Contents

Section outline

1. **Hashing**
   - Introduction to hashing
   - Hash functions
   - Deletion from a hash table

Introduction to hashing

- Insert, search, or delete from a table using address computation
- Given key is converted to an index for a position in the table

Introduction to hashing

- Insert, search, or delete from a table using address computation
- Given key is converted to an index for a position in the table
- Multiple keys may be mapped to the same index – collision
- Various collision resolution schemes
- Chaining
- Open addressing – linear probing, quadratic probing and double hashing

Introduction to hashing

- Insert, search, or delete from a table using address computation
- Given key is converted to an index for a position in the table
- Multiple keys may be mapped to the same index – collision
- Various collision resolution schemes
- Chaining
- Open addressing – linear probing, quadratic probing and double hashing

Example (Hashing keys in a table of size 8 using the division hash function)

- Table size: 8
- $H : K \to K \mod 8$
- $36 \to 36 \mod 8 = 4$
- $18 \to 18 \mod 8 = 2$
- $72 \to 72 \mod 8 = 0$
- $43 \to 43 \mod 8 = 3$
- $6 \to 6 \mod 8 = 6$

0:	72
1:	
2:	18
3:	43
4:	36
5:	
6:	6
7:	

Hash functions

Some hash functions – table size n

Division hash

- $H(x) = x \mod n$

Critique

Hash functions

Some hash functions – table size n

Division hash

- $H(x) = x \mod n$

Critique

Division hash

- If n is even, x and $H(x)$ have the same parity
- If $n = 2^k$, only the last k bits serve as the hash
- $H(x+1) \equiv H(x) + 1 \mod n$

Hash functions

Some hash functions – table size n

Division hash

- $H(x) = x \mod n$

Modular multiplicative hashing (MMH)

- $H(x) = [(ax + b) \mod p] \mod n$, p is a prime, $p > n$, $a, b \in [0..p-1]$, $a \neq 0$

Critique

Division hash

- If n is even, x and $H(x)$ have the same parity
- If $n = 2^k$, only the last k bits serve as the hash
- $H(x+1) \equiv H(x) + 1 \mod n$

Hash functions

Some hash functions – table size n

Division hash

- $H(x) = x \mod n$

Modular multiplicative hashing (MMH)

- $H(x) = [(ax + b) \mod p] \mod n$, p is a prime, $p > n$, $a, b \in [0..p-1]$, $a \neq 0$

Multiplication hash (FMH) – D E Knuth

- $H(x) = \lceil n(ax - \lfloor ax \rfloor) \rceil$, $a = \dfrac{\sqrt{5}-1}{2}$

Binary multiplicative hashing (BMH)

- $H(x) = \left\lfloor \frac{ax \mod 2^w}{2^{w-l}} \right\rfloor$

Critique

Division hash

- If n is even, x and $H(x)$ have the same parity
- If $n = 2^k$, only the last k bits serve as the hash
- $H(x+1) \equiv H(x) + 1 \mod n$

Hash functions

Some hash functions – table size n

Division hash

- $H(x) = x \mod n$

Modular multiplicative hashing (MMH)

- $H(x) = [(ax + b) \mod p] \mod n$, p is a prime, $p > n$, $a, b \in [0..p-1]$, $a \neq 0$

Multiplication hash (FMH) – D E Knuth

- $H(x) = \lceil n(ax - \lfloor ax \rfloor) \rceil$, $a = \dfrac{\sqrt{5}-1}{2}$

Binary multiplicative hashing (BMH)

- $H(x) = \left\lfloor \frac{ax \mod 2^w}{2^{w-l}} \right\rfloor$

Critique

Division hash

- If n is even, x and $H(x)$ have the same parity
- If $n = 2^k$, only the last k bits serve as the hash
- $H(x+1) \equiv H(x) + 1 \mod n$

Multiplicative hash

- Maximal use of operand bits
- FMH with $a = \phi$ achieves optimal spacing between consecutive keys
- BMH uses middle l-bits of ax and is easy to compute

Hash functions

Some hash functions – table size n

Division hash

- $H(x) = x \mod n$

Modular multiplicative hashing (MMH)

- $H(x) = [(ax + b) \mod p] \mod n$, p is a prime, $p > n$, $a, b \in [0..p-1]$, $a \neq 0$

Multiplication hash (FMH) – D E Knuth

- $H(x) = \lceil n(ax - \lfloor ax \rfloor) \rceil$, $a = \dfrac{\sqrt{5}-1}{2}$

Binary multiplicative hashing (BMH)

- $H(x) = \left\lfloor \frac{ax \mod 2^w}{2^{w-l}} \right\rfloor$

Hashing strings: $b = 128$ for 7-bit ASCII

- $c_0 c_1 \ldots c_m \to \sum_{\imath=0}^{m} c_\imath b^\imath \mod n$

Critique

Division hash

- If n is even, x and $H(x)$ have the same parity
- If $n = 2^k$, only the last k bits serve as the hash
- $H(x+1) \equiv H(x) + 1 \mod n$

Multiplicative hash

- Maximal use of operand bits
- FMH with $a = \phi$ achieves optimal spacing between consecutive keys
- BMH uses middle l-bits of ax and is easy to compute

Deletion from a hash table

- Probing requires stepping over table entries to find a vacant cell
- If a cell stepped over for entering a certain key is deleted, searching for that key will fail – unless it is filled with some other key
- Lazy deletion leaves behind an indication that a cell should be stepped over even if there is no key in it
- Avoids failure of searches

Section outline

Method and example

- When a collision happens, the new item is added to the existing items
- Items mapped to the same table entry may be maintained as a linked list
- For uniform distribution of m keys, expected number of items in collision in a table of size m is $\alpha = \frac{m}{n}$ – the load factor
- Time complexities remain $O(1)$ for low load factors

Example (Hashing keys in a table of size 8 using chaining)

- Table size: 8
- $H : K \to K \mod 8$
- $36 \to 36 \mod 8 = 4$
- $18 \to 18 \mod 8 = 2$
- $72 \to 72 \mod 8 = 0$
- $43 \to 43 \mod 8 = 3$
- $6 \to 6 \mod 8 = 6$
- $5 \to 5 \mod 8 = 5$
- $15 \to 15 \mod 8 = 7$

0:	72
1:	
2:	10, 18
3:	43
4:	36
5:	
6:	6
7:	

Analysis of hashing with chaining

- How many steps does it take to know that a key is absent?
- Any key, including the search key, is equally likely to be in any one of the n slots
- Length of the chain in any slot is $\frac{m}{n} = \alpha$
- For failure, after computing the hash function each of these α keys will have to be examined
- Total time needed for failure: $\Theta(1 + \alpha)$

Analysis of hashing with chaining

- How many steps does it take to know that a key is absent?
- Any key, including the search key, is equally likely to be in any one of the *n* slots
- Length of the chain in any slot is $\frac{m}{n} = \alpha$
- For failure, after computing the hash function each of these α keys will have to be examined
- Total time needed for failure: $\Theta(1 + \alpha)$
- The average time for searching in a chain is obtained as
- $\frac{\sum_{\imath=1}^{\alpha} \imath}{\alpha} = \frac{1+\alpha}{2}$
- Both are $O(1)$

Section outline

3 **Hashing with linear probing**

- Linear probing method
- Hashing with linear probing example

Linear probing method

1. Calculate $k = H(K)$, where H is the hash function and K is the key
2. If position k is empty or contains K, the search is complete
3. Otherwise, repeat earlier step setting
   $k = k + 1 \mod n$, where n is the table size
   until the starting position is revisited

- Simple generalisation is to probe using the key sequence $k + ai$
- Probe updation will then be: $k = k + a$

Hashing with linear probing example

Example (Hashing keys in a table of size 8)

0:	72
1:	
2:	18
3:	43
4:	36
5:	
6:	6
7:	

Add keys 10, 5 and 15 to the previous table

- Table size: 8
- $H : K \to K \mod 8$
- $10 \to 10 \mod 8 = 2 \xrightarrow{+3} 5$
- $5 \to 5 \mod 8 = 5 \xrightarrow{+2} 7$
- $15 \to 15 \mod 8 = 7 \xrightarrow{+2} 2$

0:	72
1:	15
2:	18
3:	43
4:	36
5:	10
6:	6
7:	5

Section outline

4 **Clustering**
- Primary and secondary clustering
- Clustering problem with linear problem

Primary and secondary clustering

Definition (Primary clustering)

The tendency for a collision resolution scheme to create runs of filled slots near the hash function position of keys

Definition (Secondary clustering)

The tendency for a collision resolution scheme to create runs of filled slots away the hash function position of keys

Clustering problem with linear problem

- Two key sequences originating from k_1 and k_2 may come together so that $H(k_1) + a\imath = H(k_2) + a\jmath$
- Thereafter, all subsequent probes turn out to be identical
- Collisions lead to clustering of keys
- Primary clustering results for $a = 1$
- Larger values of a lead to secondary clustering
- Increase in search time over the expected values where the distribution of keys is assumed to be truly random
- Problem is aggravated when the clusters are bridged with new keys

Section outline

5 **Double hashing**

- Method of double hashing
- Example of double hashing

Method of double hashing

- Double is another solution to the primary and secondary clustering problem
- Another hash function is used along with the primary hash function
- In the event of a collision, the probing sequence used is $z_\imath = H(K) + \imath H_2(K), \imath \geq 0$
- Required properties for the second hash function are:
  - it must never evaluate to 0
  - must ensure that all table entries can be probed
- An example of such a hash function is $H_2(K) = R - (K \mod R)$, R being a prime number smaller than the size of the hash table – R is chosen prime to minimise the pitfalls of division hashing

Example of double hashing

Example (Hashing keys in a table of size 10)

- Table size: 10
- $H : K \to K \mod 10$
- $H_2 : K \to 7 - (K \mod 7)$
- $89 \to 89 \mod 10 = 9$
- $18 \to 18 \mod 10 = 8$
- $49 \to 49 \mod 10 = 9; 7 - (49 \mod 7) = 7$
- $58 \to 58 \mod 10 = 8; 7 - (58 \mod 7) = 5;$
- $69 \to 69 \mod 10 = 9; 7 - (69 \mod 7) = 1$

0:	69
1:	
2:	
3:	58
4:	
5:	
6:	49
7:	
8:	18
9:	89

Section outline

6. **Hashing with quadratic probing**
   - Method and advantage over linear probing
   - Example of hashing with quadratic probing
   - Table coverage
   - Period of a quadratic sequence
   - Quadratic sequence with full table coverage
   - Example of an aperiodic quadratic sequence
   - Faster computation of quadratic probing sequence

Method and advantage over linear probing

- Calculate $k = H(K)$, as before
- Probing loop is similar to linear probing
- However, probing sequence is $k + a\imath + b\imath^2 \mod n$, where n is the table size

Method and advantage over linear probing

- Calculate $k = H(K)$, as before
- Probing loop is similar to linear probing
- However, probing sequence is $k + a\imath + b\imath^2 \mod n$, where n is the table size
- Primary clustering is avoided
- Elements that hash to the same address will always probe the same alternative cells, leading to secondary clustering
- Unlike in linear probing, once two sequences meet, they do not continue lock-step
- Table coverage is an issue

Example of hashing with quadratic probing

Example (Hashing keys in a table of size 10)

- Table size: 10
- $H : K \to K + i^2 \mod 10$
- $89 \to 89 \mod 10 = 9$
- $18 \to 18 \mod 10 = 8$
- $49 \to 49 \mod 10 = 9$; $(9 + 1^2) \mod 10 = 0$
- $58 \to 58 \mod 10 = 8$; $(8 + 1^2) \mod 10 = 9$; $(8 + 2^2) \mod 10 = 2$
- $69 \to 69 \mod 10 = 9$; $(9 + 2^2) \mod 10 = 3$

0:	49
1:	
2:	58
3:	69
4:	
5:	
6:	
7:	
8:	18
9:	89

Table coverage

- Consider the sequence: $z_\imath = z_0 + a\imath + b\imath^2 \mod n$, where n is the table size
- Does this sequence cover all the entries in the table?

Editor: Naive program for the quadratic sequence

```
main(int argc, char **argv){ // called with a b n
#define h(a,b,n,x) ((a + b*x)*x % n)
  int a = atoi(argv[1]);
  int b = atoi(argv[2]);
  int n = atoi(argv[3]);
  int i;

  printf("a=%d, b=%d, n=%d\n", a, b, n);
  for (i=0;i<n;i++) printf("%d->%d, ", i, h(a,b,n,i));
}
```

Some examples of quadratic sequences

Editor: Sample quadratic sequence

$ ./quadHashPT 3 7 10
a=3, b=7, n=10
0: 0, 1: 0, 2: 4, 3: 2, 4: 4, 5: 0, 6: 0, 7: 4, 8: 2, 9: 4

Editor: Sample quadratic sequence

$ ./quadHashPT 3 7 9
a=3, b=7, n=9
0: 0, 1: 1, 2: 7, 3: 0, 4: 7, 5: 1, 6: 0, 7: 4, 8: 4

Editor: Sample quadratic sequence

$ ./quadHashPT 3 7 8
a=3, b=7, n=8
0: 0, 1: 2, 2: 2, 3: 0, 4: 4, 5: 6, 6: 6, 7: 4

Termination condition becomes difficult/inefficient with such sequences

Period of a quadratic sequence

- Repetition is not too simple
- For $z_\imath = z_0 + a\imath + b\imath^2 \mod n$, consider whether there exists integers $\imath$ and $\jmath$ such that

  $z_\imath = z_\jmath$ and $0 \le \imath < \jmath < n$ [by subtraction]

$\therefore z_\imath = z_\jmath \equiv (\jmath - \imath)(a + b(\imath + \jmath)) \equiv 0 \mod n$

Period of a quadratic sequence

- Repetition is not too simple
- For $z_\imath = z_0 + a\imath + b\imath^2 \mod n$, consider whether there exists integers $\imath$ and $\jmath$ such that

  $z_\imath = z_\jmath$ and $0 \leq \imath < \jmath < n$ [by subtraction]

$\therefore$ $z_\imath = z_\jmath \equiv (\jmath - \imath)(a + b(\imath + \jmath)) \equiv 0 \mod n$

- When n is prime, its residues of $0, 1, \ldots, n-1$ form a field, where $a + bx \equiv 0 \mod n$ has a unique solution $b^{-1}(n-a) \mod n$ $(= w$ say) for any a and b ($b \not\equiv 0 \mod n$)
- For any $\imath$, repetition starts at $\jmath$ satisfying $\imath + \jmath \equiv w \mod n$ and $0 \leq \imath < \jmath < n$

Period of a quadratic sequence

- Repetition is not too simple
- For $z_\imath = z_0 + a\imath + b\imath^2 \mod n$, consider whether there exists integers $\imath$ and $\jmath$ such that

  $z_\imath = z_\jmath$ and $0 \leq \imath < \jmath < n$ [by subtraction]

$\therefore$ $z_\imath = z_\jmath \equiv (\jmath - \imath)(a + b(\imath + \jmath)) \equiv 0 \mod n$

- When n is prime, its residues of $0, 1, \ldots, n-1$ form a field, where $a + bx \equiv 0 \mod n$ has a unique solution $b^{-1}(n - a) \mod n$ ($= w$ say) for any a and b ($b \not\equiv 0 \mod n$)
- For any $\imath$, repetition starts at $\jmath$ satisfying $\imath + \jmath \equiv w \mod n$ and $0 \leq \imath < \jmath < n$

$\therefore$ At most $\lceil \frac{n}{2} \rceil$ table slots are examined until repetition sets in

- A free slot is not found even if available – can this be avoided?

Quadratic sequence with full table coverage

- We want to avoid $a + b(\imath + \jmath) \equiv 0 \mod n$ having a solution
- Need coefficients a and b for which $\imath$ and $\jmath$, $0 \leq \imath < \jmath < n$ don't exist to satisfy $a + b(\imath + \jmath) \equiv 0 \mod n$

Quadratic sequence with full table coverage

- We want to avoid $a + b(\imath + \jmath) \equiv 0 \mod n$ having a solution
- Need coefficients a and b for which $\imath$ and $\jmath$, $0 \leq \imath < \jmath < n$ don't exist to satisfy $a + b(\imath + \jmath) \equiv 0 \mod n$
- Let $n = \prod_{\imath \in \mathbb{Z}} p_\imath{}^{\alpha_\imath}$, $\alpha_\imath \geq 1$, $\exists \alpha_\imath > 1$ and each $p_\imath$ is prime,
- Let $B = \prod_{\imath \in \mathbb{Z}} p_\imath$ (st $p_\imath | n$), $A \in \mathbb{Z}$, $(A, B) = 1$ // $\gcd(A, B) = 1$
- Let $a = A$, $b = BC$, $C \in \mathbb{Z}$ and $m = a + b(\imath + \jmath)$

Quadratic sequence with full table coverage

- We want to avoid $a + b(\imath + \jmath) \equiv 0 \mod n$ having a solution
- Need coefficients a and b for which $\imath$ and $\jmath$, $0 \leq \imath < \jmath < n$ don't exist to satisfy $a + b(\imath + \jmath) \equiv 0 \mod n$
- Let $n = \prod_{\imath \in \mathbb{Z}} p_\imath^{\alpha_\imath}$, $\alpha_\imath \geq 1$, $\exists \alpha_\imath > 1$ and each $p_\imath$ is prime,
- Let $B = \prod_{\imath \in \mathbb{Z}} p_\imath$ (st $p_\imath | n$), $A \in \mathbb{Z}$, $(A, B) = 1$ // gcd$(A, B) = 1$
- Let $a = A$, $b = BC$, $C \in \mathbb{Z}$ and $m = a + b(\imath + \jmath)$
- Let $d = (m, n)$, $\therefore d = \prod_{\imath \in \mathbb{Z}} p_\imath^{\beta_\imath}$, $\beta_\imath \geq 0$ // for d to divide n $(d|n)$

Quadratic sequence with full table coverage

- We want to avoid $a + b(\imath + \jmath) \equiv 0 \mod n$ having a solution
- Need coefficients a and b for which $\imath$ and $\jmath$, $0 \leq \imath < \jmath < n$ don't exist to satisfy $a + b(\imath + \jmath) \equiv 0 \mod n$
- Let $n = \prod_{\imath \in \mathbb{Z}} p_\imath^{\alpha_\imath}$, $\alpha_\imath \geq 1$, $\exists \alpha_\imath > 1$ and each $p_\imath$ is prime,
- Let $B = \prod_{\imath \in \mathbb{Z}} p_\imath$ (st $p_\imath | n$), $A \in \mathbb{Z}$, $(A, B) = 1$ // gcd$(A, B) = 1$
- Let $a = A$, $b = BC$, $C \in \mathbb{Z}$ and $m = a + b(\imath + \jmath)$
- Let $d = (m, n)$, $\therefore d = \prod_{\imath \in \mathbb{Z}} p_\imath^{\beta_\imath}$, $\beta_\imath \geq 0$ // for d to divide n $(d|n)$
- If $d \neq 1$, $\exists \beta_\imath \geq 1 \Rightarrow p_\imath \mid d \wedge p_\imath \mid B$ // $p_\imath$ divides d and B

Quadratic sequence with full table coverage

- We want to avoid $a + b(\imath + \jmath) \equiv 0 \mod n$ having a solution
- Need coefficients a and b for which $\imath$ and $\jmath$, $0 \leq \imath < \jmath < n$ don't exist to satisfy $a + b(\imath + \jmath) \equiv 0 \mod n$
- Let $n = \prod_{\imath \in \mathbb{Z}} p_\imath^{\alpha_\imath}$, $\alpha_\imath \geq 1$, $\exists \alpha_\imath > 1$ and each $p_\imath$ is prime,
- Let $B = \prod_{\imath \in \mathbb{Z}} p_\imath$ (st $p_\imath | n$), $A \in \mathbb{Z}$, $(A, B) = 1$ // gcd$(A, B) = 1$
- Let $a = A$, $b = BC$, $C \in \mathbb{Z}$ and $m = a + b(\imath + \jmath)$
- Let $d = (m, n)$, $\therefore d = \prod_{\imath \in \mathbb{Z}} p_\imath^{\beta_\imath}$, $\beta_\imath \geq 0$ // for d to divide n $(d|n)$
- If $d \neq 1$, $\exists \beta_\imath \geq 1 \Rightarrow p_\imath \mid d \wedge p_\imath \mid B$ // $p_\imath$ divides d and B
- $d = (m, n) \Rightarrow d \mid m \Rightarrow p_\imath \mid m$, but does $p_\imath \mid A + BC(\imath + \jmath)$?

Quadratic sequence with full table coverage

- We want to avoid $a + b(\imath + \jmath) \equiv 0 \mod n$ having a solution
- Need coefficients a and b for which $\imath$ and $\jmath$, $0 \leq \imath < \jmath < n$ don't exist to satisfy $a + b(\imath + \jmath) \equiv 0 \mod n$
- Let $n = \prod_{\imath \in \mathbb{Z}} p_\imath{}^{\alpha_\imath}$, $\alpha_\imath \geq 1$, $\exists \alpha_\imath > 1$ and each $p_\imath$ is prime,
- Let $B = \prod_{\imath \in \mathbb{Z}} p_\imath$ (st $p_\imath | n$), $A \in \mathbb{Z}$, $(A, B) = 1$ // $\text{gcd}(A, B) = 1$
- Let $a = A$, $b = BC$, $C \in \mathbb{Z}$ and $m = a + b(\imath + \jmath)$
- Let $d = (m, n)$, $\therefore d = \prod_{\imath \in \mathbb{Z}} p_\imath{}^{\beta_\imath}$, $\beta_\imath \geq 0$ // for d to divide n $(d|n)$
- If $d \neq 1$, $\exists \beta_\imath \geq 1 \Rightarrow p_\imath \mid d \wedge p_\imath \mid B$ // $p_\imath$ divides d and B
- $d = (m, n) \Rightarrow d \mid m \Rightarrow p_\imath \mid m$, but does $p_\imath \mid A + BC(\imath + \jmath)$?
- No! as $(A, B) = 1$, $\therefore (m, n) = 1$

$\therefore$ $(\imath - \jmath)m \equiv 0 \mod n$ or $(\imath - \jmath)m = kn$, $0 \leq \imath < \jmath < n$ has no solution

$\therefore$ Full table coverage is ensured

Quadratic sequence with full table coverage

- We want to avoid $a + b(\imath + \jmath) \equiv 0 \mod n$ having a solution
- Need coefficients a and b for which $\imath$ and $\jmath$, $0 \leq \imath < \jmath < n$ don't exist to satisfy $a + b(\imath + \jmath) \equiv 0 \mod n$
- Let $n = \prod_{\imath \in \mathbb{Z}} {p_\imath}^{\alpha_\imath}$, $\alpha_\imath \geq 1$, $\exists \alpha_\imath > 1$ and each $p_\imath$ is prime,
- Let $B = \prod_{\imath \in \mathbb{Z}} p_\imath$ (st $p_\imath | n$), $A \in \mathbb{Z}$, $(A, B) = 1$ // $\gcd(A, B) = 1$
- Let $a = A$, $b = BC$, $C \in \mathbb{Z}$ and $m = a + b(\imath + \jmath)$
- Let $d = (m, n)$, $\therefore d = \prod_{\imath \in \mathbb{Z}} {p_\imath}^{\beta_\imath}$, $\beta_\imath \geq 0$ // for d to divide n $(d|n)$
- If $d \neq 1$, $\exists \beta_\imath \geq 1 \Rightarrow p_\imath \mid d \wedge p_\imath \mid B$ // $p_\imath$ divides d and B
- $d = (m, n) \Rightarrow d \mid m \Rightarrow p_\imath \mid m$, but does $p_\imath \mid A + BC(\imath + \jmath)$?
- No! as $(A, B) = 1$, $\therefore (m, n) = 1$

$\therefore$ $(\imath - \jmath)m \equiv 0 \mod n$ or $(\imath - \jmath)m = kn$, $0 \leq \imath < \jmath < n$ has no solution

$\therefore$ Full table coverage is ensured

- **Example:** $z_\imath = z_0 + (2a + 1)\imath + 2b\imath^2 \mod 2^k$ // NB $(2a + 1, 2) = 1$
- C may be chosen as some $H_2(K) \neq 0$ (encompassing the benefit of double hashing) to reduce secondary clustering

Example of an aperiodic quadratic sequence

Editor: Naive program for an aperiodic quadratic sequence

```
main(int argc, char **argv){
#define h(a,b,n,x) (((2*a+1) + 2*b*x)*x % n)
  int a = atoi(argv[1]);
  int b = atoi(argv[2]);
  int k = atoi(argv[3]);
  int i, n;
  for (n=1, i=0; i<k; i++) n*=2;

  printf("a=%d, b=%d, n=%d\n", a, b, n);
  for (i=0;i<d;i++) printf("%d->%d, ", i, h(a,b,d,i));
}
```

Editor: Generated aperiodic quadratic sequence

```
$ ./quadHashFT 3 7 3
a=3, b=7, n=8
0: 0, 1: 5, 2: 6, 3: 3, 4: 4, 5: 1, 6: 2, 7: 7
```

Faster computation of quadratic probing sequence

- Computation of the quadratic probing sequence $z_\imath = z_0 + a\imath + b\imath^2$ is simplified using the method of finite differences
- $z_0 = H(K)$
- $\delta z = z_{\imath+1} - z_\imath = 2b\imath + (a + b)$
  1. Let $R = a + b$ and $Q = 2b$; also $z_0 = k \leftarrow H(K)$, $\jmath \leftarrow R$
  2. If position k is empty or contains K, the search is complete
  3. Otherwise, repeat earlier step setting $k \leftarrow k + \jmath, \jmath \leftarrow \jmath + Q \mod n$, where n is the table size until the starting position is revisited (for full table coverage)
- Let $H(K) = K \mod n$; $n = 10, K = 49, a = 3, b = 5$; $Q = 2b = 10$
- $z_0 = k = 9$; $\jmath \leftarrow R = 8$
- $z_1 = k \leftarrow k + \jmath = 9 + 8 \equiv 7 \mod n$; $\jmath \leftarrow \jmath + Q = 8 + 10 \equiv 8 \mod n$
- $z_2 = k \leftarrow k + \jmath = 7 + 8 \equiv 5 \mod n$; $\jmath \leftarrow \jmath + Q = 8 + 10 \equiv 8 \mod n$
- $z_3 = k \leftarrow k + \jmath = 5 + 8 \equiv 3 \mod n$; $\jmath \leftarrow \jmath + Q = 8 + 10 \equiv 8 \mod n$

Faster computation of quadratic probing sequence

- Computation of the quadratic probing sequence $z_\imath = z_0 + a\imath + b\imath^2$ is simplified using the method of finite differences
- $z_0 = H(K)$
- $\delta z = z_{\imath+1} - z_\imath = 2b\imath + (a+b)$
  1. Let $R = a + b$ and $Q = 2b$; also $z_0 = k \leftarrow H(K)$, $\jmath \leftarrow R$
  2. If position k is empty or contains K, the search is complete
  3. Otherwise, repeat earlier step setting $k \leftarrow k + \jmath, \jmath \leftarrow \jmath + Q \mod n$, where n is the table size until the starting position is revisited (for full table coverage)
- Let $H(K) = K \mod n$; $n = 10, K = 49, a = 3, b = 5$; $Q = 2b = 10$
- $z_0 = k = 9$; $\jmath \leftarrow R = 8$
- $z_1 = k \leftarrow k + \jmath = 9 + 8 \equiv 7 \mod n$; $\jmath \leftarrow \jmath + Q = 8 + 10 \equiv 8 \mod n$
- $z_2 = k \leftarrow k + \jmath = 7 + 8 \equiv 5 \mod n$; $\jmath \leftarrow \jmath + Q = 8 + 10 \equiv 8 \mod n$
- $z_3 = k \leftarrow k + \jmath = 5 + 8 \equiv 3 \mod n$; $\jmath \leftarrow \jmath + Q = 8 + 10 \equiv 8 \mod n$
- Equivalent to linear probing! as $2b \equiv 0 \mod n$

Section outline

7 **Analysis of open addressing**

- Expected time for unsuccessful search
- Average time for successful search

Expected time for unsuccessful search

Table size is taken as n and number of entries is m

Uniformity Each hash value of the probing sequence $h_\imath(x)$, $\imath \geq 0$ is equally likely to be any integer in the set $\{0, 1, 2, \ldots, n-1\}$

- $\Pr[T[h_0(x)]]$ is occupied $= \frac{m}{n}$

Independence After the first probe, on failure, the remaining probe sequence $h_\imath(x)$, $\imath \geq 1$ is equally likely to be any integer in the set $\{0, 1, 2, \ldots, n-1\} \setminus \{h_0(x)\}$

- $\mathrm{E}[T(m,n)] = \begin{cases} 1 + \frac{m}{n}\mathrm{E}[T(m-1, n-1)], m > 0, & h_0(x) \text{ and beyond} \\ 1, m = 0, & \text{failure in empty table} \end{cases}$

Expected time for unsuccessful search

Table size is taken as n and number of entries is m

Uniformity Each hash value of the probing sequence $h_\imath(x)$, $\imath \geq 0$ is equally likely to be any integer in the set $\{0, 1, 2, \ldots, n-1\}$

- $\Pr[T[h_0(x)]]$ is occupied $= \frac{m}{n}$

Independence After the first probe, on failure, the remaining probe sequence $h_\imath(x)$, $\imath \geq 1$ is equally likely to be any integer in the set $\{0, 1, 2, \ldots, n-1\} \setminus \{h_0(x)\}$

- $\mathrm{E}[T(m,n)] = \begin{cases} 1 + \frac{m}{n}\mathrm{E}[T(m-1, n-1)], m > 0, & h_0(x) \text{ and beyond} \\ 1, m = 0, & \text{failure in empty table} \end{cases}$

A unit cost of probing the cell in question is always incurred – irrespective of whether there are elements in the table or not

The term $\mathrm{E}[T(m-1, n-1)]$ is added with the probability of $\frac{m}{n}$ when the cell probed is not empty

Expected time for unsuccessful search

Table size is taken as n and number of entries is m

Uniformity Each hash value of the probing sequence $h_\imath(x)$, $\imath \geq 0$ is equally likely to be any integer in the set $\{0, 1, 2, \ldots, n-1\}$

- $\Pr[T[h_0(x)]]$ is occupied $= \frac{m}{n}$

Independence After the first probe, on failure, the remaining probe sequence $h_\imath(x)$, $\imath \geq 1$ is equally likely to be any integer in the set $\{0, 1, 2, \ldots, n-1\} \setminus \{h_0(x)\}$

- $\mathrm{E}[T(m,n)] = \begin{cases} 1 + \frac{m}{n}\mathrm{E}[T(m-1, n-1)], m > 0, \ h_0(x) \text{ and beyond} \\ 1, m = 0, \ \text{failure in empty table} \end{cases}$
- Assuming $\mathrm{E}[T(m,n)] \leq \frac{n}{n-m}$, inductively $[T(m-1, n-1) \Rightarrow T(m,n)]$

$$\begin{aligned} \mathrm{E}[T(m,n)] &= 1 + \frac{m}{n}\mathrm{E}[T(m-1,n-1)] \leq 1 + \frac{m}{n}\frac{(n-1)}{(n-1)-(m-1)} \\ &\leq 1 + \frac{m}{n}\frac{n}{n-m} = \frac{n}{n-m} = \frac{1}{1-\frac{m}{n}} = \frac{1}{1-\alpha} \in O(1) \end{aligned}$$

- Expected time for an unsuccessful search is $O(1)$, unless hash table is almost full; same for insertion

Average time for successful search

- Average search time $\overline{T}_{m,n}$ for a successful search may be determined by averaging over all entries
- Successful search has the same probe sequence as when the element was inserted (an unsuccessful search)
- The search time for entry $\imath$ is the load factor at the time of inserting that element: $\dfrac{1}{1-\frac{\imath}{n}}$
- $\overline{T}_{m,n} = \dfrac{1}{m}\displaystyle\sum_{\imath=0}^{\imath=m-1}\dfrac{1}{1-\frac{\imath}{n}}$

$\therefore\ \overline{T}_{m,n} = \dfrac{1}{m}\displaystyle\sum_{\imath=0}^{\imath=m-1}\dfrac{n}{n-\imath}$

$\therefore\ \overline{T}_{m,n} = \dfrac{n}{m}\displaystyle\sum_{\imath=0}^{\imath=m-1}\dfrac{1}{n-\imath} = \dfrac{n}{m}\sum_{\imath=n-m+1}^{n}\dfrac{1}{\imath} = \dfrac{1}{\alpha}(H_n - H_{n-m})$

- $\dfrac{1}{\alpha}\displaystyle\sum_{\imath=n-m+1}^{n}\dfrac{1}{\imath} \leq \dfrac{1}{\alpha}\int_{n-m}^{n}\dfrac{dx}{x} = \dfrac{1}{\alpha}\ln\dfrac{n}{n-m} = \dfrac{1}{\alpha}\ln\dfrac{1}{1-\alpha}$

Section outline

8. **Handling filled-up tables**
   - Rehashing
   - Costing of rehashing

Rehashing

- When hash table has too many items, searches take longer and insertions may fail
- Load factor α of a hash table of size n with m elements is $\frac{m}{n}$
- Unless multiple items are present in the same table entry, as in chaining, $0 \leq \alpha \leq 1$
- New table of double size may be used to store older items and make place for newer items
- Older items cannot be just copied, but need to be hashed to the new table – rehashing
- Rehashing is often triggered once the load factor reaches 0.75
- Other triggers could be failure to insert or the table becoming full

Costing of rehashing

Simplified algorithm:

1. Let $n = 1$ be the initial table size
2. Keep inserting until total elements $m > n$
3. Double n, create table of size $2n$
4. Move elements to new table
5. Continue inserting as before

- Worst case cost of insert: $O(m)$
- Not frequent, average cost?
- $c_i = \begin{cases} i & \text{if } (\imath - 1) = 2^k \\ 1 & \text{otherwise} \end{cases}$

Costing of rehashing

Simplified algorithm:

1. Let $n = 1$ be the initial table size
2. Keep inserting until total elements $m > n$
3. Double n, create table of size $2n$
4. Move elements to new table
5. Continue inserting as before

- Worst case cost of insert: $O(m)$
- Not frequent, average cost?
- $c_i = \begin{cases} i & \text{if } (\imath - 1) = 2^k \\ 1 & \text{otherwise} \end{cases}$

Example (Cost of rehashing)

Op	n	$c_\imath$
I(1)	1	1

1

Costing of rehashing

Simplified algorithm:

1. Let $n = 1$ be the initial table size
2. Keep inserting until total elements $m > n$
3. Double n, create table of size $2n$
4. Move elements to new table
5. Continue inserting as before

- Worst case cost of insert: $O(m)$
- Not frequent, average cost?
- $c_i = \begin{cases} i & \text{if } (\imath - 1) = 2^k \\ 1 & \text{otherwise} \end{cases}$

Example (Cost of rehashing)

Op	n	$c_\imath$
I(1)	1	1
Ins(2)	2	$1 + 1$

1
2

Costing of rehashing

Simplified algorithm:

1. Let $n = 1$ be the initial table size
2. Keep inserting until total elements $m > n$
3. Double n, create table of size $2n$
4. Move elements to new table
5. Continue inserting as before

- Worst case cost of insert: $O(m)$
- Not frequent, average cost?
- $c_i = \begin{cases} i & \text{if } (\imath - 1) = 2^k \\ 1 & \text{otherwise} \end{cases}$

Example (Cost of rehashing)

Op	n	$c_\imath$
I(1)	1	1
Ins(2)	2	$1 + 1$
Ins(3)	4	$1 + 2$

1
2
3

Costing of rehashing

Simplified algorithm:

1. Let $n = 1$ be the initial table size
2. Keep inserting until total elements $m > n$
3. Double n, create table of size $2n$
4. Move elements to new table
5. Continue inserting as before

- Worst case cost of insert: $O(m)$
- Not frequent, average cost?
- $c_i = \begin{cases} i & \text{if } (\imath - 1) = 2^k \\ 1 & \text{otherwise} \end{cases}$

Example (Cost of rehashing)

Op	n	$c_\imath$
I(1)	1	1
Ins(2)	2	$1+1$
Ins(3)	4	$1+2$
Ins(4)	4	1

1
2
3
4

Costing of rehashing

Simplified algorithm:

1. Let $n = 1$ be the initial table size
2. Keep inserting until total elements $m > n$
3. Double n, create table of size $2n$
4. Move elements to new table
5. Continue inserting as before

- Worst case cost of insert: $O(m)$
- Not frequent, average cost?
- $c_i = \begin{cases} i & \text{if } (\imath - 1) = 2^k \\ 1 & \text{otherwise} \end{cases}$

Example (Cost of rehashing)

Op	n	$c_\imath$
I(1)	1	1
Ins(2)	2	$1 + 1$
Ins(3)	4	$1 + 2$
Ins(4)	4	1
Ins(5)	8	$1 + 4$

1
2
3
4
5

Costing of rehashing

Simplified algorithm:

1. Let $n = 1$ be the initial table size
2. Keep inserting until total elements $m > n$
3. Double n, create table of size $2n$
4. Move elements to new table
5. Continue inserting as before

- Worst case cost of insert: $O(m)$
- Not frequent, average cost?
- $c_i = \begin{cases} i & \text{if } (\imath - 1) = 2^k \\ 1 & \text{otherwise} \end{cases}$

Example (Cost of rehashing)

Op	n	$c_\imath$
I(1)	1	1
Ins(2)	2	$1+1$
Ins(3)	4	$1+2$
Ins(4)	4	1
Ins(5)	8	$1+4$
Ins(6)	8	1

1
2
3
4
5
6

Costing of rehashing

Simplified algorithm:

1. Let $n = 1$ be the initial table size
2. Keep inserting until total elements $m > n$
3. Double n, create table of size $2n$
4. Move elements to new table
5. Continue inserting as before

- Worst case cost of insert: $O(m)$
- Not frequent, average cost?
- $c_i = \begin{cases} i & \text{if } (i-1) = 2^k \\ 1 & \text{otherwise} \end{cases}$

Example (Cost of rehashing)

Op	n	c_i
I(1)	1	1
Ins(2)	2	$1+1$
Ins(3)	4	$1+2$
Ins(4)	4	1
Ins(5)	8	$1+4$
Ins(6)	8	1
Ins(7)	8	1

1
2
3
4
5
6
7

Costing of rehashing

Simplified algorithm:

1. Let $n = 1$ be the initial table size
2. Keep inserting until total elements $m > n$
3. Double n, create table of size $2n$
4. Move elements to new table
5. Continue inserting as before

- Worst case cost of insert: $O(m)$
- Not frequent, average cost?
- $c_i = \begin{cases} i & \text{if } (\imath - 1) = 2^k \\ 1 & \text{otherwise} \end{cases}$

Example (Cost of rehashing)

Op	n	$c_\imath$
I(1)	1	1
Ins(2)	2	$1 + 1$
Ins(3)	4	$1 + 2$
Ins(4)	4	1
Ins(5)	8	$1 + 4$
Ins(6)	8	1
Ins(7)	8	1
Ins(8)	8	1

1
2
3
4
5
6
7
8

Costing of rehashing

Simplified algorithm:

1. Let $n = 1$ be the initial table size
2. Keep inserting until total elements $m > n$
3. Double n, create table of size $2n$
4. Move elements to new table
5. Continue inserting as before

- Worst case cost of insert: $O(m)$
- Not frequent, average cost?
- $c_i = \begin{cases} i & \text{if } (i-1) = 2^k \\ 1 & \text{otherwise} \end{cases}$

Example (Cost of rehashing)

Op	n	c_i
I(1)	1	1
Ins(2)	2	1 + 1
Ins(3)	4	1 + 2
Ins(4)	4	1
Ins(5)	8	1 + 4
Ins(6)	8	1
Ins(7)	8	1
Ins(8)	8	1
Ins(9)	16	1 + 8

Costing of rehashing

Simplified algorithm:

1. Let $n = 1$ be the initial table size
2. Keep inserting until total elements $m > n$
3. Double n, create table of size $2n$
4. Move elements to new table
5. Continue inserting as before

- Worst case cost of insert: $O(m)$
- Not frequent, average cost?
- $c_i = \begin{cases} i & \text{if } (i-1) = 2^k \\ 1 & \text{otherwise} \end{cases}$

Example (Cost of rehashing)

Op	n	c_i
I(1)	1	1
Ins(2)	2	$1+1$
Ins(3)	4	$1+2$
Ins(4)	4	1
Ins(5)	8	$1+4$
Ins(6)	8	1
Ins(7)	8	1
Ins(8)	8	1
Ins(9)	16	$1+8$
	Total	24

1
2
3
4
5
6
7
8
9

Average cost: $\frac{24}{9} = \frac{8}{3}$

Cost of rehashing (contd.)

Aggregate analysis

- Cost of n insertions:
  $$\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\lg n} 2^j = n + (2n - 1) < 3n$$
- Average cost of insertion: $\frac{3n-1}{n} < n$
- The cost of insertion amortised over all the insert operations is asymptotically a fixed value: $O(1)$

Cost of rehashing (contd.)

Aggregate analysis

- Cost of n insertions:
  $$\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\lg n} 2^j = n + (2n - 1) < 3n$$
- Average cost of insertion: $\frac{3n-1}{n} < n$
- The cost of insertion amortised over all the insert operations is asymptotically a fixed value: $O(1)$

  – similar to inserting in a hash table of fixed size!

Cost of rehashing (contd.)

Aggregate analysis

- Cost of n insertions:
  $\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\lg n} 2^j =$
  $n + (2n - 1) < 3n$
- Average cost of insertion:
  $\frac{3n-1}{n} < n$
- The cost of insertion amortised over all the insert operations is asymptotically a fixed value: $O(1)$

  – similar to inserting in a hash table of fixed size!

Accounting analysis

- Charge 3 units for each insertion
  1. Use 1 unit for inserting this item
  2. Save 2 units for later use

Cost of rehashing (contd.)

Aggregate analysis

- Cost of n insertions:
  $\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\lg n} 2^j = n + (2n - 1) < 3n$
- Average cost of insertion: $\frac{3n-1}{n} < n$
- The cost of insertion amortised over all the insert operations is asymptotically a fixed value: $O(1)$

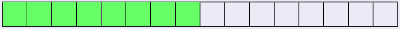
  – similar to inserting in a hash table of fixed size!

Accounting analysis

- Charge 3 units for each insertion
  1. Use 1 unit for inserting this item
  2. Save 2 units for later use
- When table is doubled from $2n$ to $4n$:
  1. n elements, each with a saving of 2 units were added since the previous doubling from n to $2n$

  2. Total saving of $2n$ units just enough to move the $2n$ elements, exhausing all savings

Cost of rehashing (contd.)

Aggregate analysis

- Cost of n insertions:
$$\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\lg n} 2^j = n + (2n - 1) < 3n$$
- Average cost of insertion: $\frac{3n-1}{n} < n$
- The cost of insertion amortised over all the insert operations is asymptotically a fixed value: $O(1)$

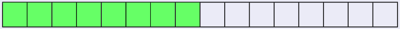
  – similar to inserting in a hash table of fixed size!

Accounting analysis

- Charge 3 units for each insertion
  1. Use 1 unit for inserting this item
  2. Save 2 units for later use
- When table is doubled from $2n$ to $4n$:
  1. n elements, each with a saving of 2 units were added since the previous doubling from n to $2n$

  2. Total saving of $2n$ units just enough to move the $2n$ elements, exhausing all savings

     
- All above operations are achieved by charging a fixed amount (3 units) per insert; thus, amortised cost is $O(1)$

Section outline

9. **Commutative rings and fields**

- Rings
- Commutative ring with identity
- Field

Rings

Definition (Ring)

A ring is a set R with two binary operations addition (denoted $+$) and multiplication (denoted $\cdot$). These operations satisfy the following axioms:

1. Addition is associative: If $a, b, c \in R$, then $a + (b + c) = (a + b) + c$
2. There is an identity for addition, denoted 0. It satisfies $0 + a = a \quad \text{and} \quad a + 0 = a \quad \text{for all} \quad a \in R$
3. Every element of R has an additive inverse; that is, if $a \in R$, there is an element $-a \in R$ which satisfies $a + (-a) = 0 \quad \text{and} \quad (-a) + a = 0$
4. Addition is commutative: If $a, b \in R$, then $a + b = b + a$
5. Multiplication is associative: If $a, b, c \in R$, then $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
6. Multiplication distributes over addition: If $a, b, c \in R$, then $a \cdot (b + c) = a \cdot b + a \cdot c$

Commutative ring with identity

Definition (Commutative ring)

A ring R is commutative if the multiplication is commutative: For all $a, b \in R$, $a \cdot b = b \cdot a$

Definition (Ring with identity)

A ring R is a ring with identity if there is an identity for multiplication. There is an element $1 \in R$ such that
$1 \cdot a = a$ and $a \cdot 1 = a$ for all $a \in R$

Definition (Commutative ring with identity)

A commutative ring which has an identity element is called a commutative ring with identity.

Field

Definition (Multiplicative inverse)

Let R be a ring with identity, and let $x \in R$. The multiplicative inverse of x is an element $x^{-1} \in R$ which satisifies $x \cdot x^{-1} = 1 \quad \text{and} \quad x^{-1} \cdot x = 1$

Definition (Field)

A field F is a commutative ring with identity in which $1 \neq 0$ and every non-zero element has a multiplicative inverse.

Examples

Example (Commutative rings)

- The integers $\mathbb{Z}$
- The rational numbers $\mathbb{Q}$
- The real numbers $\mathbb{R}$

Example (Integers modulo n)

- Let $n \geq 2$ be an integer, the integers modulo n is the set $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$ called the residues of n
- $\mathbb{Z}_2 = \{0, 1\}$ and $\mathbb{Z}_6 = \{0, 1, 2, 3, 4, 5\}$
- $\mathbb{Z}_n$ becomes a commutative ring with identity under the operations of addition modulo n and multiplication modulo n
- $\mathbb{Z}_n$ is a field if and only if n is prime