Predavanje 7

Dinamičko programiranje

Ponavljanje

1) Podijeliti problem na potprobleme
2) Pomoću memorizacije ponuditi rješenje potproblema
3) Rekurzivno riješiti potprobleme koji nisu memorizirani

VSA DP-a

$\#\text{ potproblema} \cdot T(\text{potproblem})$

Bez rekurzivnih poziva

1) Fib brojevi $[0 \dots n]$ polje duljine $n+1$
2) Najduži zajednički podniz (LCS) — problemi su prefiksi

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

$y = [1 \dots n]$ $\quad$ ↑ entry u din. tablici

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

1. način: promatraj kao rekurzivnu def. => implementacija REKURZIVA + MEMORIZACIJA
2. način: Def. entryja u tablici
   a) shvatiti uredaj
   b) izračunat elem. din. tablice

"BOTTOM-UP" (zove se u praksi DP)

VSA: $\Theta(m \cdot n)$, rekonstrukcija LCS-a iz tablice $\Theta(m+n)$

EDIT DISTANCE

- neka su dana $x[1..m], y[1..n]$ nizovi znakova (stringovi), želimo odrediti minimalan broj tzv. EDIT operacija potrebnih da se string x pretvori u y

EDIT operacije:

- INSERT $(y[j], i)$ znak $y[j]$ umeće u x na poziciju i
- REMOVE/DELETE (i) znak $x[i]$ se briše iz stringa x
- REPLACE $(x[i], y[j])$ znak $x[i]$ se zamjenjuje s $y[j]$

Pr.

$x =$ 'geek'
$y =$ 'gesek'

} x možemo pretvoriti u y koristeći samo jednu EDIT operaciju:

INSERT $(y[3], 3)$

Pr.

$x =$ 'cat'
$y =$ 'cut'

} REPLACE $(x[2], y[2])$

Pr.

$x =$ 'sunday'
$y =$ 'saturday'

} REPLACE $(x[3], y[5])$
INSERT $(y[2], 2)$
INSERT $(y[3], 2)$

→ uvijek indekse koristimo u terminu originalnog stringa x

Općenito, edit operacije mogu imati nekakvu cijenu koja ovisi o problemu
- problem usporedbe dvije DNK sekvence (neke mutacije su više, a neke manje vjerojatne)
- spellchecker (neke pogreške su više vjerojatne, npr. (s ⟷ a))

DANAS:
- pretp. ćemo da su sve edit operacije cijene 1 (HAMMING DISTANCE)

ALGORITAM:
1 TASK: Kako def. potprobleme?
) kod LCS ; kod EDIT-DIST ulazni podatci su stringovi

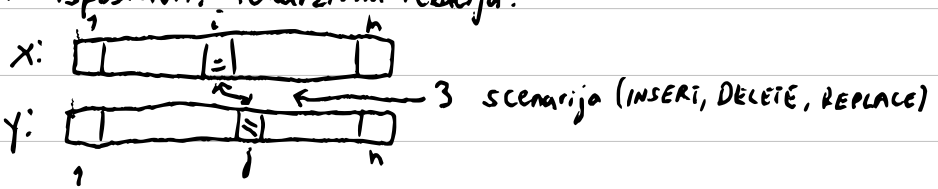
Potprobleme na stringovima: - prefiks (|LCS|) $x[1..i]$
- sufiks $x[i..m]$
- podstring $x[i..j]$

DZ.

Reformulirajte DP za LCS u kontekstu sufiksa

Def.: $c(i,j) = \text{EDIT-DISTANCE}(x[i..m], y[j..m])$

- želimo uspostaviti rekurzivnu relaciju:



Dogodi se:

1) INSERT $(y[j], i)$: $c(i,j) = c(i,j+1) + 1$ ← cijena za INSERT

2) DELETE (i) : $c(i,j) = c(i+1, j) + 1$ <- cijena

3) REPLACE (x[i], y[j]) : $c(i,j) = c(i+1, j+1) + 1$ <- cijena

Rekurzivna relacija

$$c(i,j) = \min\{c(i,j+1)+1,\ c(i+1,j)+1,\ c(i+1,j+1)+1\}$$

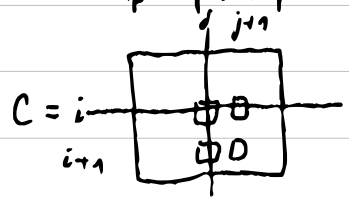
Rekurzija + memorizacija

```
C = [m x n] // dinamicka tablica
EDIT-DIST(x, y, i, j)
  if C[i,j] == nil
    if i>m or j>n return 0
    else
      C[i,j] = min{EDIT-DIST(x,y,i+1,j), EDIT-DIST(x,y,i,j+1), EDIT-DIST(x,y,i+1,j+1)}
  return C[i,j]
```

Inicijalni poziv

EDIT-DIST (x, y, 1, 1)

- "bottom-up" pristup



DZ. Napisite pseudokod za "bottom-up"

Vremenska složenost: # POT PROBLEMA $\cdot T$ (POT PROBLEM)

$\Theta(m \times n)$ $\quad$ $O(1)$

Napomena:

LCS se može modelirati kao EDIT-DIST problem

=> LCS je specijalan slučaj EDIT-DIST-a

- cijena INSERT i DELETE je 1
- cijena za REPLACE $(c_1, c_2) = \begin{cases} 0, & c_1 = c_2 \\ \infty, & \text{inače} \end{cases}$

REPLACE NIJE DOZVOLJEN

Primjer:

x: A B C B D A B

y: B D C A D A

BDAB
BCAB
BCBA

BDAB: DELETE(A), DEL(C), DEL(B), INSERT(C), end of string x

BCAB: DEL(A), INS(D), DEL(D), DEL(D), end of string x

Problem poravnavanja teksta (text justification)

- želimo poravnati tekst u linije od margine do margine na optimalan način

Kako def. optimalnost u ovom problemu?

- input: tekst $T[1 \ldots n]$ niz riječi
- PAGE-WIDTH: duljina stranice od margine do margine
- TOTAL-WIDTH: duljina koju čine riječi i do j uključujući razmake

45
45
45
25

$$cost(i,j) = \begin{cases} \infty, & \text{niz riječi od } i \text{ do } j \text{ ne stane u redak} \\ (\text{PAGE-WIDTH} - \text{TOTAL-WIDTH}(i,j))^3, & \text{inače} \end{cases}$$

↑ cijena da u neku retku stoje riječi od i do j

↑ LATEX

2 : 15+25 = 2 : 40

9+12 : 40 = 11 : 40

UKUPNA CIJENA za tekst T je suma $cost(i,j)$ po svim linijama

=> Želimo raspodijeliti tekst T u linije t.d. ukupna cijena bude minimalna

PITANJE: Greedy strategija (MS WORD) je:

pakiraj riječi u redak dok god stane

(nije optimalna strategija)

ALGORITAM:

(1.) POTPROBLEMI? $T[1 \dots n]$ (string → PREFIKS, SUFIKS, PODSTRING)

$cost(1,i)$ + optimalna cijena potproblema $[i \dots n]$

sufiks

Def.: Cijena potproblema

$c(i) = tekst\_justification(T[i \dots n])$

Rekurzivna relacija

$$c(i) = \min_{j=i+1 \dots n} \{ cost(i,j) + c(j) \}$$

Rekurzija + memorizacija

$c = [1..n]$ // DP tablica

```
Text.Justify(T,i)
  if c_i == nil
  |  if i = n return ???
  |  else
  |     c[i] = min_{j=i+1...n} { cost(i,j) + Text.Justify(T,j) }
  return c[i]
```

INICIJALNI POZIV

Text.Justify(T,1)

Vremenska složenost

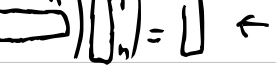
$\Theta(n) \cdot (n-i) = \Theta(n^2)$

PARENTHESIZATION

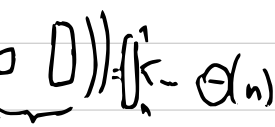
Kako optimalno evaluirati neki asocijativan izraz?

$A_1, A_2, A_3, \ldots, A_n$ ← MATRICE

Pr.:



$\leftarrow \Theta(n^2)$



$\Theta(n)$ $\Theta(n)$ $\leftarrow \Theta(n)$

ulaz: $A_1 \dots A_n$ niz objekta ("string")

ALG.

- potproblem

$(A_1 \cdot A_2 \cdot A_3)(A_4 \cdot A_5)$

$c(i,j)$ = opt. cijena za postupak $A_i, \dots, A_{j-1}$

Rekurzivna relac.

$$c(i,j) = \min_{k=i+1 \dots j} \{ c(i,k) + c(k,j) + \text{cijena operacije } [i \dots k-1] \cdot [k \dots j-1] \}$$