Predavanje 6

Dinamičko programiranje

- tehnika dizajniranja alg. poput podijeli-pa-vladaj

MOTIVACIJA: Fibonaccijevi br.

$$F_n = \begin{cases} 0, & n=0 \\ 1, & n=1 \\ F_{n-1}+F_{n-2}, & n \geq 2 \end{cases}$$

NAIVAN REKURZIVAN ALG.

```
fib(n)
    if n=0 return 0
    if n=1 return 1
    else return fib(n-1) + fib(n-2)
```

Pr.:

$$\begin{array}{c} fib(4) \\ \swarrow \quad \searrow \\ fib(3) \quad fib(2) \\ \swarrow \quad \searrow \\ fib(2) \quad fib(1) \end{array}$$

VSA: $T(n) = T(n-1) + T(n-2) + \Theta(1)$

$\geq 2T(n-2)$

$= \Omega(2^n)$

eksp. vrijeme izvršavanja alg.

Problem dijelimo na potprobleme koji se preklapaju

MEMORIZACIJA:

Ideja: Pamtite ono što je već izračunato (pomoću rj. ili polja)

PSEUDOKOD:

memo = [0...n] → polje s indeksima 0 do n

```
fib(n)
  if(n in memo)
      return memo[n]
  if (n==0)
      return 0
  if (n==1)
      return 1
  else
    f=fib(n-1)+fib(n-2)
    return f
```

$\Theta(1)$ bez rek. poziva

Najduži zajednički podniz (Longest Common Subsequence)

Pr.: Dana su nam dva stringa (niza)

x: A B C B D A B

y: B D C A B A

BDAB

BCAB

BCBA

LCS(x, y) = BCBA

nije f-ja

- BRUTE FORCE alg. za LCS problem

- za svaki alg. koji pronalazi rj. t.d. pretražuje cjelokupni prostor rj. nazivamo BRUTE FORCE

Definirajmo BF alg. za LCS

x $[1 \ldots m]$ ← br. podniza od x je 2^m

y $[1 \ldots n]$ provjerit u $\Theta(n)$ vremenu je li podniz od x

VSA od BF alg. je $2^m \cdot \Theta(n) = \Theta(n \cdot 2^m)$ ← eksp. alg.

Možemo li bolje od ovoga? Kako def. potpr.?

VSA od memorizirane implementacije:

- fib(k) se računa samo jednom za $\forall k$
- složenost fib(k) bez rekurzivnih poziva je $\Theta(1)$
- ukupna složenost je broj funkcijskih poziva puta ocjena po pozivu

$\Rightarrow \Theta(n)$ ✓

Općenito kod DP pristupa

- podjeliti problem na $\overbrace{\text{potprobleme koji se preklapaju}}^{\text{NETRIVIJALNO}}$
- pomoću memorizacije pamtiti rješenja potproblema
- rekurzivno rješavati potprobleme koji nisu memorizirani

VSA od DP-a:

- broj potproblema koji rješavamo $* T(\text{potpr.})$ ignoriramo rekurzivne pozive

DP alg. - bottom up. (najčešće ovo zovu DP-om)

- pristup potpuno analogan rekurzija + memorizacija pristupa, ali bez korištenja rek. poziva

PSEUDOKOD:

```
fib(n)
  memo = [0...n]
  for k ← 0 to n
    if k==0 set memo[0]=0
    else if k==1 set memo[1]=1
    else memo[k]=memo[k-1]+memo[k-2]
  return memo[n]
```

VSA: $\Theta(n)$, prostorna složenost: $\Theta(n)$

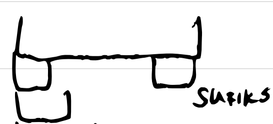
x - niz duljine m

y - niz duljine n

Notacija:

$|b|$ označava duljinu niza S

$S[b, i]$ - prefiks niza S duljine i



Def.:

$$c[i,j] = |LCS(x[1,\dots,i], y[1,\dots,j])|$$

Dakle, nas zanima

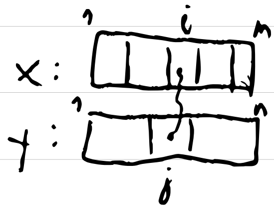
$$c[m,n] = |LCS(x,y)|$$

Tm.:

$$c[i,j] = \begin{cases} c[i-1,j-1]+1, & \text{ako } x[i]=y[j] \\ \max\{c[i-1,j],\, c[i,j-1]\}, & \text{inače} \end{cases}$$

Dokaz:

DISKUSIJA:



NOTACIJA:

BSO neka $c[i,j]=k$ i neka $z[1\ldots k] = LCS(x[1\ldots i], y[1\ldots j])$

sl. 1. $x[i]=y[j]$

Primjetite da vrijedi $x[i]=y[j]=z[k]$

$\Rightarrow c[i-1,j-1] = |LCS(x[1\ldots i], y[1\ldots j-1])| = k-1$

TRIVIJALNO VRIJEDI: $c[i,j] = c[i-1,j-1]+1$

sl. 2. $x[i] \neq y[j]$

Primjeti $\underbrace{z[k] \neq x[i]}$ ili $z[k] \neq y[j]$

$$z[1\ldots k] = LCS(x[1\ldots i-1], y[1\ldots j-1])$$

$\rightarrow z[1...k] = LCS(x[1...i], y[1...j])$

Primjeti $\underbrace{z[k] \neq x[i]}$ ili $z[k] \neq y[j]$

$$z[1...k] = LCS(x[1...i-1], y[1...j])$$

Stoga slijedi:

$$c[i,j] = \max\{c[i-1,j], c[i,j-1]\}$$

Preth. teorem izravno definira rekurzivni alg. (do na rubne uvjete)

$c = [n \times m]$

```
LCS (x, y, i, j)
 if(i=0 or j=0) return 0
 if(x[i] = y[j]
    c[i,j] ← LCS (x, y, i-1, j-1) + 1
 else
    c[i,j] ← max{LCS(x, y, i, j-1), LCS(x, y, i-1, j)}
 return c[i,j]
```

Inicijalni poziv $LCS(x, y, m, n) \rightarrow$ return $c[m,n]$

VSA: broj potproblema $*$ T(potproblema)

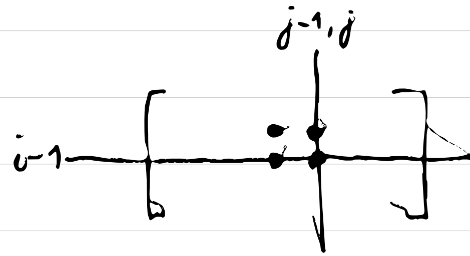
$n * m$ potproblema i konstantno vrijeme po potproblemu

$\Rightarrow \Theta(n \cdot m)$

Prostornu složenost: $\Theta(n \cdot m)$

Bottom-up (DP pristup) (Izračunati DP tablicu C bez rek.)

		A	B	C	B	D	A	B
	0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1	1
D	0	0	1	1	1	2	2	2
C	0	0	1	2	2	2	2	2
A	0	1	1	2	2	2	3	3
B	0	1	2	2	3	3	3	4
A	0	1	2	2	3	3	4	$\boxed{4}$ C[m,n]



VSA: $\Theta(m \cdot n)$

VSA od rekonstruiranog LCS(X,Y): $\Theta(m+n)$