NAGY BEADANDÓ

1) SZUFFIX TÖMB GEN + ALK.: $O(n \cdot \log n)$

Számoljuk meg egy string különböző részstringjeit

2) KÜLÖNBÖZŐ PARIKH-VEKTOROK SZÁMOLÁSA

PREPROCESSZÁLÁS UTÁN.

ABAAB → 3 db A 2 db B

PV: (3,2)

KÉT BETŰ: LIN.

HÁROM BETŰ: CIKK alapján

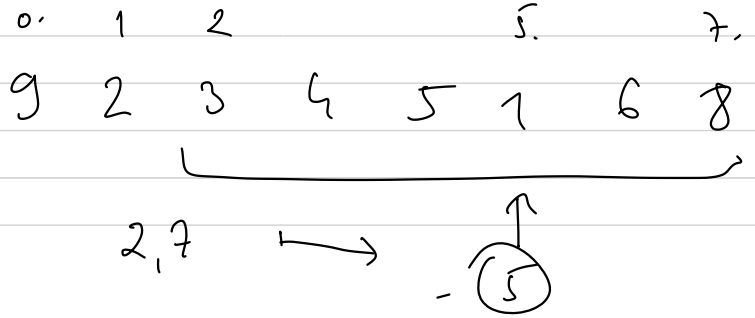
3) RMQ és LCA implementálási lehetőségei és alapjai

RMQ: RANGE MINIMUM QUERY

LCA: LOWEST COMMON ANCESTOR

RMQ: pl. számokból álló tömb A csúcsai

query: i, j output: A[i] ... A[j] rész minimális elem pozíciója.



MEGOLDÁS : i..j szór minimumkeresés $O(j-i)$ idő.

DE. ELŐFELDOLGOZÁS utáni sok query.

↓

leggyorsabb ideje / tárhely ill a query-k futási ideje.

- TRIV. megközelítés $\forall i,j$ -re tároljuk el, hol van $A[i..j]$ min. pozíció.

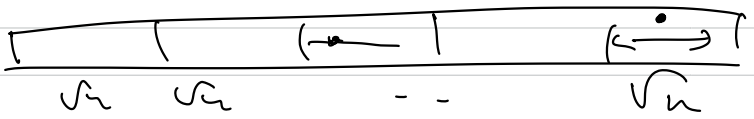
  $O(n^3)$ után $O(1)$ query

  ↓

  kicsit okosan: $O(n^2)$ de n nagy → nem OK,

- $O(n)$, $O(\sqrt{n})$ query: darabolás.

$\sqrt{n}$ db $\sqrt{n}$ hosszú intervallumra számoljuk ki előre



$\rightarrow O(n)$ idő

Query:

$O(\sqrt{n})$

"SPARSE TABLE" használata

$\forall i \in 0 \ldots (n-1) \quad \forall j : 0 \ldots \log_2 n$

RMQ $[i \ldots i+2^j-1]$ eltároljuk. $(2^j$ hosszú is.$)$

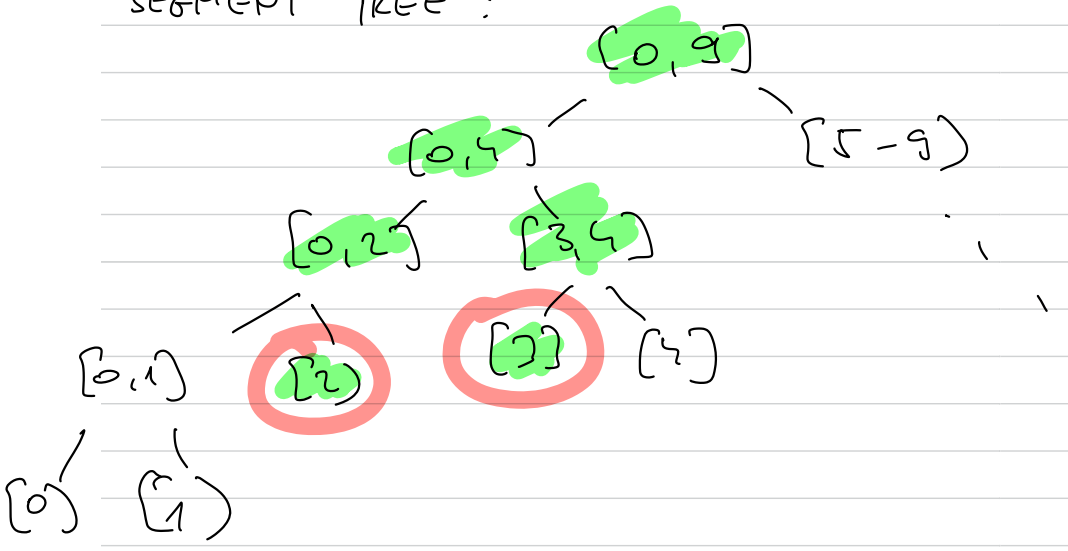
QUERY:

5..17 hol a minimum

$O(1)$

legalább: $O(n \cdot \log n)$ (dinamikusan jó-re)

"SEGMENT TREE":



← kitrámolja előre

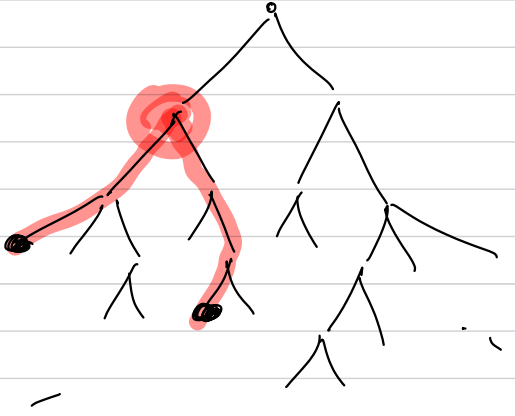
$O(n)$: lekérés

$O(\log n)$: query

CÉL: $O(n)$ index

$O(1)$ q.

LOWEST COMMON ANCESTOR:



CÉL: $\forall i,j$:

$LCA(i,j)$

query legyen gyors

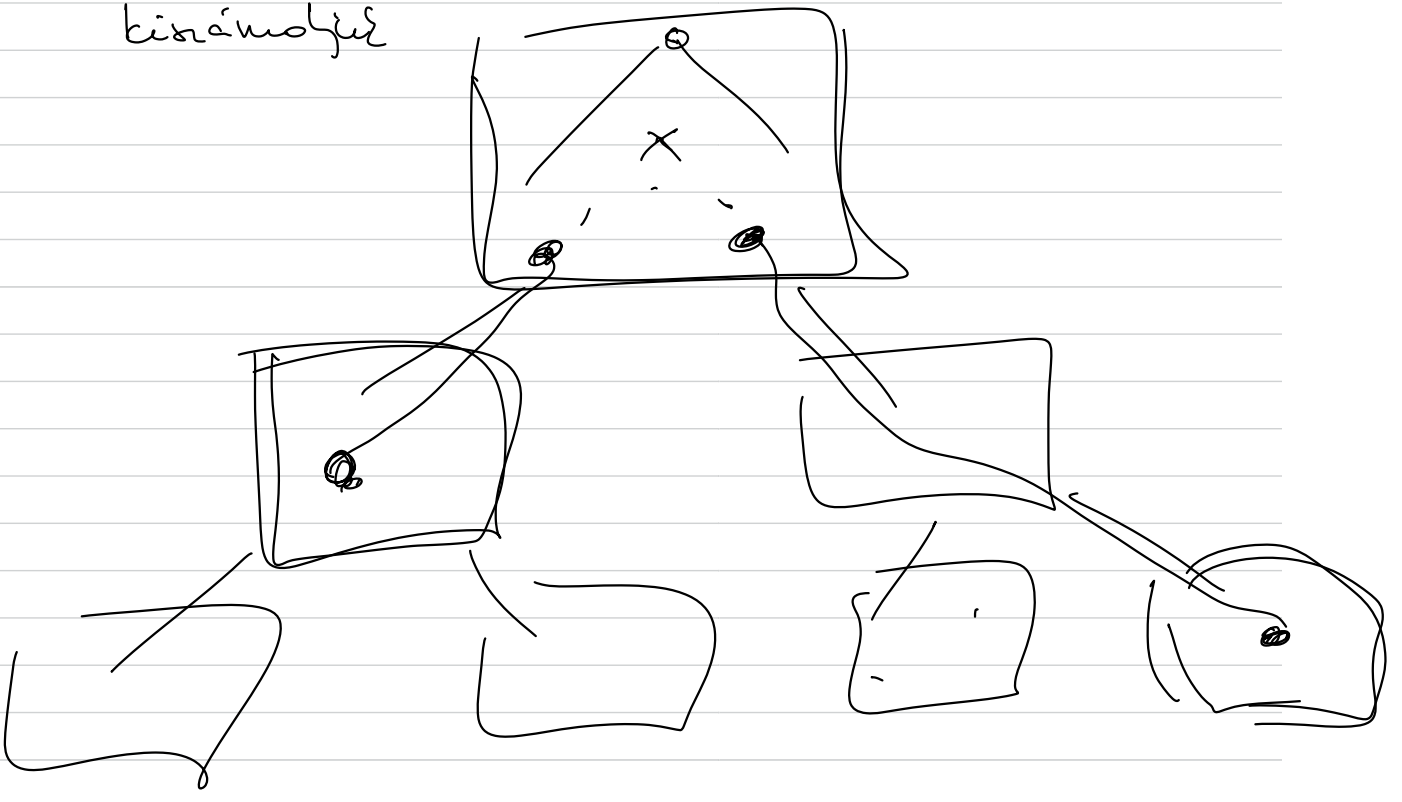
NAIV: $O(n^2)$, $O(1)$

nem jó

ALK. pl. szuffix fa: leghosszabb közös prefix (LCP)

DARABOLÁS : $O(\sqrt{n})$ db $O(\sqrt{n})$ méretű részre elősz

kiszámolás



$O(n)$, $O(\sqrt{n})$

"SPARSE TABLE" -szerű

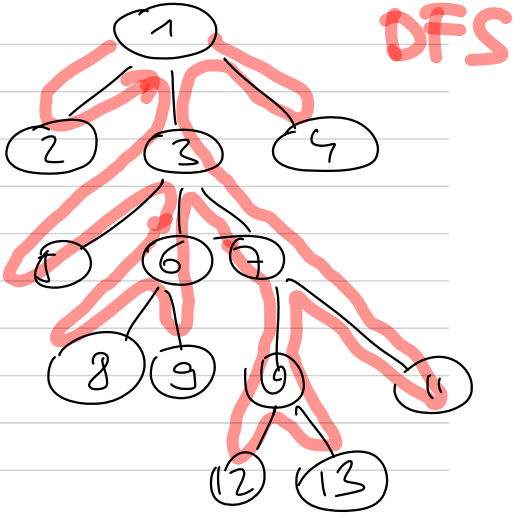
MINDENKINEK ELŐRE a 2^j-edik őst eltárolja! (DP)

$O(n \log n)$ $O(\log n)$ query

BINÁRIS KERESÉS

RMQ és LCA ugyanaz

LCA MEGOLDÁSA RMQ-ként



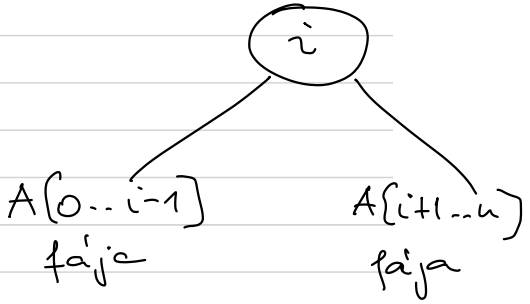
DFS → VEKTOR

1, 2, 1, 3, 5, 3, 6, 8, 6, 9, 6, 3, 7, ...

LCA(5,7) ⟷ RMQ(...)

RMQ visszavezetés LCA-re

végeredmény: fa, gyökér: vektor min pozíciója: i



RMQ[i,j] query $\Leftrightarrow$ fában LCA(i,j)

MINDKÉT VISSZAVEZETÉS $O(n)$

DINAMIKUS LEGYÁRTÁS, VEREM SEGÍTSÉGÉVEL

VÉGEREDMÉNY:

RMQ ~ LCA

ÉSREVÉTEL: RMQ → LCA → RMQ : nem ugyanaz az RMQ,

itt a szomszédos elemek közti különbség ± 1.

~~3 7 2 5~~ ,

3 4 5 4 3 2 3 4 5 6 5 4

↓ index ehhez

MINDEN $\log n$ hosszú rész kódolható ± 1-ekkel

pl. 3 4 5 4 3 2 1 2 3 4 3

+1 +1 −1 −1 −1 −1 +1 +1 +1 −1 ∈ változás műveletek.

$\log n$ hosszú bináris sorozat.

Összes $2^{\log n}$ - féle lehetőség: n

MINDEGYIKRE A MINTÁZAT MEGHATÁROZZA A MIN. HELYET

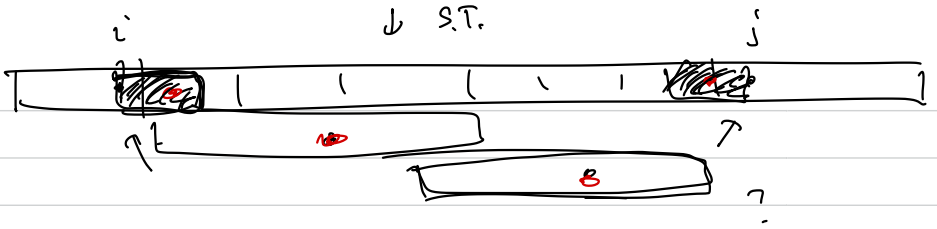
$O(n)$ időben / helyen $\log n$ hosszú rész minimum pozíciói.

Felbontjuk a vektort $(\log n)/2$ hosszú blokkokra

i, j query: i-ből j ugyanabban a blokkban a blokk eleje.

i, j különböző blokk?

$(N/\log N)$ db blokkminimum: "SPARSE TABLE": $O(N)$



$\rightsquigarrow$ $O(n)$, $O(1)$

legales , factibles