UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA

DE

MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA: MATEMATICĂ-INFORMATICĂ

LUCRARE DE LICENŢĂ: SkipList, un model de structură de date probabilistic

Absolvent:

Ghadamiyan Lida

Coordonator ştiinţific:

Prof. Dr. Popescu Ionel

BUCUREŞTI

Iunie 2020

UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA

DE

MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA: MATEMATICĂ-INFORMATICĂ

LUCRARE DE LICENŢĂ: SkipList, un model de structură de date probabilistic

Absolvent:

Ghadamiyan Lida

Coordonator ştiinţific:

Prof. Dr. Popescu Ionel

BUCUREŞTI

Iunie 2020

Rezumat

Aceasta lucrare prezintă importanța și aplicabilitatea matematicii, mai exact a teoriei probabilităţilor si a statisticii în informatică - algoritmi și structuri de date, având la bază articolul în care W. Pugh a introdus o nouă structură de date numită skip list.

Se va demonstra complexitatea timp a acestei structuri, prezentându-se marginea superioară obținută de W. Pugh în 1990 în [Pug90a] și rezultatul exact al acestei valori, dat de T. Papadakis în 1993 în [Pap93]. Aceste valori au fost găsite pornind de la cele mai pesimiste ipoteze, dovedindu-se a fi echivalente din punct de vedere al vitezei de execuție cu arborii binari de căutare, fiind în final mai ușor de folosit, neavând nevoie să fie reechilibrați. Simplitatea structurii skip list reiese din asemănarea cu listele, acest aspect fiind realizat prin modul de construcție.

Există varietăți ale acestei structuri, ele fiind modificate în funcție de scopul pentru care urmează a fi folosite. Printre acestea se numără skip list determinist, acesta fiind, la rândul său de mai multe feluri, și skip list pentru date multidimensionale, ce are la baza, fie un skip list clasic, fie unul determinist. Având în vedere scopul principal al acestei lucrări, și anume ilustrarea importanței matematicii în vederea eficientizării procesului de automatizare, prezentarea tuturor variantelor de skip list ar fi redundantă, abătându-ne de la subiect. Aceste varietăți vor avea o descriere succintă, fiind prezentate pe parcursul lucrării suficient încât să se contureze o comparație cu skip list clasic, din punct de vedere matematic. O atenție deosebită va fi acordată structurii skip list pentru date multidimensionale, deoarece consider că acestă variantă a structurii se va integra cel mai bine în rezolvarea sarcinilor de tip computațional, fiind mai practică în circumstanțele realității.

Summary

This paper displays the importance and the applicability of mathematics, more precisely, the theory of probabilities and statistics in computer science - algorithms and data structures, based on the article where W. Pugh introduced a new data structure named skip list.

The time complexity of this structure will be demonstrated, specifying the upper limit obtained by W. Pugh in 1990 [Pug90a] and the exact result of this value, given by T. Papadakis in 1993 [Pap93]. These values were found starting from the most pessimistic hypothesis, proving to be equivalent in terms of execution speed with binary search trees, being, in the end, easier to use without needing to be self-balanced. The simplicity of the skip list structure results from the resemblance to the lists, this aspect being achieved through the construction mode.

There are varieties of this structure, these being modified according to the purpose for which they are going to be used.They include the deterministic skip list, which can also be of several types, and the skip list for multidimensional data based on either a classic skip list, either on a deterministic one. Considering the aim of this paper, precisely illustrating the importance of mathematics in order to make the automation process more efficient, the presentation of all kinds of skip lists would be redundant, deviating from subject.

These will have a succinct description, being sufficiently presented throughout this paper in order to outline the comparison to the classical one, mathematically. The multidimensional skip list structure will have a considerable attention due to the fact that I consider that this kind of structure will be best integrated in solving computational tasks, being more practical in the circumstances of reality.

Cuprins

Listă de figuri

Introducere

Aceasta este o lucrare de sinteză în domeniul matematic și informatic. Tema lucrării constă în prezentarea unei variante probabilistice de structură de date, ce îmbină avantajul complexității cu simplitatea.

Motivația alegerii acestei teme se regăsește în prezentarea manierei în care modelele matematice intervin în săvârșirea sarcinilor computaționale, ce stau la baza automatizării diverselor nevoi ale societății actuale, în special a celor ce necesită reținerea datelor. Un motiv suplimentar al acestei alegeri constă în actualitatea subiectului, acesta fiind din ce în ce mai exploatat.

Lucrarea debutează prin definirea termenilor, noțiunilor și fundamentelor matematice și informatice folosite ulterior. Printre acestea se numără și descrierea succintă a listelor și arborilor, în scopul comparației cu skip list.

Componenta principală a lucrării se constituie în prezentarea structurii skip list, și a eficienței acesteia urmând rezultate existente, prezentate într-o manieră proprie, atât din punct de vedere al ordinii parcurgerii pașilor necesari în vederea obținerii rezultatului final, cât și a unor demonstrații, expunând totul în modul în care am considerat că este cel mai potrivit pentru a pune în valoare factorul matematic.

În final, ultima parte a lucrării prezintă skip list pentru date multidimensionale, acesta fiind, în opinia mea, o varietate a acestei structuri ce se mulează cel mai bine pe sarcinile care trebuie a fi automatizate.

Studiul acestei teme este de actualitate, fiind introdus de Pugh în 1989, și revizuit în 1990 în [Pug90b]. Skip list devine din ce în ce mai folosit în industrie, în diverse forme, întrucât poate înlocui cu succes alte structuri de date.

Capitolul 1

Preliminarii

1.1 Structuri Asemănătoare

Nevoia de a eficientiza automatizarea sarcinilor de tip computațional este foarte mare, întrucât acestea înlocuiesc procese greu de realizat manual, precum ținerea evidenței oamenilor în instituții, ca și cele de învățământ, spitale, sisteme de adopții etc.

În vederea efectuării unor algoritmi ajutători acestor tipuri de activități este nevoie de structuri de date cât mai eficiente. Aceste structuri au la baza un set, adică o colecție ordonată de elemente, în scopul reținerii datelor într-un mod organizat, devenind astfel ușor de manipulat. Pentru realizarea acestui scop, este necesar să avem abilitatea de a insera, șterge și căuta elemente. Un set ce prezintă calitățile enumerate anterior, se numește un dicționar. Fiecare element al unui dicționar conține mai multe câmpuri (datele necesare în vederea executării sarcinii), dintre care un câmp numeric unic, numit valoare cheie, după care se vor face operațiile de căutare, ștergere și inserare.

Evaluarea unei astfel de structuri se realizează prin analiza simplității și a complexității timp și spațiu (timpul de execuție), care se definesc după cum urmează. Complexitatea timp a algoritmului constă în timpul necesar rulării acestuia, fiind reprezentată de de numărul de operații pe cheie necesare (precum comparațiile). Complexitatea spațiu

reprezintă cantitatea de memorie necesară efectuării calculelor, iar în cazul structurilor prezentate în această lucrare, se definește ca fiind numărul de celule de memorie utilizate. În continuare, vom nota complexitatea cu O(-), iar, spre exemplu O(n) înseamnă ca au fost efectuate n operații pentru ca sarcina să fie îndeplinită, sau au fost necesare n celule de memorie.

În timp, au apărut mai multe structuri ce pot reprezenta un dicționar, unele fiind simple din punct de vedere al implementării dar slabe în complexitate, iar altele, care prezintă o complexitate bună să fie dificil de folosit. Printre alternativele apărute se numără listele înlănțuite și arborii, care exemplifică perfect situația descrisă, întrucât, listele fiind foarte simplu de implementat sunt cu mult depășite în complexitate de către arbori, despre care, o părere generală este ca sunt dificil de programat.

1.1.1 Liste înlănțuite

Listele înlănțuite sunt printre primele structuri de date apărute. Acestea reprezintă o înlănțuire de noduri, legate prin atribuirea fiecărui nod adresa de memorie (pointer) către următorul, precum în figura alăturată.

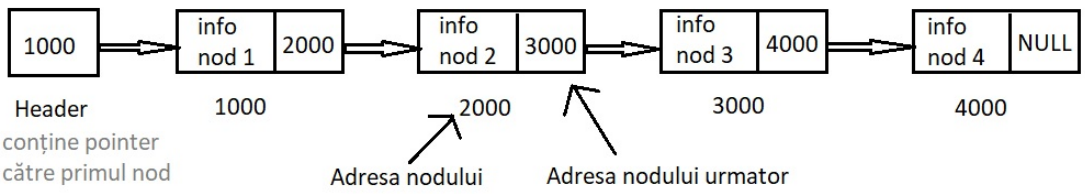

Figura 1.1: Listă simplu înlănțuită

Astfel, fiecare nod va avea două componente, și anume: info și adresa-următor.

Inserarea se face prin modificarea câmpului adresa-următor al nodului din fața celui ce urmează adăugat, astfel încât să aibă valoarea adresei nodului nou, iar acesta v-a prelua adresa-următor pe care nodul precedent a modificat-o, astfel:

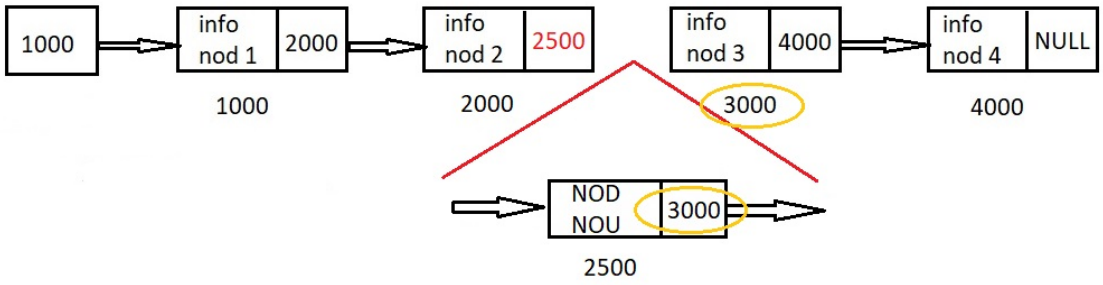

Figura 1.2: Inserare în listă

Pentru a șterge un element, trebuie sa găsim nodul din fața acestuia și să modifică adresa nodului următor, ocolind nodul care se dorește a fi șters, apoi să eliberăm memoria alocată acestuia.

Astfel, pentru adăugarea și ștergerea elementelor, fiind suficient doar sa alocăm/eliberăm memorie și să schimbăm valorile câmpului adresa-următor, aceasta fiind singura operație efectuată, în termeni de complexitate, aceste operații costa O(1). Dar, ca să ștergem/adăugăm noduri, este nevoie ca mai întâi să găsim nodul pe care dorim să îl ștergem, sau locul în care dorim să adăugăm. Deci, pentru a căuta nodul de pe poziția k, trebuie sa trecem prin k noduri, și astfel să facem k comparații intre valorile cheie ale fiecărui nod. Așadar, costul operației de căutare al nodului k o sa fie O(k). În această manieră, este clar, ca în cel mai ineficient caz va trebui sa căutam ultimul nod, deci, complexitatea, în cel mai rău caz, pentru operația de căutare este O(n), unde n este numărul ttoal de noduri.

Acești algoritmi vor fi de folos pe parcursul lucrării, întrucât aceștia stau la baza inserării și ștergerii în cazul structurilor skip list.

1.1.2 Arbori echilibrați

Un arbore este o structură de date neliniară, formată dintr-un nod părinte care are subarbori ce au aceeași configurație de arbore. Dacă adăugăm condiția ca fiecare nod trebuie să aibă exact doi sub-arbori, cu excepția celor de pe ultimul rând, care pot avea și câte unul, obținem o structură numită arbore binar. În această structură nu se pot efectua căutări deoarece nu există o relație de ordine. Așadar, dacă stabilim ca în sub-arborele stâng al oricărui nod se vor afla noduri cu valori mai mici decât a acestuia, iar cele mai mari în sub-arborele drept, obținem un arbore binar de căutare. Această structură nu se va păstra în urma operațiilor de inserare și ștergere, fapt pentru care, arborele trebuie reechilibrat de fiecare dată.

Înălțimea unui arbore reprezintă numărul de noduri din cel mai lung drum de la rădăcină (primul nod părinte) până la nodurile de pe ultimul nivel.

Un arbore echilibrat este un arbore binar de căutare a cărui înălțime este mai mică sau egală cu $log_2(n)$, unde n este numărul de noduri.

În vederea căutării unui nod, la fiecare pas va trebui să se compare nodul curent cu nodul scop pentru a se decide în care sub-arbore se va continua căutarea. În cel mai nefavorabil caz, nodul scop se va afla la distanță maximă față de rădăcină. Am stabilit ca această distanță este chiar înălțimea arborelui. Având în vedere faptul ca arborele este binar, înălțimea sa este maxim $log_2(n)$. Așadar, în cel mai ineficient caz, costul unei căutări este O($log_2(n)$).

Arborii prezintă eficiență mult mai bună decât listele, însă, acest rezultat aduce dificultăți care constau în reechilibrarea structurii la fiecare operație.

1.2 Obiective

O descriere scurta care sa înglobeze, la prima vedere, motivele folosirii unui skip list în detrimentul celorlalte structuri prezentate, este cea incipienta articolului lui Pugh [Pug90b].

Acesta evidențiază faptul că principalul punct slab al arborilor este nevoia de reechilibrare după fiecare inserare sau ștergere, întrucât aceasta se face algoritmic, forțând de fiecare dată modificarea structurii. În cazul listelor, această problemă nu apare datorită structurii liniare a acesteia.

Astfel, o alternativă ideală ar fi o structură liniară la bază, precum o listă, pentru a asigura ordinea elementelor, dar cu o căutare rapidă precum a arborilor. În acest mod, căutarea ar fi eficientă, iar ștergerea și inserarea ar fi ușor de executat, precum în cazul listelor. Concluzia obținută de Pugh este că dificultatea reechilibrării este eliminată dacă aceasta s-ar face probabilistic, fără ca un algoritm să fie necesar.

Scopul acestei lucrări este de a prezenta felul în care includerea factorului matematic în domeniul informaticii, stă la baza eficientizării procesului de reținere a informației. În acest mod, a fost creată o structură de date care sa unifice avantajele listelor și arborilor.

În vederea realizării acestui obiectiv, se va demonstra complexitatea timp a operațiilor efectuate pe un skip list, simplitatea realizării unei astfel de structuri fiind evidențiată din figurile grafice si modul de construcție. Se vor aminti si alte tipuri de skip list, o atenție deosebită fiind acordată structurii de tip skip list pentru date multidimensionale.

1.3 Fundamentele teoretice folosite

Se vor folosi pe parcursul acestei lucrări anumite noțiuni matematice, ce urmează a fi notate în această secțiune.

Definiţia 1.3.1. Fie (Ω, E, P) spațiu de probabilitate, cu Ω fiind spațiul de stări, E $\sigma-$ algebră de evenimente și P : E $\rightarrow$ [0,1] funcția de probabilitate. $X : \Omega \rightarrow \mathbb{R}^n$ se numește variabilă aleatoare dacă este funcție măsurabilă, i.e. $\{X \in A\} \in E$ $\forall$ $A \in \mathcal{B}(\mathbb{R}^n)$, unde $\mathcal{B}$ reprezintă mulțimea borelienelor .

În cele ce urmează, vom lucra doar cu variabile aleatoare discrete, i.e. care iau un număr finit de valori.

Definiţia 1.3.2. Fie X o variabilă aleatoare. Funcția cumulativă de distribuție a lui X evaluată în k este $F_X(k) = P(X \leq k)$.

Din ultima definiție, se deduce că P(X > k) = 1 - $F_X(k)$.

Definiţia 1.3.3. Fie X o variabilă aleatoare. Definim media lui X ca fiind $E[X] = \sum_{k \geq 1} kP(X = k)$.

Definiţia 1.3.4. Fie X o variabilă aleatoare distribuită Binomial(n,p). X reprezintă numărul de evenimente reușite din n încercări. $P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$.

Definiţia 1.3.5. Fie X o variabilă aleatoare distribuită Geometric(n,p). X reprezintă numărul de încercări până la primul eșec. $P(X = k) = p^{k-1}(1-p)$.

Se vor folosi, de asemenea, proprietatea de absorbție a binomialei $\binom{n}{k}\frac{k}{n} = \binom{n-1}{k-1}$, binomul lui Newton $(x-y)^n = \sum_{k=0}^{n} \binom{n}{k}(-1)^k y^k x^{(n-k)}$, suma seriei geometrică, $\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}$ și $\sum_{k=0}^{n} r^k = \frac{1-r^{n+1}}{1-r}$

Capitolul 2

Skip List

2.1 Prezentarea Structurii

Procesul prin care skip list a fost creat este prezentat de Pugh [Pug90b], în felul următor. Acesta a pornit de la o listă simplu înlănțuită, adăugând pe rând cate un pointer suplimentar, mai întâi nodurilor plasate pe poziții pare, apoi celor plasate pe poziții ce erau multiplu de 2^2, și așa mai departe . În această manieră se obține o listă în care nodurile aflate la o înălțime i, au drum până la un nod la distanță 2^i. Astfel, timpul de căutare nu depășește $log_2(n)$. Deși complexitatea timp este bună în acest caz, apar două probleme.

În primul rând, va crește numărul de celule de memorie necesare. Pentru o listă cu n noduri, se vor adăuga $\sum_{n=1}^{[log_2(i)]}[\frac{n}{2^i}]$ celule.

$$\left[\frac{n}{2}\right] \leq \sum_{n=1}^{[log_2(i)]}\left[\frac{n}{2^i}\right] \leq \sum_{n=1}^{log_2(i)}\frac{n}{2^i} = \frac{n(1-\frac{1}{2^{log_2(n)}})}{\frac{1}{2}} = 2n-1$$

Marginea inferioară este întrecută mereu, deoarece aceasta reprezintă primul strat de pointeri care sunt adăugați.

Valoarea exactă a sumei este dată de Graham, Knuth și Patashnik în "Concrete Mathematics" [GKP94], ca fiind n - ν(n), unde $\nu(n)$ este numărul de cifre de 1 din reprezentarea

binară a lui n. În aceste calcule, nu s-a ținut cont de numărul de pointeri din header, întrucât, aceștia fiind $log_2(n)$, nu modifică complexitatea.

Al doilea impediment constă în dificultatea de a păstra structura după fiecare inserare sau ștergere, ceea ce este mai dificil decât reechilibrarea arborilor.

Pugh a propus păstrarea numărului de pointeri adăugați, structura urmând să aibă același număr de pointeri pe etaj, dar distribuiți aleatoriu. Astfel, numărul pointerilor inserați pe nod vor fi variabile aleatoare independente distribuite geometric. Astfel, modificările apărute în urma inserării și a ștergerii s-ar produce local, nu pe toată structura, iar numărul de pointeri alocați, în medie, unui nod ar depinde de probabilitatea aleasă.

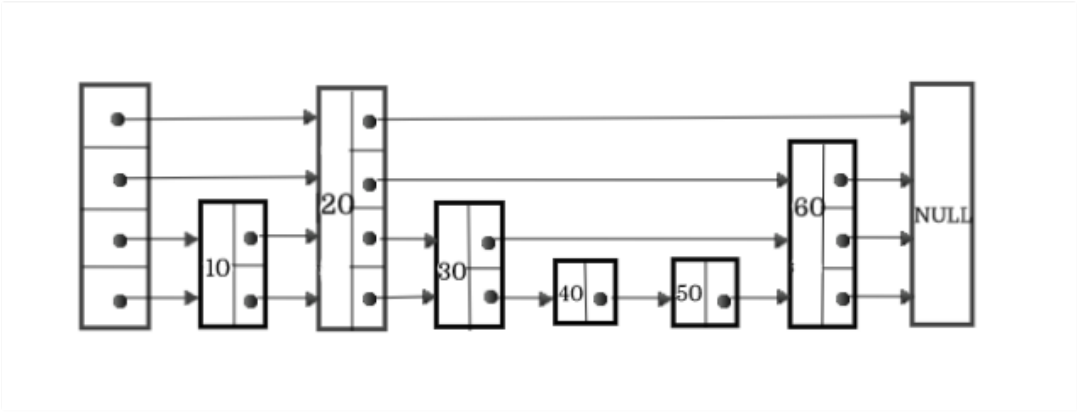

Figura 2.1: Skip list cu pointerii adăugați cu probabilitate de 0.5

2.2 Algoritmi

Skip list s-a dorit a fi o structură care să execute ștergerile și adăugările la fel de simplu precum o listă, iar căutarea optimă precum cea a unui arbore. Astfel, operațiile de inserție și ștergere se execută precum în cadrul unei liste. Algoritmic, schimbările se petrec ca și cum ar fi mai multe liste suprapuse, fiind astfel nevoie să modificăm mai multe adrese de memorie. Căutarea, în schimb, are loc mult mai eficient, având posibilitatea de a sări peste elemente, nefiind nevoie ca toate să fie evaluate.

Căutarea începe de la cel mai înalt nivel al structurii, verificând fiecare element din dreapta sa. Dacă acesta nu este mai mare, se merge în jos, la un nivel mai mic. Se continuă în această manieră până când evaluarea ajunge pe primul nivel. În acest stadiu se verifică dacă elementul curent este sau nu egal cu cel căutat.

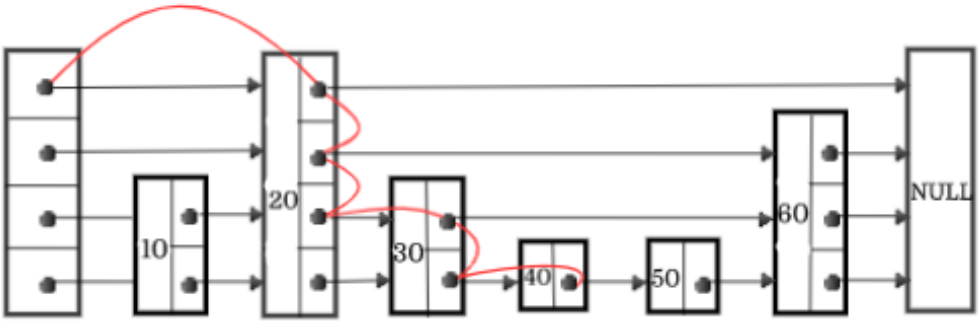

Figura 2.2: Căutarea elementului 40 în skip list

În cazul inserării unui nou element, algoritmul constă în căutarea locului în care acesta trebuie adiționat, adică, găsirea celui mai mare element mai mic decât cel ce urmează a fi inclus. Acest lucru este echivalent cu a căuta valoarea ce se dorește introdusă, deoarece algoritmul de căutare se oprește când un element mai mare este găsit, ceea ce garantează ca locul în care căutarea va lua sfârșit este cel potrivit adăugării. În plus, în acest mod se verifică și dacă elementul ce urmează a fi inserat există deja sau nu în structură. După

ce este efectuată operația de căutare, nodul se adaugă în același mod precum în cazul listelor. Înălțimea acestui nod reprezintă o variabilă aleatoare de distribuție geometrica de probabilitate p. În cazul structurii din figură, această probabilitate este 0.5.

Astfel, a alege înălțimea, în acest caz, este echivalent cu a spune că dăm cu banul până când pica cap. Numărul de aruncări până să apară cap, va reprezenta înălțimea nodului nou inserat. În Figura 2.3 sunt ilustrate, cu culoarea roșie modificările aduse de adăugarea nodului 35.

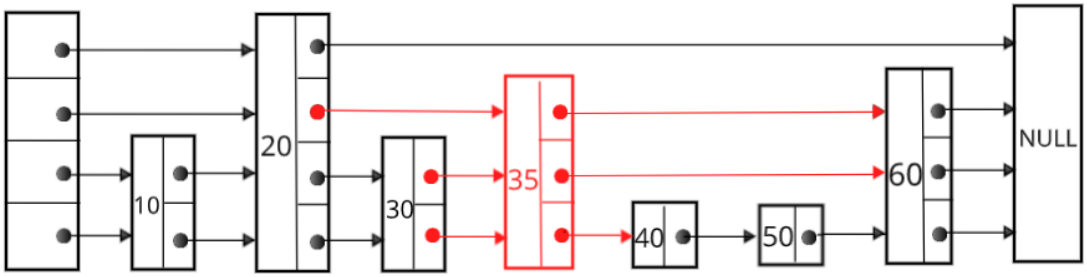

Figura 2.3: Inserarea elementului 35 în skip list conform cu Figura 2.2

Ștergerea, începe, de asemenea, prin căutarea elementului ce se dorește a fi înlăturat, iar procesul propriu zis de eliminare este analog celui prezentat la liste.

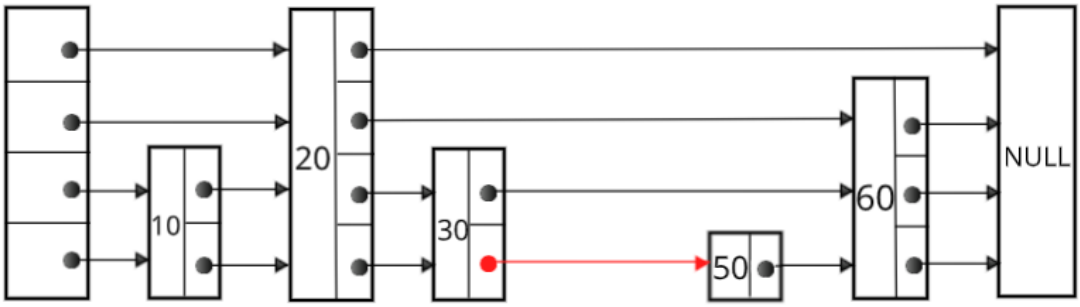

Figura 2.4: Ștergerea elementului 40 în skip list conform cu Figura 2.2

O primă observație este că structura obținută este, structural, echivalentă cu cea de la care s-a pornit, fără să fie nevoie de alte modificări, atât în cazul adăugării, cât și în

cel al ștergerii de elemente, indiferent de ordine sau frecvență.

O altă observație constă în faptul că în situațiile ilustrate anterior nu a fost șters sau adăugat un element de înălțime maximă. Structura de skip list, va reține o variabilă ce va defini cel mai mare nivel, ceea ce va permite adăugarea sau ștergerea ultimului nivel și în header. Astfel că și aceste situații vor fi tratate in aceeași manieră.

În continuare, în vederea analizei acestor algoritmi, se vor considera cele mai nefavorabile cazuri (structură de dimensiune infinită), exceptând situația în care niciun element al structurii nu "crește". În această situație, structura ar deveni o listă înlănțuită, iar costurile s-ar reduce la O(n). Aceste circumstanțe, au de asemenea, o probabilitate aproape nulă de a se întâmpla, deoarece, numărul de noduri este invers proporțional cu probabilitatea de a avea un skip list plat, și anume $(1-p)^n \to 0$ pentru p $\in (0,1)$ și n $\to \infty$.
Se va păstra această presupunere, a unei structuri diferite de o listă înlănțuită, și pentru următorul capitol, skip list pentru date multidimensionale, motivele fiind aceleași.

2.3 Analiza probabilistică a structurii

Întrucât ștergerea și inserarea se bazează pe căutarea elementului, se va urmări analiza complexitatății operației de căutare. Se va prezenta această analiză probabilistică urmărind pașii lui Pugh în "A Skip List Cookbook" [Pug90a], și continuând cu expunerea rezultatului exact a lui Papadakis [Pap93].

După cum am stabilit în preliminarii, complexitatea reprezintă numărul operațiilor elementare necesare în vederea efectuării acțiunilor, deci, pentru a căuta un element, vom defini costul ca fiind numărul de comparații necesare pentru a ajunge la nodul dorit. Astfel, acest cost este echivalent cu numărul nodurilor examinate în drumul până la nodul final (fără acesta).

Pugh a propus o metodă de a găsi un majorant al costului mediu, pornind de la finalul drumului, împărțind această cale în mutări în sus și către stânga, considerând o structură cu n noduri, număr ce tinde către infinit. Notăm cu C(k) costul unei căutări ce conține k nivele și cu p probabilitatea cu care este stabilită înălțimea fiecărui nod. Ținând cont că în orice etapă a drumului va trebui să mergem în sus (cu probabilitate p), sau în stânga (cu probabilitate 1-p), avem următoarele relații.

$$
\begin{aligned}
& C(k) = p(1 + C(k-1)) + (1-p)(1 + C(k)) \\
\Leftrightarrow \quad & pC(k) = 1 + pC(k-1) \\
\Leftrightarrow \quad & C(k) = \frac{1}{p} + C(k-1) \\
\Leftrightarrow \quad & C(k) = \frac{k}{p} \qquad (2.1)
\end{aligned}
$$

Având în vedere presupunerea conform căreia lista se continua la infinit, nu putem folosi C(k) pentru întreaga căutare, ci doar până la un nivel pe care se afla un număr de noduri ce nu depinde de numărul de noduri. Vom nota acest nivel cu L(n), nivel pe care se află $\frac{1}{p}$ noduri.

Probabilitatea ca un nod să existe și la nivelul l este $p^{(l-1)}$. Numărul de noduri aflate pe un nivel l al structurii este o variabilă aleatoare distribuită Binomial(n,$p^{(l-1)}$), deoarece acesta reprezintă numărul de reușite dintr-o secvență de n încercări cu probabilitatea dedusă mai sus. Media acestei variabile este $np^{(l-1)}$, valoare ce semnifică numărul de noduri pe care ne așteptăm să îl avem la un nivel l al structurii.

Așadar, înlocuind în relația de mai sus numărul de noduri de pe un nivel cu $\frac{1}{p}$, putem obține L(n).

$$\begin{aligned}
& \frac{1}{p} = np^{(L(n)-1)} \\
\Leftrightarrow \quad & p^{-L(n)} = n \\
\Leftrightarrow \quad & ln((\frac{1}{p})^{L(n)}) = ln(n) \\
\Leftrightarrow \quad & L(n)ln(\frac{1}{p}) = ln(n) \\
\Leftrightarrow \quad & L(n) = \frac{ln(n)}{ln(\frac{1}{p})} \\
\Leftrightarrow \quad & L(n) = log_{\frac{1}{p}}(n) \qquad (2.2)
\end{aligned}$$

Pentru a afla costul final al unei căutări, trebuie calculat costul drumului rămas de la L(n) până la Header. Vom considera această valoare ca fiind numărul de noduri rămase în listă, mai sus de L(n).

Știind că fiecare nod avansează un nivel cu probabilitatea p, numărul de noduri rămase la un nivel k + L(n) este $\frac{1}{p}p^{(k-1)}$. Astfel, obținem numărul total de noduri peste L(n) sumând nodurile de pe fiecare nivel mai mare decât L(n) până la infinit.

$$\sum_{k=1}^{\infty} p^{(l-1)} = \frac{1}{1-p} \qquad (2.3)$$

Așadar, din (2.1), (2.2) și (2.3) obținem costul unei căutări sumând costul drumului până la nivelul L(n) cu nodurile rămase de la acel nivel în sus.

$$Cost \leq C(L(n)) + \frac{1}{1-p} = \frac{L(n)}{p} + \frac{1}{1-p} = \frac{(1-p)log_{\frac{1}{p}}(n) + p}{p(1-p)}$$

Astfel, costul unei căutări într-un caz nefavorabil, reprezentat de o structură cu o infinitate de elemente nu depășește O(log(n)).

Continuăm cu prezentarea exactă a costului, calculat de T. Papadakis în "Skip Lists and Probabilistic Analysis of Algorithms" [Pap93].

Căutarea unui element se poate finaliza prin a găsi nodul dorit, sau prin a ajunge la cel mai mare element din listă, care este mai mic decât elementul dorit (i.e. locul în care nodul ar trebui să fie adăugat). Astfel, în a doua situație rezultatul va fi cu 1 mai mare decât în prima. În cazul în care elementul căutat este cel mai mare din listă (conține n elemente), poziția returnată va fi n+1, aferentă nodului ce reprezintă finalul ($+\infty$) . Astfel, se definește următoarea variabilă aleatoare.

$C_n^{(m)}$ =costul căutării elementului de pe poziția m, când acesta este în listă (i.e. $1 \leq m \leq$ n)

costul căutării elementului de pe poziția m când acesta nu este în listă (i.e. $1 \leq m \leq$ n+1)

Pentru m = 1, i.e. căutarea primului element, vom nota $C_n^{(1)}$ cu T_n, variabilă ce reprezintă înălțimea structurii, deoarece, pentru a ajunge la acest element, se va parcurge Headerul, de la nivelul maxim, până pe primul, fără nicio mutare orizontală. În cazul în care m = n+1, $C_n^{(n+1)}$ va fi notat cu F_n, fiind costul căutării cheii finale ($+\infty$), notată cu "NULL".

Alte variabile ce urmează a fi folosite sunt S_n și U_n, reprezentând media empirică a costurilor căutării unei chei, în cazul în care aceasta există, respectiv, când aceasta nu există.

$$S_n = \frac{1}{n}\sum_{m=1}^{n} C_n^{(m)} \qquad U_n = \frac{1}{n+1}\sum_{m=1}^{n+1} C_n^{(m)}$$

Divizăm drumul parcurs în căutarea unei valori, precum în Figura2.5, în mutări verticale și orizontale, și notăm cu L_{m-1} numărul mutărilor orizontale efectuate pentru a ajunge la nodul de pe poziția m-1. Având în vedere faptul că procesul de căutare se oprește când se ajunge pe primul nivel, rezultă că, în orice căutare vor fi efectuate T_n mutări verticale. Așadar, costul va consta în înălțimea listei alături de L_{m-1}

$$\begin{aligned} & C_n^{(m)} = T_n + L_{m-1} \\ \Rightarrow \quad & E[C_n^{(m)}] = E[T_n] + E[L_{m-1}] \end{aligned} \tag{2.4}$$

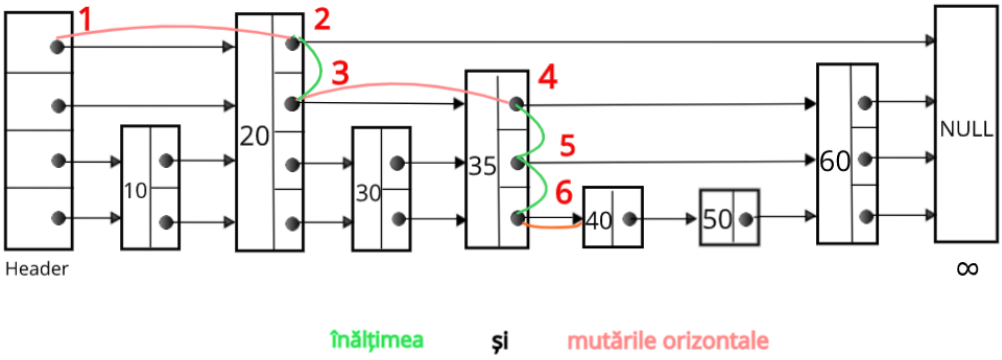

Figura 2.5: Costul căutării lui 40 sau al unui nod inexistent între 36 si 40

Propoziţia 2.3.1. *Valoarea medie a înălțimii unui skip list cu n elemente și probabilitate p este*

$$E[T_n] = \sum_{k \geq 1} [1 - (1 - p^{(k-1)})^n]$$

Demonstraţie.

$$\begin{aligned} E[T_n] &= \sum_{k \geq 1} kP(T_n = k) \\ &= P(T_n = 1) + \end{aligned}$$

$$
\begin{aligned}
&P(T_n = 2) + P(T_n = 2) + \\
&P(T_n = 3) + P(T_n = 3) + P(T_n = 3) + ... \\
&= P(T_n \geq 1) + P(T_n \geq 2) + P(T_n \geq 3) + ... \\
&= \sum_{k\geq 1} P(T_n \geq k) \quad = \sum_{k\geq 1} [1 - P(T_n \leq k-1)]
\end{aligned}
$$

Fie H_i, cu $1 \leq i \leq$ n înălțimea nodului de pe poziția i. Cum T_n este înălțimea structurii, avem că $T_n = max_{i\leq n} H_i$.

$$
\begin{aligned}
E[T_n] &= \sum_{k\geq 1} [1 - P(max_{i\leq n} H_i \leq k-1)] \\
&= \sum_{k\geq 1} [1 - P(H_1 \leq k-1, H_2 \leq k-1, ...)]
\end{aligned}
$$

Știind că H_i sunt variabile aleatoare independente și identic distribuite geometric, avem că

$$
\begin{aligned}
P(H_i \leq k-1) &= (1-p) + (1-p)p + ... + (1-p)p^k - 2 \\
&= (1-p)\frac{1-p^{(k-1)}}{1-p} = 1 - p^{(k-1)} \text{ pentru orice i} \in \{1, ..., n\} \\
\Rightarrow E[T_n] &= \sum_{k\geq 1} [1 - P(H_1 \leq k-1) P(H_2 \leq k-1) ...] \\
&= \sum_{k\geq 1} [1 - (1 - p^{(k-1)})^n]
\end{aligned}
$$

□

Propoziţia 2.3.2.

$$
E[T_n] = \sum_{k\geq 1} [k((1-p^k)^n - (1-p^{(k-1)})^n)]
$$

Demonstraţie. Aplicând Binomul lui Newton pentru termenul stâng, putem ajunge la concluzie.

$$
\begin{aligned}
&\sum_{k\geq 1} [k((1-p^k)^n - (1-p^{(k-1)})^n)] \\
&= \sum_{k\geq 1} k \left[\sum_{j=0}^{n} \binom{n}{j} (-1)^j p^{kj} - \sum_{j=0}^{n} \binom{n}{j} (-1)^j p^{(k-1)j} \right]
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{j=1}^{n} \binom{n}{j} (-1)^j \left[\sum_{k\geq 1} k p^{jk} - \sum_{k\geq 1} k p^{(k-1)j} \right] && \text{(Termenul j=0 se reduce)} \\
&= \sum_{j=1}^{n} \binom{n}{j} (-1)^j \left[\frac{p^j}{(p^j-1)^2} - \frac{1}{(p^j-1)^2} \right] \\
&= \sum_{j=1}^{n} \binom{n}{j} (-1)^j \frac{1}{p^j-1} \\
&= \sum_{j=1}^{n} \binom{n}{j} (-1)^{j+1} \sum_{k\geq 1} p^{(k-1)j} && \text{(Serie geometrică aplicată invers)} \\
&= \sum_{k\geq 1} \left[1 - \sum_{j=0}^{n} \binom{n}{j} (-1)^j p^{(k-1)j} \right] && \text{(Introducem termenul j=0)} \\
&= \sum_{k\geq 1} \left[1 - (1-p^{(k-1)})^n \right]
\end{aligned}
$$

□

Propoziţia 2.3.3. *Numărul mediu al mutărilor orizontale în vederea căutării elementului de pe poziția (m-1) într-un skip list cu n elemente și probabilitate p este*

$$
E[L_{m-1}] = \begin{cases} 1 + \frac{(1-p)}{p} \sum_{j=1}^{m-2} \sum_{k\geq 1} p^k (1-p^k)^j & \text{dacă } m \neq 1 \\ 0 & \text{dacă } m = 1 \end{cases}
$$

Demonstraţie. Fie variabila aleatoare I_j, cu j=2,...,m-2, definită astfel

$$
I_j = \begin{cases} 1 & \text{dacă nodul de pe poziția j se află în drumul către nodul căutat} \\ 0 & \text{altfel} \end{cases}
$$

Numărul mutărilor orizontale este egal cu numărul nodurilor prin care are loc căutarea, deci acesta se poate exprima în funcție de I_j în modul următor

$$
L_{m-1} = \begin{cases} 1 + \sum_{j=1}^{m-2} I_j & \text{dacă } m \neq 1 \\ 0 & \text{dacă } m = 1 \end{cases}
$$

$$\Leftrightarrow E[L_{m-1}] = \begin{cases} 1 + \sum_{j=1}^{m-2} E[I_j] & \text{dacă } m \neq 1 \\ 0 & \text{dacă } m = 1 \end{cases} \tag{2.5}$$

Fie H_k, cu $1 \leq k \leq$ m-2 înălțimea nodului de pe poziția k. Pentru ca un nod aflat pe poziția j să existe în drumul către nodul căutat, acesta trebuie să fie cel mai înalt nod din stânga celui căutat.

$$\begin{aligned} E[I_J] &= 1P(I_j = 1) + 0P(I_j = 0) \\ \Leftrightarrow E[I_J] &= P(I_j = 1) \\ \Leftrightarrow E[I_j] &= P(H_j \geq H_{j+1}, ..., H_j \geq H_{m-1}) \\ \Leftrightarrow E[I_j] &= \sum_{k\geq 1} P(H_j = k)P(H_{j+1} \leq k)...P(H_{m-1} \leq k) \\ \Leftrightarrow E[I_j] &= \sum_{k\geq 1} p^{(k-1)}(1-p)(1-p^k)^{m-1-j-1+1} \\ \Leftrightarrow E[I_j] &= \frac{(1-p)}{p} \sum_{k\geq 1} p^k(1-p^k)^{m-j-1} \end{aligned}$$

Înlocuind ultimul rezultat in (2.5) se obține

$$E[L_{m-1}] = \begin{cases} 1 + \frac{(1-p)}{p} \sum_{j=1}^{m-2} \sum_{k\geq 1} p^k(1-p^k)^j & \text{dacă } m \neq 1 \\ 0 & \text{dacă } m = 1 \end{cases}$$

□

Acest ultim rezultat se poate rescrie sub o formă care v-a fi utilă într-o viitoare demonstrație. Pentru l $\in \mathbb{N}^*$, $E[L_l]$ devine:

$$E[L_l] = 1 + \frac{(1-p)}{p} \sum_{k\geq 1} 1 - p^k - (1-p^k)^l \tag{2.6}$$

Demonstraţie.

$$E[L_l] = 1 + \frac{(1-p)}{p} \sum_{j=1}^{l-1} \sum_{k\geq 1} p^k(1-p^k)^j$$

$$= 1 + \frac{(1-p)}{p} \sum_{k\geq 1} p^k \sum_{j=1}^{l-1} (1-p^k)^j \qquad \text{(Serie geometrică)}$$
$$= 1 + \frac{(1-p)}{p} \sum_{k\geq 1} p^k (-p^{-k}((1-p^k)^l + p^k - 1))$$
$$= 1 + \frac{(1-p)}{p} \sum_{k\geq 1} 1 - p^k - (1-p^k)^l$$

□

Definim următoarele variabile aleatoare, ce vor fi de folos pentru demonstrațiile ce vor urma. Prin convenție, $W_p(0) = W_p(1) = V_p(0) = V_p(1) = 0$

$$W_p(l) = \sum_{k=2}^{l} \binom{l}{k} (-1)^k \frac{p^{k-1}}{1-p^{k-1}} \qquad V_p(l) = \frac{1}{l} \sum_{k=2}^{l} \binom{l}{k} (-1)^k \frac{kp^{k-1}}{1-p^{k-1}}$$

Relația dintre cele doua variabile aleatoare definite anterior este următoarea:

$$W_p(l) - W_p(l-1) = V_p(l)$$

Demonstraţie.

$$W_p(l) - W_p(l-1) =$$
$$= \sum_{k=2}^{l} \binom{l}{k} (-1)^k \frac{p^{k-1}}{1-p^{k-1}} - \sum_{k=2}^{l} \binom{l-1}{k} (-1)^k \frac{p^{k-1}}{1-p^{k-1}}$$
$$= \sum_{k=2}^{l} (-1)^k \frac{p^{k-1}}{1-p^{k-1}} \left[\frac{l!}{k!(l-k)!} - \frac{(l-1)!}{k!(l-1-k)!} \right]$$
$$= \sum_{k=2}^{l} (-1)^k \frac{p^{k-1}}{1-p^{k-1}} \frac{l! - (l-1)!(l-k)}{k!(l-k)!}$$
$$= \sum_{k=2}^{l} (-1)^k \frac{p^{k-1}}{1-p^{k-1}} \frac{(l-1)!}{(k-1)!(l-k)!}$$
$$= \sum_{k=2}^{l} (-1)^k \frac{p^{k-1}}{1-p^{k-1}} \frac{k\ l!}{l\ k!(l-k)!}$$

$$= \frac{1}{l} \sum_{k=2}^{l} \binom{l}{k} (-1)^k \frac{k p^{k-1}}{1 - p^{k-1}} = V_p(l)$$

□

Având în vedere variabilele adăugate, avem următoarea propoziție.

Propoziţia 2.3.4. *Valoarea medie a înălțimii unui skip list cu n elemente și probabilitate p este*

$$E(T_n) = 1 + V_p(n+1)$$

Demonstraţie.

$$\begin{aligned}
E(T_n) &= 1 + \sum_{k=2}^{n+1} \binom{n+1}{k} \frac{k}{n+1} (-1)^k \frac{p^{k-1}}{1 - p^{k-1}} \\
&= 1 + \sum_{k=2}^{n+1} \binom{n}{k-1} (-1)^k \frac{p^{k-1}}{1 - p^{k-1}} \\
&= \sum_{k=2}^{n+1} \binom{n}{k-1} (-1)^k \frac{1}{1 - p^{k-1}} \\
&= \sum_{k=2}^{n+1} \binom{n}{k-1} (-1)^k \sum_{m=1}^{\infty} p^{(m-1)(k-1)} && \text{(Serie geometrică)} \\
&= \sum_{k=2}^{n+1} \sum_{m=1}^{\infty} \binom{n}{k-1} (-1)^k p^{(m-1)(k-1)} \\
&= \sum_{m=1}^{\infty} \sum_{k=1}^{n+1} \binom{n}{k} (-1)^{k+1} p^{(m-1)k} && \text{(Shimbăm sumarea la k=1)} \\
&= \sum_{m=1}^{\infty} \left[1 + \sum_{k=0}^{n+1} \binom{n}{k} (-1)^{k+1} p^{(m-1)k} \right] && \text{(Schimbam sumarea la k=0)} \\
&= \sum_{m=1}^{\infty} \left[1 - \sum_{k=0}^{n+1} \binom{n}{k} (-1)^k p^{(m-1)k} 1^{(n-k)} \right] \\
&= \sum_{m=1}^{\infty} \left[1 - (1 - p^{(m-1)})^n \right]
\end{aligned}$$

□

Propoziţia 2.3.5. *Relația dintre înălțimea medie și numărul mediu de mutări orizontale ale unui skip list cu l elemente este*

$$E[T_l] - \frac{p}{q}(E[L_l] - 1) = \frac{1}{1-p}$$

Demonstraţie. Considerând forma lui E$[L_l]$ de la (2.6), avem că

$$\begin{aligned} E[T_l] - \frac{p}{q}(E[L_l] - 1) = & \\ &= \sum_{k=1}^{\infty} \left[1 - (1 - p^{(k-1)})^l\right] - \sum_{k=1}^{\infty} \left[1 - p^k - (1 - p^k)^l\right] \\ &= \sum_{k=1}^{\infty} \left[1 - (1 - p^{(k-1)})^l - 1 + (1 - p^k)^l\right] + \sum_{k=1}^{\infty} p^k \\ &= \sum_{k\geq 1} ((1 - p^k)^l - (1 - p^{(k-1)})^l) + \sum_{k\geq 1} p^k \\ &= 1 + \frac{p}{1-p} \qquad (*) \\ &= \frac{1}{1-p} \end{aligned}$$

(*)Prima sumă este telescopică, aceasta având ca rezultat 1, iar a doua, fiind geometrică, este egală cu $\frac{p}{1-p}$. □

O consecință a ultimei propoziții este aceea că putem exprima $E[L_{m-1}]$ în funcție de $V_p(m)$, înlocuind $E[T_l]$ conform cu Propoziția2.3.4, obținând:

$$E[L_{m-1}] = \frac{1-p}{p} V_p(m) \tag{2.7}$$

Teorema 2.3.6. *Într-un skip list cu n elemente și probabilitate p, costul mediu pentru o căutare a cheii de pe poziția m, în cazul în care aceasta există, sau costul căutării poziției în care aceasta trebuie inserat în cazul în care nu se află în listă, este*

$$E[C_n^{(m)}] = V_p(n+1) + \frac{1-p}{p} V_p(m) + 1$$

Demonstraţie. Relația se obține imediat, înlocuind în (2.4) înălțimea medie obținută la Propoziția2.3.4, și numărul mediu de mutări orizontale din ecuația (2.7). □

Teorema 2.3.7. *Într-un skip list cu n elemente și probabilitate p, costul mediu al căutărilor, în cazul în care nodurile există, și costul căutării poziției în care aceastea trebuie inserate, în cazul în care nu se află în listă, sunt:*

$$E[S_n] = V_p(n+1) + \frac{1-p}{p}\frac{W_p(n)}{n} + 1$$
$$E[U_n] = V_p(n+1) + \frac{1-p}{p}\frac{W_p(n+1)}{n+1} + 1$$

Demonstraţie.

$$S_n = \frac{1}{n}\sum_{m=1}^{n} C_n^{(m)} \qquad \text{(din definiție)}$$
$$\Leftrightarrow E[S_n] = \frac{1}{n}\sum_{m=1}^{n} E[C_n^{(m)}] \qquad \text{(proprietățile mediei)}$$
$$\Leftrightarrow E[S_n] = \frac{1}{n}\sum_{m=1}^{n}\left[V_p(n+1) + \frac{1-p}{p}V_p(m) + 1\right] \qquad \text{(Teorema 2.3.6)}$$
$$\Leftrightarrow E[S_n] = V_p(n+1) + \frac{1-p}{p}\frac{1}{n}\sum_{m=1}^{n}[V_p(m)] + 1$$
$$\Leftrightarrow E[S_n] = V_p(n+1) + \frac{1-p}{p}\frac{1}{n}\sum_{m=1}^{n}[W_p(m) - W_p(m+1)] + 1 \qquad \text{(Relația dinte } W_p \text{ si } V_p\text{)}$$
$$\Leftrightarrow E[S_n] = V_p(n+1) + \frac{1-p}{p}\frac{W_p(n)}{n} + 1 \qquad \text{(Sumă telescopică)}$$

Pentru E[U_n] demonstrația se face în mod analog. □

În continuare, Papadakis, a calculat valorile asimptotice ale costurilor de căutare, pornind cu expresiile asimptotice ale lui $V_p(m)$ și $W_p(m)$, astfel:

$$W_p(m) = m\, log_{\frac{1}{p}}(m) + m\left[\frac{1-\gamma}{ln(p)} - \frac{1}{2} + f_{-1,\frac{1}{p}}(m)\right] + O(1) \quad \text{cu m } \to \infty$$
$$V_p(m) = log_{\frac{1}{p}}(m) - \frac{\gamma}{ln(p)} - \frac{1}{2} - f_{0,\frac{1}{p}}(m-1) + O\left(\frac{1}{m}\right) \quad \text{cu m } \to \infty$$

unde γ (=0,5772) este constanta lui Euler, $f_{-1,\frac{1}{p}}(m)$ și $f_{0,\frac{1}{p}}(m-1)$ sunt funcții oscilatoare definite de Papadakis în [Pap93], în felul următor

$$f_{r,x}(l) = \frac{2}{ln(x)} \sum_{k=1}^{\infty} \Re \left(\Gamma \left(r - i\frac{2k\pi}{ln(x)} \right) e^{i2k\pi \, log_x(l)} \right) \tag{2.8}$$

unde i $=\sqrt{-1}$, r $\in \mathbb{Z}$, x $\in \mathbb{R}$, l $\in (0, \infty)$ și $\Re(a)$ reprezintă partea reală a numărului a.

În funcție de acestea, Papadakis a exprimat costul mediu al unei căutări astfel

$$E(C_n^{(m)}) = log_{\frac{1}{p}} n + \frac{1-p}{p} log_{\frac{1}{p}} m - \frac{1}{p}\left(\frac{\gamma}{ln\ p} + \frac{1}{2}\right) + 1 - f_{0,\frac{1}{p}}(n) - \frac{1-p}{p} f_{0,\frac{1}{p}}(m) + O\left(\frac{1}{m}\right)$$

cu m,n $\to \infty$

În continuare, costul mediu al căutărilor într-un skip list devine

$$E[S_n] = E[U_n] =$$
$$= \frac{1}{p} log_{\frac{1}{p}} n - \frac{1}{p}\left(\frac{\gamma}{ln\ p} + \frac{1}{2}\right) + \frac{1-p}{p\ ln\ p} + 1 + \frac{1-p}{p} f_{-1,\frac{1}{p}}(n) - f_{0,\frac{1}{p}}(n) + O\left(\frac{1}{n}\right)$$

cu n $\to \infty$

unde $f_{-1,\frac{1}{p}}(n)$ și $f_{0,\frac{1}{p}}(n)$ sunt cele defniite la (2.8).

Astfel, stabilindu-se de către W.Pugh un majorant al costului căutării, iar valoarea exactă fiind dată de T. Papadakis, avem complexitatea timp a unui skip list O(log n).

Capitolul 3

Skip List pentru date multidimensionale

În acest capitol se va trata cazul în care datele reținute au mai multe câmpuri ce trebuie ordonate. Până în acest moment am prezentat structurile ca fiind ordonate după o valoare cheie, care reprezenta fiecare nod.

Considerând un exemplu practic, putem examina următoarea situație. În cadrul unei întreprinderi, apar multe momente în care angajații trebuie ordonați în funcție de salariu, și la fel de multe ocazii în care am vrea să fie aranjați după anul angajării. Astfel, avem nevoie de aceeași listă, ordonată în două moduri. Situații de acest tip pot sa apară într-o companie aeriană, fiind nevoie ca zborurile să fie sortate după ora plecării și ora sosirii, sau, într-o societate comercială pentru sortarea produselor în funcție de anul apariției, preț, valabilitate, sau în cadrul agențiilor de turism pentru a reține locațiile în funcție de preț, distanța față de punctele de interes.

În întâmpinarea unei situații de această natură, cu un simplu skip list, ar trebui să rearanjăm structura de fiecare dată, deși este cunoscut faptul că se va ajunge în acest punct. Pentru astfel de situații, se poate folosi un skip list pentru date multidimensionale.

3.1 Prezentarea Structurii

La baza acestui capitol stă descrierea lui Nickerson [Nic94] în ceea ce privește prima sa variantă a unui skip list multidimensional, cu mici modificări.

În primul rând, acesta consideră skip list determinist, introdus de J.I. Muro și Papadakis în [MPS92], pe care, în cele ce vor urma, l-am înlocuit cu un skip list clasic. Un skip list determinist diferă de cel clasic prin faptul că nodurile nu sunt adăugate aleatoriu, ci după reguli aferente arborilor binari de căutare. Spre exemplu, un skip list determinist 1-2, este cel de la care Pugh a pornit crearea celui clasic, proces care a fost descris incipient capitolului 2. Astfel, un skip list determinist are nevoie să fie reechilibrat, motiv pentru care am făcut respectiva înlocuire.

A doua modificare adusă constă în eliminarea straturilor de pointeri suplimentari fiecărei liste, deoarece, avantajul adus de acestea (abilitatea de a lega noduri cu componente identice pe o anumită dimensiune) nu justifică memoria suplimentară adusă de dn pointeri suplimentari (d-dimensiunea, n-numărul de noduri). În acest mod, fiecare nod, din fiecare listă, va avea toate informațiile, valoarea cheie variind în funcție de dimensiunea pe care se află respectivul skip list.

Așadar, în acest capitol se va prezenta o variantă de skip list multidimensional obținută prin legarea mai multor structuri skip list.

Pentru următorul exemplu se vor folosi date bidimensionale. Presupunem contextul unei structuri ce memorează salariul și anul recrutării a cinci angajați.

Structura aferentă acestor date se constituie din doua structuri skip list, ce sunt unite la bază de un nod ce stabilește dimensiunea pe care se va desfășura o acțiune, spre exemplu, dacă se vrea căutarea angajatului ce are un salariu de 3000, se va alege primul pointer al nodului de direcție, și se va efectua căutarea în primul skip list respectiv.

Angajat	Salariu	Anul angajării
A	1800	2018
B	2000	2019
C	3000	2016
D	7000	2014
E	3500	2015

Figura 3.1: Date bidimensionale

3.2 Algoritmi

Presupunem un skip list cu d dimensiuni și n elemente inserate cu probabilitate p. Numărul de noduri al unei astfel de structuri, va fi suma numărului de noduri din fiecare skip list, și, în plus, nodul cu cei d pointeri pentru direcție.

Numărul nodurilor unui skip list reprezintă suma nodurilor de pe fiecare nivel al structurii. Pe un singur nivel m al structurii, sunt, așa cum am precizat în Capitolul 2.3, $np^{(m-1)}$ noduri, așadar, într-un skip list avem $\sum_{m\geq 1} = \frac{n}{1-p}$ noduri.

Astfel, ajungem la concluzia conform căreia, numărul de noduri dintr-un skip list multidimensional este $\frac{nd}{1-p}$.

Costul procesului de creare a unei asemenea structuri, este O(dn log n), deoarece,

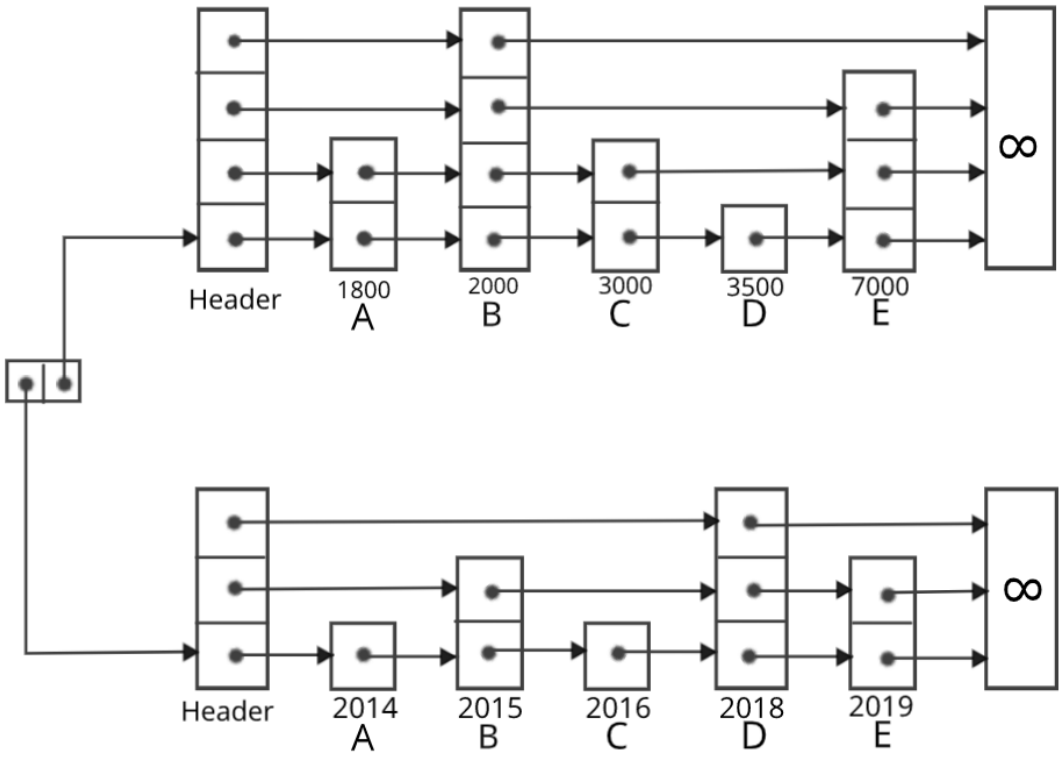

Figura 3.2: SkipList pentru date bidimensionale

pentru fiecare element n din fiecare cele d dimensiuni, costul inserării este, așa cum s-a notat capitolul anterior, O(log n).

Printr-un raționament similar cu cel anterior, deducem faptul că timpul pentru efectuarea operațiilor de căutare, ștergere, și inserare este proporțional cu cel al unui skip list clasic.

Așadar, deoarece pentru efectuarea acțiunilor de ștergere sau inserare este necesară parcurgerea a d skip listuri, apoi în fiecare, executarea operației cerute, se ajunge la un cost echivalet cu O(d log n).

Capitolul 4

Concluzii

În societatea actuală se dorește automatizarea cât mai multor tipuri de sarcini, în special a celor repetitive, care necesită memorarea datelor. În raport cu alte structuri de date, skip list este atât mai eficient, cât și mai simplu, la baza acestui rezultat stând conceptele matematice folosite și aplicate.

Timpul de execuție este foarte important deoarece, o tendință generala a oamenilor este cuantificarea timpului în bani, drept urmare aceștia sunt foarte preocupați de diminuarea timpului alocat diverselor activități, profitând de progresul tehnologic pentru a le atribui pe acestea unui program care poate face respectiva activitate în timp scurt cu costuri minime . O aplicație bună, dar slabă din punct de vedere al timpului de execuție nu va fi utilizată. Din aceste motive, este ideală utilizarea unor structuri rapide și ușor de manipulat.

S-a demonstrat complexitatea logaritmică a skp list, concluzie ce atestă faptul că acestea se pot integra în programe de memorare a datelor. Aceste motive conduc către utilizarea din ce în ce mai frecventă a acestor structuri.

Dan Wang și Jiangchuan Liu, prezintă în [WL06] o utilizare a acestor structuri în domeniul tehnic, oferind acces aleator conținutului video, transmis de la utilizator la utilizator, fără sa fie nevoie de descărcarea materialului video. În acest fel se creează o

rețea în care datele se transmit între utilizatori, fără sa fie nevoie de un server central.

Multe dintre procesele ce se doresc a fi automatizate și necesită memorarea datelor, sunt folosite pentru elemente complexe, ce au mai multe câmpuri de interes, în funcție de care se dorește manipularea datelor. Skip list poate fi modificat astfel încât să suporte și acest tip de date.

Așadar, skip list, prin eficiență și simplitate se califică utilizării într-o gamă foarte mare de programe. Fiind un subiect recent apărut, sunt șanse mari să se dovedească a fi utile în mai multe domenii, în diverse forme ce deja sunt sau urmează sa apară. Datorită rezultatelor de acest tip, matematica devine din ce în ce mai folosită în variate domenii, stând la baza progreselor, economisind timp, resurse si energie.

Bibliografie

[GKP94] Ronald L. Graham, Donald E. Knuth **and** Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. 2nd. USA: Addison-Wesley Longman Publishing Co., Inc., 1994, **pages** 113–114. ISBN: 0201558025.

[MPS92] J. Ian Munro, Thomas Papadakis **and** Robert Sedgewick. "Deterministic Skip Lists". **in**: *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '92. Orlando, Florida, USA: Society for Industrial **and** Applied Mathematics, 1992, **pages** 367–375. ISBN: 089791466X.

[Nic94] Bradford Nickerson. "Skip List Data Structures for Multidimensional Data". **in**: (April 1994), **pages** 4–10.

[Pap93] Thomas Papadakis. "Skip Lists and Probabilistic Analysis of Algorithms". phdthesis. CAN, 1993, **pages** 14, 37–42.

[Pug90a] William Pugh. *A Skip List Cookbook*. techreport. USA, **june** 1990.

[Pug90b] William Pugh. "Skip Lists: A Probabilistic Alternative to Balanced Trees". **in**: *Commun. ACM* 33.6 (**june** 1990), **pages** 668–676. ISSN: 0001-0782. DOI: 10.1145/78973.78977.

[WL06] Dan Wang **and** Jiangchuan Liu. "Peer-to-Peer Asynchronous Video Streaming using Skip List". **in**: **july** 2006, **pages** 1397–1400. DOI: 10.1109/ICME.2006.262800.