Liste dublu înlănțuite de tip XOR

O structură de date pentru a reduce spațiul de memorare pentru a stoca legăturile din nodurile unei liste dublu înlănțuite.

- Pentru memorarea legăturilor în listă, se folosește alocare dinamică a memoriei (pointeri).

- Un nod al listei nu memorează legăturile **următor** (spre următorul nod al listei) și **precedent** (spre precedentul nod al listei), ci o singură legătură **link.**

  Nod

  e: TElement {informația utilă a nodului}
  link: ↑Nod {pointer spre Nod}

- De exemplu, dacă lista este A→B→C.....atunci

  - **link**(B) = *addr*(A) XOR *addr*(C)
  - *addr* este funcția care indică adresa de memorare a unui anumit nod
  - XOR e operația **SAU exclusiv** (⊕)

Input		output
A	B	C = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

- Această convenție de memorare permite determinarea adresei de memorare a nodului următor unui anumit nod

  - De exemplu, dacă parcurgem lista de la stânga la dreapta și suntem la nodul B (deci cunoaștem adresa de memorare a nodului A), putem determina adresa *addr*(C) la care e memorat nodul C în felul următor

    - *addr*(C) = **link**(B) XOR *addr*(A)
      (= *addr*(A) XOR *addr*(C) XOR *addr*(A)

      = *addr*(A) XOR *addr*(A) XOR *addr*(C)

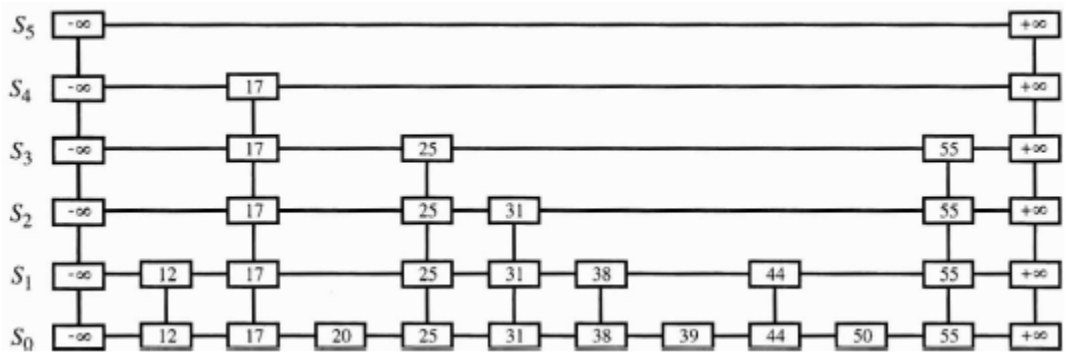
      = 0 XOR *addr*(C)

      = *addr*(C)
)

Skip Lists

O structură aleatorie (randomizată – *randomized data structure*) de stocare a datelor, eficientă pentru memorarea unui **dicționar ordonat** este **Skip List**

Ex: cheile 20, 17, 50, 44, 55, 12, 44, 31, 39,25



- Operațiile specifice (adăugare, căutare, modificare) necesită $O(\log_2 n)$ cu o probabilitate mare (în medie) - $O(n)$ caz defavorabil, dar puțin probabil să apară
- În Java există implementarea *ConcurrentSkipListMap*
- Intrările din S_{i+1} sunt alese aleator din intrările din S_i, alegând ca fiecare intrare din S_i să fie și în S_{i+1} cu probabilitatea 0.5.
- O poziție are 4 legături (*următor, precedent, sus, jos*)
- Căutare
  - Cu succes **39**: -∞, 17, 17, 25, 25, 31, 31, 38, 38, **39**
  - Fără succes **41**: -∞, 17, 17, 25, 25, 31, 31, 38, 38, **39**

- https://www.ics.uci.edu/~pattis/ICS-23/lectures/notes/Skip%20Lists.pdf
- http://web.eecs.utk.edu/~bvz/cs302/notes/skip-lists.html