TRIE

Kahoot

Discuții Examen

- Ca de obicei, mă găsiți online, unde răspund la întrebări
- Pe **7 iunie ora 10** facem un Q&A
- Puteți pune întrebări aici și eventual eu voi răspunde la ele, dacă le vreți anonime puteți chiar folosi acest forms
- 4-5 studenți pe o oră
- vor exista 2 părți, care se vor desfășura oarecum în paralel:
  - Q&A despre Structuri de date și Algoritmi
  - verificarea temelor

Discuții Examen

- La examen:
  - Voi verifica parțial tema 2 (mai ales 2.3, 2.4 și 2.5). Vă pot pune întrebări din problemele pe care le aveți trimise
  - Dacă aveți surse copiate, ați picat examenul și riscati și exmatriculare
- Apoi vă voi pune întrebări despre:
  - operațiile unor structuri de date
  - complexitatea operațiilor
  - vă voi pune să-mi arătați cum se fac acele operații

Subiecte Examen

- Gen
  - Inserare în stivă,
    - cum se face
    - cum se numește în general
    - ce complexitate are
    - dacă vreau să mă uit la baza stivei, cât mă costă
  - Heap
    - inserare / căutare / ștergere
    - când e bun heap-ul
    - când nu e util ... gen când am de căutat elemente
  - Ce complexitate are RMQ? Ce face ?
  - șamd

Trie

- Am mai multe cuvinte și apoi am întrebări de genul:
  - este cuvântul în dicționar sau nu?
- Cum putem rezolva?
  - Hash-uri!
  - Cât mă costă un query?
    - O(l), unde l e lungimea cuvântului
  - Câtă memorie mă costă să reținn hash-ul?
    - O(n*l)
  - Ce credeți că am putea optimiza?
    - Memoria (poate)
    - Timpul pentru query-uri nereușite … oarecum

Trie

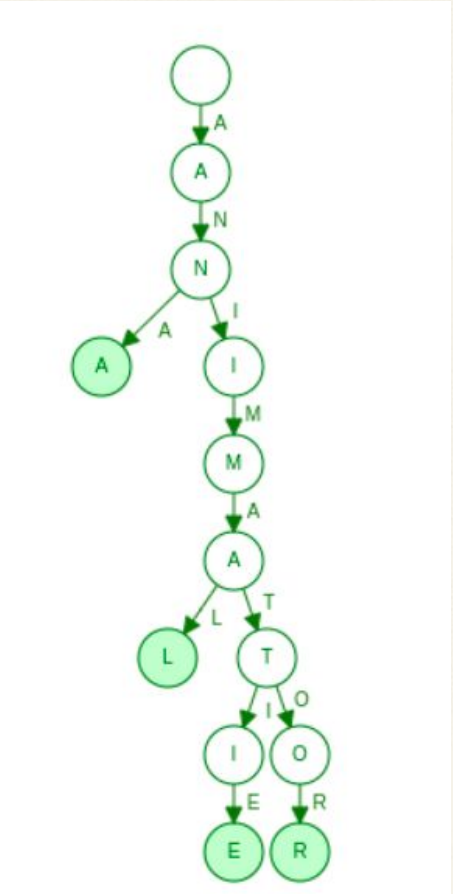
- Am mai multe cuvinte și apoi am întrebări de genul:
  - este cuvântul în dicționar sau nu?
  - care este cel mai lung prefix al cuvântului în dicționar?
- Mai merge cu hash-uri ?
  - Nu prea ...
- Alte soluții?
  - Sortăm toate cuvintele lexicografic și apoi căutăm binar
  - Ținem toate cuvintele într-un arbore binar de căutare echilibrat
- Ambele soluții au O(n*l) memorie și O(logn *l) complexitate pe search
- Arborele binar permite, totuși, și inserări și ștergeri!!

Trie

- Dacă avem cuvintele **anima, animal, animație, animator, animare,** reținem, pentru fiecare, prefixul **anim** comun
- Cum credeți că putem îmbunătăți memoria folosită?
  - Am putea, când le ținem sortate, să le ținem ceva de genul
    - anima
    - 5l
    - 5tie
  - Adică, să ținem lungimea prefixului față de elementul anterior
  - Putem duce o idee similară și spre arbori binari de căutare, dar să nu ne mai complicăm :)

Trie

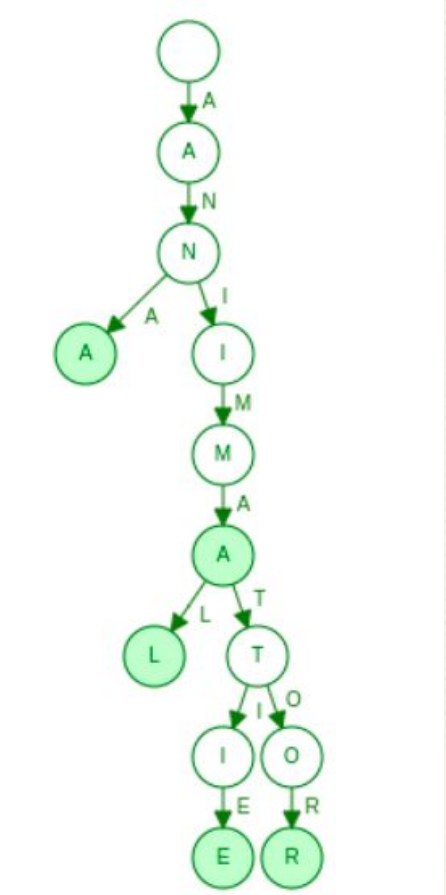
Trie cu cuvintele **ana, animator, animație, animal**



Trie

Trie cu cuvintele **ana, animator, animație, animal, anima**

vizualizare trie



Trie - Memorare

- Cum îl reținem?
  - Fiecare nod are un vector cu 26 de vecini, una pentru fiecare literă (sau mărimea alfabetului)
  - Ce facem dacă alfabetul e mare?
  - Fiecare nod ține un hash_map care pentru fiecare literă tine pointerul catre nodul cu acea litera

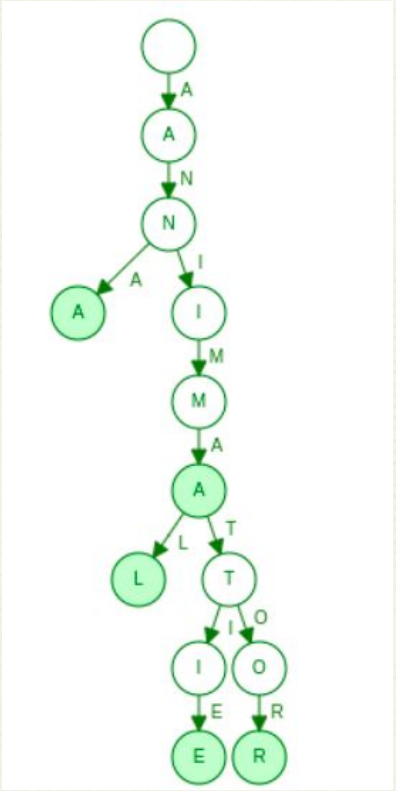
Trie - Inserare

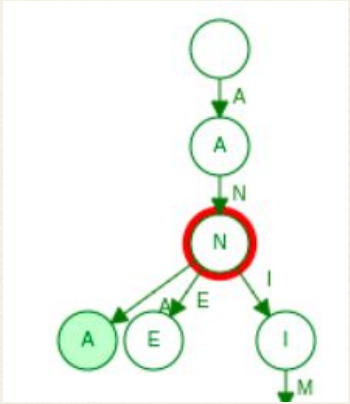
Pornim din rădăcină și, la fiecare literă, mergem în nodul corespunzător literei, eventual creăm acel nod

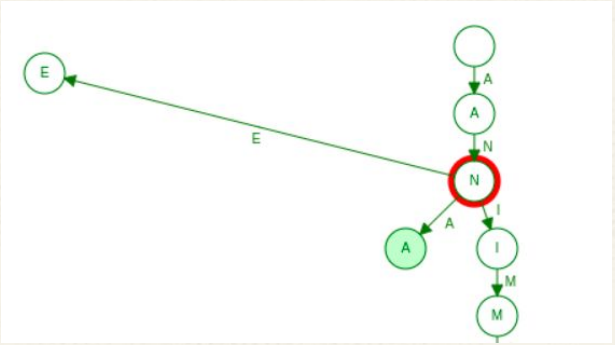
https://www.cs.usfca.edu/~galles/visualization/Trie.html

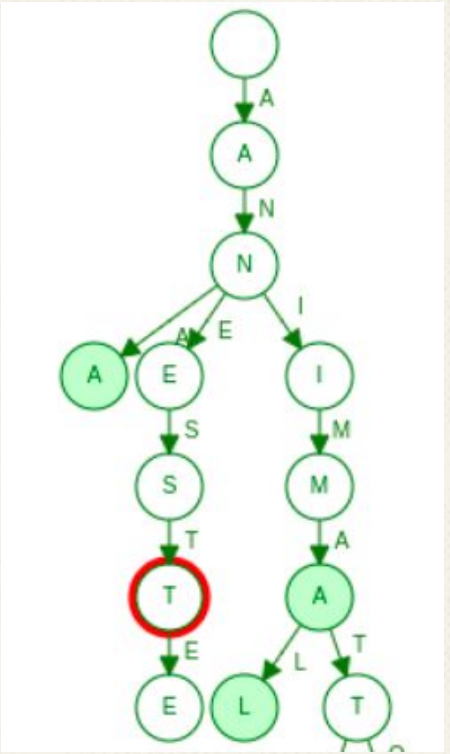
Trie - Inserare

Inserăm **anestezie**









Trie - Inserare

Complexitate: O(l)

Trie - Căutare

Pornim din rădăcină și mergem, la fiecare pas, pe litera corespunzătoare

Complexitate O(l) pentru căutare reușită

În practică, mai rapid pentru căutare ineficientă

Căutare prefix maxim:

- Căutăm elementul până nu găsim nod corespunzător acelei litere

Succes în sesiune :)