# Cel mai mic stramos comun (LCA)

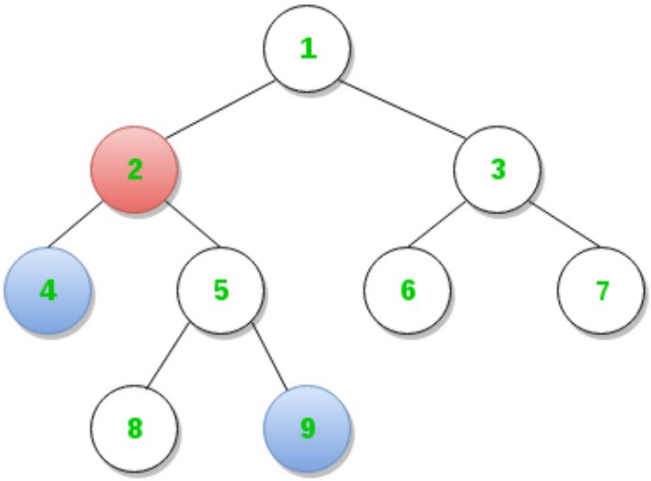
Cel mai mic stramos comun ale unor doua noduri oarecare u si v dintr-un arbore T este definit ca fiind nodul cel mai departat de nodul-root, dar care ii are si pe u si v simultan descendenti. Sunt numeroase feluri prin care se poate afla stramosul comun al unor doua noduri. Diferentierea se face prin calcularea complexitatilor lor, astfel determinandu-se cel mai eficient algoritm.

## 1.RMQ

Range Minimum Query (Interogarea minima a intervalului) este utilizat pe tablouri pentru a găsi poziția unui element cu valoarea minimă între doi indici specificați.
Timpul de preprocesare al unui segment de arbore este O (n), pe când timpul pentru interogarea minimă a intervalului este O (Logn). Spațiul suplimentar necesar este O (n) pentru a stoca arborele de segmente.

Reducand LCA la RMQ:

Ideea rezolvarii RMQ este de a traversa arborele pornind de la rădăcină printr-un tur Euler (traversarea drumului intre doua noduri fara a ridica creionul de pe foaie), care este de tip DFS cu caracteristici de traversare preordine.
Să ne imaginam: nodul nostru este nodul la cel mai mic nivel și singurul nod la acest nivel dintre toate nodurile care apar între apariții consecutive (oricare) din u și v în turul Euler din T.




Ex:
LCA intre nodurilor 4 și 9 este nodul 2, care se întâmplă să fie nodul cel mai apropiat de rădăcină dintre toate cele întâlnite între vizitele 4 și 9 în timpul unui DFS.

Pentru implementarea RMQ avem nevoie de trei tablouri:

1. Nodurile vizitate în ordine in turul Euler din arbore.
2. Nivelul fiecărui nod vizitat în turul Euler din arbore.
3. Indexul primei apariții a unui nod în turul Euler din arbore

Algoritm de implementare:

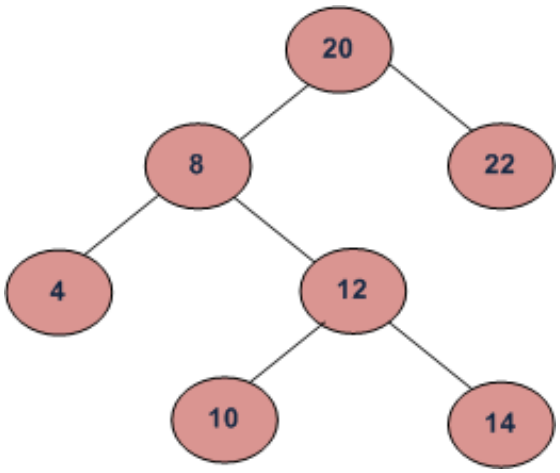
Facem un tur Euler pe arbore și completam tablourile de noduri vizitate, nivele și prima apariție.

Utilizând primul tablou de apariție, obținem indicii corespunzători celor două noduri care vor fi colțurile intervalului din tabloul de nivel care este alimentat cu algoritmul RMQ pentru valoarea minimă.
Odată ce algoritmul returnează indicele nivelului minim din interval, îl utilizăm pentru a determina LCA folosind tabloul traversat DFS.

## 2. Cautare binara a Stramosilor cu putere 2∧k (BST)

Dacă ni se oferă un BST în care fiecare nod are indicatorul părinte, atunci LCA poate fi determinat cu ușurință parcurgând cu indicatorul părinte și prin tipărirea primului nod intersectant.

Putem rezolva această problemă folosind proprietăți BST. Putem traversa recursiv BST de la root. Ideea principală a soluției este că, în timp ce parcurgem de sus în jos, primul nod n pe care îl întâlnim cu valoare între n1 și n2, adică n1 <n <n2 sau la fel ca unul dintre n1 sau n2, este LCA al n1 și n2 (presupunând că n1 <n2). Așadar, traversam BST în mod recursiv, dacă valoarea nodului este mai mare decât n1 și n2, atunci LCA-ul nostru se află în partea stângă a nodului, dacă este mai mic decât atât n1 cât și n2, LCA se află în partea dreaptă. În caz contrar, rădăcina este LCA (presupunând că atât n1 cât și n2 sunt prezente în BST).
Exemplu binary tree:




- LCA intre 10 si 14 este 12
- LCA intre 14 si 8 este 8
- LCA intre 10 si 22 este 20

Surse:
https://www.geeksforgeeks.org/lowest-common-ancestor-in-a-binary-search-tree/
https://www.geeksforgeeks.org/find-lca-in-binary-tree-using-rmq/