Tema Nr. 5: Căutarea în tabele de dispersie
Adresare deschisa, verificare pătratică

Timp alocat: 2 ore

Implementare

Se cere implementarea **corectă** și **eficientă** a operațiilor de *inserare* și *căutare* într-o tabela de dispersie ce folosește adresarea deschisa cu verificare pătratică.

Informații utile și pseudo-cod găsiți în notițele de curs sau în carte (Cormen), in secțiunea *11.4 Open addressing*.

Noțiunile închis/deschis (closed/open) specifica dacă se obliga folosirea unei poziții sau structuri de date.

Hashing (se referă la tabela de dispersie (hash table))

- Open Hashing
    - Pe o anumită poziție se pot stoca mai multe elemente ex: chaining
- Closed Hashing
    - Se poate stoca doar un singur element pe o anumita pozitie ex. linear/quadratic probing

Addressing (se referă la poziția finală a unui element fata de poziția inițială)

- Open Addressing (Adresare Deschisa)
    - Adresa finală (poziția finală) nu este complet determinat de către codul hash. Poziția depinde și de elementele care sunt deja în tabela. ex: linear/quadratic probing (verificare liniara/patratica)
- Closed Addressing (Adresare Închisă)
    - Adresa finală este întotdeauna determinată the codul hash (poziția inițială calculata) și nu exista probing (verificare) ex: chaining

Pentru această temă tabela de dispersie va avea o structură similară cu cea de mai jos, unde adresa finală va fi calculată aplicând funcția de hashing pe campul ***id*** din

structură. Câmpul ***name*** va fi folosit doar pentru demonstrarea corectitudinii operațiilor de căutare și ștergere.

typedef struct { int id; char name[30]; } Entry;

Praguri notare

Nota	Cerințe
5	Implementarea operației de inserare într-o tabela de dispersie; demo pe factor de umplere 95%
7	Implementarea operației de căutare în tabela de dispersie, cu demo
9	Implementarea corecta și completa al algoritmului, cu demo
10	Implementare operației de ștergere într-o tabela de dispersie. Demo și evaluare operație căutare după ștergerea unor elemente.

Evaluare

! Înainte de a începe să lucrați pe partea de evaluare, asigurați-vă că aveți un **algoritm corect**! Corectitudinea algoritmilor va trebui demonstrată pe date de intrare de dimensiuni mici.

Se cere evaluarea operației de *căutare* în tabele de dispersie cu adresare deschisa și verificare pătratică, în cazul **mediu statistic** (nu uitați să repetați măsurătorile de 5 ori). Pentru a obține evaluarea, trebuie sa:

1. Alegeți N, dimensiunea tabelei, un număr prim în jur de 10000 (e.g. 9973, sau 10007);

2. Pentru fiecare din următoarele valori pentru factorul de umplere $\alpha \in \{0.8, 0.85, 0.9, 0.95, 0.99\}$:

    a. Inserati în tabela n elemente aleator, astfel incat sa ajungeți la valoare lui α ($\alpha = n/N$)

    b. Căutați, în fiecare caz, m elemente aleator ($m \sim 3000$), astfel incat aproximativ jumătate din elemente sa fie *gasite*, iar restul sa *nu fie găsite* (în tabela). Asigurati-va ca elementele *găsite* sunt generate

uniform, i.e. să căutați elemente care au fost introduse la moment diferite, cu probabilitate egală (există mai multe moduri în care se poate asigura acest lucru)

c. Numărați operațiile efectuate de procedura de căutare (i.e. numărul de celule accesate)

d. Atenție la valorile pe care le căutați, sa fie într-o ordine aleatoare din cele introduse. Dacă sunt primele 1500 adaugate, implicit efortul mediu găsite va fi 1.

3. Generați un tabel de forma:

Factor de umplere	Efort mediu *gasite*	Efort maxim *gasite*	Efort mediu *ne-gasite*	Efort maxim *ne-gasite*
0.8				
0.85				
...	...		...	

Efort mediu = efort_total / nr_elemente

Efort maxim = număr maxim de accese efectuat de o operație de căutare

4. Interpretati rezultatele.

5. Pentru evaluarea operației de de căutare după ștergere, umpleți tabela de dispersie pana la factor de umplere 0.99. Stergeti elemente din tabela până ajungeți la factor umplere 0.8, după care căutați m elemente aleator (m ~ 3000), astfel incat aproximativ jumătate din elemente sa fie gasite, iar restul sa nu fie găsite (în tabela). Măsurați efortul necesar și adăugați în tabelul generat anterior.

6. Nu preluăm teme care nu sunt indentate și care nu sunt organizate în funcții (de exemplu nu preluam teme unde tot codul este pus în main).

7. ***Punctajele din barem se dau pentru rezolvarea corectă și completă a cerinței, calitatea interpretărilor din comentariul bloc și răspunsul corect dat de voi la întrebările puse de către profesor.***