Dinamično programiranje (2. del)

Luka Fürst

ponedeljek, 3. januarja 2022

Danes

- najdaljše skupno podzaporedje
- Levenštejnova razdalja
- najdaljše naraščajoče podzaporedje
- primer uporabe DP na drevesih

Najdaljše skupno podzaporedje

- angl. longest common subsequence (LCS)
- vhod
  - niz $A = a_1 a_2 \ldots a_n$
  - niz $B = b_1 b_2 \ldots b_m$
- izhod
  - najdaljši možni niz $C = c_1 c_2 \ldots c_k$, tako da
    - za vsak i velja, da znak c_i nastopa tako v nizu A kot v nizu B
    - če je $i < j$, potem c_i nastopa pred c_j v obeh vhodnih nizih
  - pogosto nas zanima samo dolžina takega niza ($\mathrm{lcs}(A, B)$)

Primer

- vhod
  - $A =$ PROGA
  - $B =$ ROLETA
- izhod
  - $C =$ ROA (PROGA, ROLETA)
  - $\mathrm{lcs}(A, B) = 3$

Reševanje

- $A_i = a_1 a_2 \ldots a_i$
- $B_j = b_1 b_2 \ldots b_j$

$$\mathrm{lcs}(A_i, B_j) = \begin{cases} 0, & i = 0 \text{ ali } j = 0 \\ \mathrm{lcs}(A_{i-1}, B_{j-1}) + 1, & a_i = b_j \\ \max\{\mathrm{lcs}(A_i, B_{j-1}),\ \mathrm{lcs}(A_{i-1}, B_j)\} & \text{sicer} \end{cases}$$

Reševanje

- rekurzivno
  - po formuli
  - memoizacijska tabela $(n+1) \times (m+1)$
  - element $[i]\,[j]$ hrani že izračunano vrednost $\mathrm{lcs}(A_i, B_j)$
- iterativno
  - tabelo $(n+1) \times (m+1)$ polnimo po vrsticah
- rekonstrukcija podzaporedja
  - za vsako celico (i, j) hranimo položaj sosede, iz katere izhaja vrednost v celici (i, j)
  - $\nwarrow$, če je $a_i = b_j$
  - $\leftarrow$ (leva soseda $\geq$ zgornja soseda) oz. $\uparrow$ (leva soseda $<$ zgornja soseda), če je $a_i \neq b_j$
- časovna zahtevnost
  - $O(nm)$

Primer

		R	O	L	E	T	A
	0	0	0	0	0	0	0
P	0	←0	←0	←0	←0	←0	←0
R	0	↖1	←1	←1	←1	←1	←1
O	0	↑1	↖2	←2	←2	←2	←2
G	0	↑1	↑2	←2	←2	←2	←2
A	0	↑1	↑2	←2	←2	←2	↖3

Levenštejnova razdalja

- $A = a_1 a_2 \ldots a_n$
- $B = b_1 b_2 \ldots b_m$
- niz A preoblikujemo v niz B, pri čemer v vsakem koraku storimo eno od sledečega:
  - izbrišemo znak
  - vstavimo znak
  - zamenjamo znak
- Levenštejnova razdalja = najmanjše potrebno število korakov
- pogosto: Levenštejnova razdalja = urejevalna razdalja (angl. edit distance), čeprav obstajajo tudi druge metrike, ki jim lahko rečemo »urejevalna razdalja«

Primer

- $A =$ PROGA
- $B =$ ROLETA
- 1. korak: odstranimo P (dobimo ROGA)
- 2. korak: G $\rightarrow$ L (dobimo ROLA)
- 3. korak: vstavimo E (dobimo ROLEA)
- 4. korak: vstavimo T (dobimo ROLETA)
- d(PROGA, ROLETA) $= 4$

Reševanje

- $A_i = a_1 a_2 \ldots a_i$
- $B_j = b_1 b_2 \ldots b_j$
- naj bo $d_{i,j} = d(A_i, B_j)$
- robni pogoji: $d_{i,0} = d_{0,i} = i$
-

$$d_{i,j} = \begin{cases} d_{i-1,j-1}, & a_i = b_j \\ 1 + \min\{\underbrace{d_{i-1,j}}_{\text{brisanje}}, \underbrace{d_{i,j-1}}_{\text{vstavljanje}}, \underbrace{d_{i-1,j-1}}_{\text{zamenjava}}\}, & a_i \neq b_j \end{cases}$$

- $O(nm)$

Najdaljše naraščajoče podzaporedje

- angl. longest increasing subsequence (LIS)
- vhod
  - celo število $n \in [1, 1000]$
  - cela števila $z_0, \ldots, z_{n-1} \in [-10^9, 10^9]$
- izhod
  - dolžina (ne nujno strnjenega) najdaljšega naraščajočega podzaporedja

Primer

- vhod
  - $n = 10$
  - $Z = (13, 5, 7, 12, 1, 8, 6, 11, 4, 9)$
- izhod
  - $\mathrm{lis}(Z) = 4$ (dolžina LIS)
  - LIS (če ga potrebujemo) $= (5, 7, 8, 9)$

Rešitev v $O(n^2)$

- $D(k)$: dolžina najdaljšega naraščajočega podzaporedja, ki se prične s členom z_k
- $\text{lis}(Z) = \max\{D(0), D(1), \ldots, D(n-1)\}$
- $D(n) = 0$
- $D(k) = \max\{D(i) + 1 \mid i > k \text{ in } z_i > z_k\}$ (za $k < n$)
- rešujemo rekurzivno (z memoizacijo) ali iterativno

Rešitev v $O(n \log n)$

- vzdržujemo zaporedje $L = (L_0, L_1, \ldots, L_{k-1})$, tako da element L_i hrani najmanjši zadnji element izmed vseh doslej videnih naraščajočih podzaporedij (IS) dolžine $i + 1$
- primer:
  - $z = (3, 7, 5, 6, 2, 8)$ (vhodno zaporedje)
  - IS dolžine 1: (3), (7), (5), (6), (2), (8)
  - IS dolžine 2: (3, 7), (3, 5), (3, 6), (3, 8), (7, 8), (5, 6), (5, 8), (6, 8), (2, 8)
  - IS dolžine 3: (3, 7, 8), (3, 5, 6), (3, 5, 8), (3, 6, 8), (5, 6, 8)
  - IS dolžine 4: (3, 5, 6, 8) (= LIS)
  - $L = (2, 5, 6, 8)$

Rešitev v $O(n \log n)$

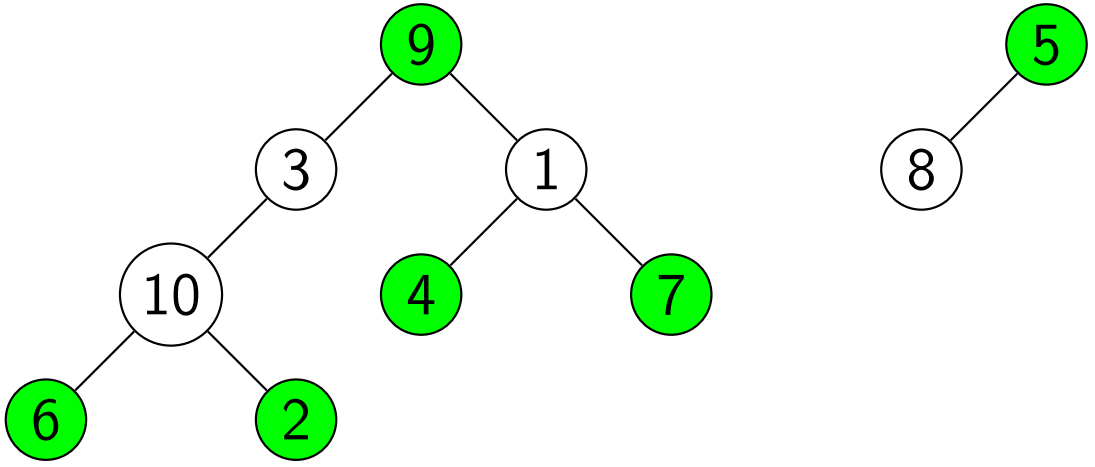
- k = trenutno število elementov v L = trenutna dolžina LIS
- za vsak indeks i v vhodnem zaporedju (z) poiščemo indeks j v zaporedju L, kjer se nahaja prvi element, ki je večji od z_i (če $z_i \geq L_{k-1}$, nastavimo $j := k$)
- nastavimo $L_j := z_i$
- če je $j = k$, smo podaljšali trenutno LIS
- če je $j < k$, smo našli manjši (= bolj perspektiven) zadnji element IS dolžine j
- zaporedje L je vedno urejeno, zato lahko po njem iščemo z dvojiškim iskanjem

Primer uporabe DP na drevesih

- naloga
  - V podjetju je n zaposlenih. Vsak ima največ enega neposredno nadrejenega in poljubno mnogo neposredno podrejenih. Vsak pozna samo neposredno nadrejenega in neposredno podrejene. Največ koliko ljudi lahko izberemo, tako da nihče med njimi ne pozna nikogar drugega?
- vhod
  - $n \in [1, 1000]$
  - $m \in [0, n-1]$
  - pari (u_i, v_i) $(i \in \{1, \ldots, m\})$, ki povedo, da je oseba u_i neposredno nadrejena osebi v_i
- izhod
  - velikost največje podmnožice, v kateri nihče nikogar ne pozna

Primer uporabe DP na drevesih

- vhod je gozd (množica dreves)
- vozlišče u je starš vozlišča $v \iff$ oseba u je neposredno nadrejena osebi v
- primer



- iščemo velikost največje neodvisne množice vozlišč (angl. maximum independent set, MIS)

Reševanje

- $N(u)$ = velikost MIS za poddrevo s korenom u
- $N(u) = \max\{N_{\text{da}}(u), N_{\text{ne}}(u)\}$
- $N_{\text{da}}(u)$
  - velikost MIS, če vozlišče u vključimo v končno množico
  - v tem primeru moramo otroke vozlišča u izpustiti
  - seštejemo $N(.)$ po vnukih vozlišča u in prištejemo 1 (za vozlišče u)
  - $N_{\text{da}}(u) = 1 + \sum_{w \in \text{vnuki}(u)} N(w)$

Reševanje

- $N_{\text{ne}}(u)$
  - velikost MIS, če vozlišča u ne vključimo v končno množico
  - seštejemo $N(.)$ po otrocih vozlišča u
  - $N_{\text{ne}}(u) = \sum_{v \in \text{otroci}(u)} N(v)$
- za vsako drevo poiščemo koren t in izračunamo $N(t)$
- dobljene vrednosti $N(t)$ seštejemo
- rekurzija + memoizacija (hranimo že izračunane vrednosti $N(.)$ za posamezna vozlišča)

Domača naloga: Politična nasprotja

- naloga
  - Na $l + d + c$ sedežev v ravni vrsti želimo razporediti l levičarjev, d desničarjev in c centristov, tako da levičar in desničar nikoli ne bosta sedela skupaj. Na koliko načinov lahko to naredimo?
- vhod
  - cela števila $l \in [0, 100]$, $d \in [0, 100]$, $c \in [0, 100]$
- izhod
  - število načinov po modulu $10^9 + 7$

Domača naloga: Politična nasprotja

- Primer vhoda in izhoda

2 3 2
15

(Možne razporeditve so LLCDDDC, LLCDDCD, LLCDCDD, LLCCDDD, LCLCDDD, LCDDDCL, DDDCLLC, DDDCLCL, DDDCCLL, DDCLLCD, DDCDCLL, DCLLCDD, DCDDCLL, CLLCDDD in CDDDCLL.)