+1$, $M[i\text{-}1,j]+1$, $M[i,j\text{-}1]+1$

	Ø	L	A	S	T
Ø	0	1	2	3	4
L	1	0			
E	2				
A	3				
D	4				

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i\text{-}1,j\text{-}1]+1$, $M[i\text{-}1,j]+1$, $M[i,j\text{-}1]+1$

	∅	L	A	S	T
∅	0	1	2	3	4
L	1	0			
E	2	1			
A	3				
D	4				

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i\text{-}1,j\text{-}1]+1$, $M[i\text{-}1,j]+1$, $M[i,j\text{-}1]+1$

	Ø	L	A	S	T
Ø	0	1	2	3	4
L	1	0			
E	2	1			
A	3	2			
D	4				

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i\text{-}1,j\text{-}1]+1$, $M[i\text{-}1,j]+1$, $M[i,j\text{-}1]+1$

	Ø	L	A	S	T
Ø	0	1	2	3	4
L	1	0			
E	2	1			
A	3	2			
D	4	3			

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i\text{-}1,j\text{-}1]+1$, $M[i\text{-}1,j]+1$, $M[i,j\text{-}1]+1$

	Ø	L	A	S	T
Ø	0	1	2	3	4
L	1	0	1		
E	2	1			
A	3	2			
D	4	3			

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i-1,j-1]+1$, $M[i-1,j]+1$, $M[i,j-1]+1$

	∅	L	A	S	T
∅	0	1	2	3	4
L	1	0	1		
E	2	1	1		
A	3	2			
D	4	3			

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i\text{-}1,j\text{-}1]+1$, $M[i\text{-}1,j]+1$, $M[i,j\text{-}1]+1$

	∅	L	A	S	T
∅	0	1	2	3	4
L	1	0	1		
E	2	1	1		
A	3	2	1		
D	4	3			

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i\text{-}1,j\text{-}1]+1$, $M[i\text{-}1,j]+1$, $M[i,j\text{-}1]+1$

	∅	L	A	S	T
∅	0	1	2	3	4
L	1	0	1		
E	2	1	1		
A	3	2	1		
D	4	3	2		

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i-1,j-1]+1$, $M[i-1,j]+1$, $M[i,j-1]+1$

	∅	L	A	S	T
∅	0	1	2	3	4
L	1	0	1	2	
E	2	1	1		
A	3	2	1		
D	4	3	2		

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot
$M[i-1,j-1]+1$, $M[i-1,j]+1$, $M[i,j-1]+1$

	∅	L	A	S	T
∅	0	1	2	3	4
L	1	0	1	2	
E	2	1	1	2	
A	3	2	1		
D	4	3	2		

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i\text{-}1,j\text{-}1]+1$, $M[i\text{-}1,j]+1$, $M[i,j\text{-}1]+1$

	∅	L	A	S	T
∅	0	1	2	3	4
L	1	0	1	2	
E	2	1	1	2	
A	3	2	1	2	
D	4	3	2		

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i\text{-}1,j\text{-}1]+1$, $M[i\text{-}1,j]+1$, $M[i,j\text{-}1]+1$

	∅	L	A	S	T
∅	0	1	2	3	4
L	1	0	1	2	
E	2	1	1	2	
A	3	2	1	2	
D	4	3	2	2	

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i-1,j-1]+1$, $M[i-1,j]+1$, $M[i,j-1]+1$

	∅	L	A	S	T
∅	0	1	2	3	4
L	1	0	1	2	3
E	2	1	1	2	
A	3	2	1	2	
D	4	3	2	2	

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot
$M[i\text{-}1,j\text{-}1]+1$, $M[i\text{-}1,j]+1$, $M[i,j\text{-}1]+1$

	∅	L	A	S	T
∅	0	1	2	3	4
L	1	0	1	2	3
E	2	1	1	2	3
A	3	2	1	2	
D	4	3	2	2	

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot
$M[i\text{-}1,j\text{-}1]+1$, $M[i\text{-}1,j]+1$, $M[i,j\text{-}1]+1$

	∅	L	A	S	T
∅	0	1	2	3	4
L	1	0	1	2	3
E	2	1	1	2	3
A	3	2	1	2	3
D	4	3	2	2	

Tabulka MED

Příklad tabulky MED

Postup: pro každou buňku opakujeme porovnání hodnot $M[i\text{-}1,j\text{-}1]+1$, $M[i\text{-}1,j]+1$, $M[i,j\text{-}1]+1$

	∅	L	A	S	T
∅	0	1	2	3	4
L	1	0	1	2	3
E	2	1	1	2	3
A	3	2	1	2	3
D	4	3	2	2	3

Zobecnění minimální editační vzdálenosti

Zatím jsme řekli, že operace Vložení, Odstranění a Nahrazení mají všechny stejnou cenu

Můžeme ale nastavit různé náklady pro každdou operaci

V tomto případě porovnáváme hodnoty:

$M[i\text{-}1, j\text{-}1]$ + náklady Nahrazení

$M[i, j\text{-}1]$ + náklady Vložení

$M[\mathrm{i}\text{-}1, j]$ + náklady Odstranění (Mazání)

Případný domácí úkol: Napište pseudokód algoritmu