CSC2001F: Data Structures II

Omowunmi Isafiade

Email: omowunmi.isafiade@uct.ac.za

Office: Room 306

A Quick Reflection...

- The problem of overflow
- Efficient Hash Function Implementation
- Expected number of probes (analysis)
  - Naive analysis: $1/(1-\alpha)$
  - Better estimate: $(1+1/(1-\alpha)^2)/2$

On the Problem of Overflow & Hash Functions

- Recall: Methods of creating hash functions (compression map)
- AIMS: Circumvent the problem of overflow
- Overflow occurs in a hash table when an item cannot be inserted because the hash table is too small
- Having to deal with large sparsely occupied tables
  - Memory intensive

On the Problem of Overflow & Hash Functions

- BUT!!! We need to also take care when implementing the hash function in order to avoid expensive computations

- And... even though speed is important we want to distribute keys equitably to avoid collisions

- Two things!!!
  - Load Factor
  - Cost of Probing

Outline

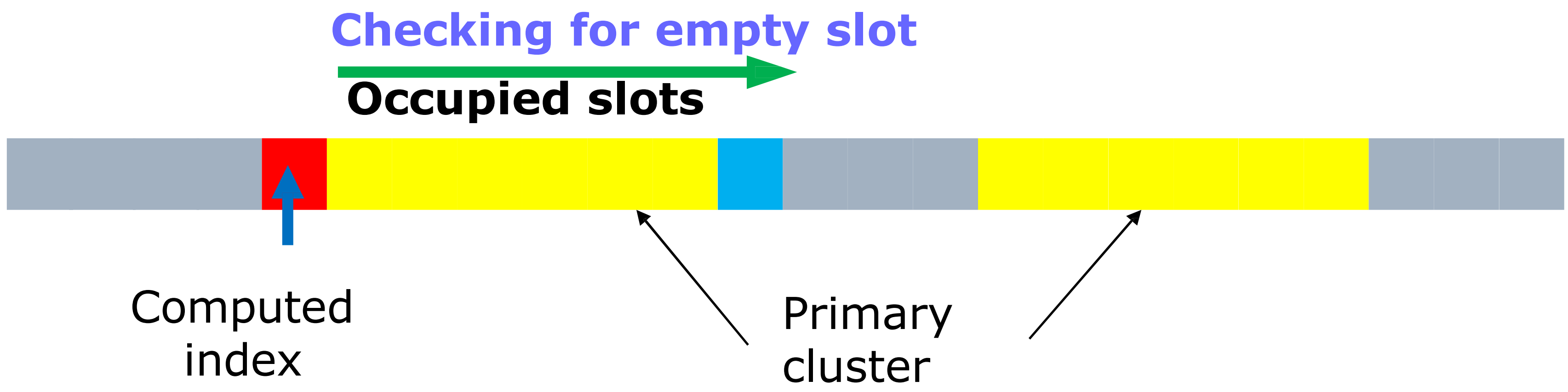
- Quadratic probing – A method of Collision Resolution
- Load Factor (re-visited)
- Rehashing and Overflow (The problem of Table Size)
- Secondary Clustering
- Summary

What is Quadratic Probing?

- Quadratic probing is another collision resolution method (**examine cells away from the original probe point** – ”in a particular sequence”
- **Recall #1**: Collisions are more likely to occur, with many-to-one hash functions.
  - So several keys collide at the same index
- **Recall #2:** Primary clustering (consequence of linear probing) makes collision resolution expensive

Quadratic Probing

- A **method of addressing primary clustering of linear probing**
  - Examine specific cells away from the original probe point
- Copes better with insertions in the presence of collisions in high load factor large tables



Quadratic Probing – How it works

- The concept of quadratic probing comes from the formula $F(i)=(i^2), i \in 1, 2, \cdots$
- Where i^2 represents the i^{th} position after the one at which the collision occurs
- If the hash function evaluates to H and a search in cell H leads to a collision, the resolution strategy is to examine cells... $H+(1^2), H+(2^2), \cdots, H+i^2$
- **Note:** Contrasts with linear probing where sequential cells $H+1, H+2, \cdots, H+i$ are examined

Quadratic Probing – Example

- Using quadratic probing, insert the set of keys {337, 123, 617, 93, 63} into a hash table
- Using the hash function $h(k) = \lfloor m(KA \bmod 1) \rfloor$
- K = key, A= 0.2, m = 10
- Hash function computation gives?

Quadratic Probing – Example

- Using quadratic probing, insert the set of keys {337, 123, 617, 93, 63} into a hash table
- Using the hash function $h(k) = \lfloor m(KA \bmod 1) \rfloor$
- K = key, A= 0.2, m = 10
- Hash function computation gives?

K	337	123	617	93	63
h(k)	4	6	4	6	6

K	337	123	617	93	63
h(k)	4	6	4	6	6

Comparison to linear probing..

Example- Solution (Quadratic Probing)

- Step 1: Insert 337
- Step 2: Insert 123
- **Step 3**: Insert 617 (**collision!**)
  - $H+(i^2)=4+(1^2)=5$ (so it goes to index 5)
- **Step 4:** Insert 93 (**collision!**)
  - $H+(i^2)=6+(1^2)=7$
- **Step 5**: Insert 63 (**collision!**)
  - $H+(i^2)=6+(1^2)=7$ (Occupied)
  - $H+(i^2)=6+(2^2)=10$ (Out of bounds...)

0	63
1	
2	
3	
4	337
5	617
6	123
7	93
8	
9	

Quadratic Probing- Table Size Issue

- Consider inserting 17, 10, 53, into the table...

- What happens?

0	63
1	
2	
3	
4	437
5	617
6	123
7	93
8	
9	

Quadratic Probing- Table Size Issue

- Consider inserting 17, 10, 53, into the table...
- What happens?
- **Load factor** ?

K	337	123	617	93	63 17	10	53
h(k)	4	6	4	6	6 4	20 (0)	6

0	63
1	10
2	x
3	
4	437
5	617
6	123
7	93
8	17
9	

Quadratic Probing – Table Size Issue

- With quadratic probing we have the problem of inserting 53
- Issue…
  - Choice of Table Size & Load Factor (>= 0.5)
- Solution:
  - If table size is prime then load factor is within range
- Advantage: we can always insert a new item and no cell is probed more than twice during an access.

Quadratic Probing- Prime Table Size

- Example: M = 11, instead of 10
- Step 1: Insert 337
- Step 2: Insert 123
- **Step 3**: Insert 617 (**collision!**)
  - $H+(i^2)=4+(1^2)=5$ (goes to index 5)
- **Step 4:** Insert 93 (**collision!**)
  - $H+(i^2)=6+(1^2)=7$
- **Step 5**: Insert 63 (**collision!**)
  - $H+(i^2)=6+(1^2)=7$ (Occupied)
  - $H+(i^2)=6+(2^2)=10$

K	337	123	617	93	63
h(k)	4	6	4	6	6

0	
1	
2	
3	
4	337
5	617
6	123
7	93
8	
9	
10	63

Quadratic Probing- Prime Table Size

- Example: M = 11, instead of 10
- **Step 6: Insert 17 (collision) → 8**
- **Step 7: Insert 10 (collision) → 0**

  - **Load Factor > 0.5 !!!**
- **Step 8: Insert 53 (collision) → ??**
  - $H+(i^2)=6+(1^2)=7$ (Occupied)
  - $H+(i^2)=6+(2^2)=10$ (Occupied)
  - $H+(i^2)=6+(3^2)=4$ (Occupied)

K	337	123	617	93	63	17	10	53
h(k)	4	6	4	6	6	4	20 (0)	6

0	10
1	
2	
3	
4	337
5	617
6	123
7	93
8	17
9	
10	63

Quadratic Probing – Table Size & Load Factor

- What happens if the table size is too small
- What happens if quadratic probing cannot resolve the collision?
- Possible Solutions:
  - Adjust the load factor of the hash table by expanding the table size (**Rehashing**)
  - Requires that the load factor satisfies a constraint (<= threshold value)
  - Set pre-conditions for the table size

Reflection: If we want to **insert an item** and **table is not full**, **does quadratic probing always guarantee an insertion**?

Load Factor- Revisited

- If load factor > 0.5, the insertion could fail
- Typically, use a prime number for table size and a load factor $\alpha < 0.5$, guarantee success for insertion
- Example: inserting 53 into the table using quadratic probing (issues)

0	10
1	
2	
3	
4	337
5	617
6	123
7	93
8	17
9	
10	63

Overflow & Rehashing

- "Overflow occurs in a hash table when an item cannot be inserted because the hash table is too small"

- "**Rehashing** is the concept of dynamically expanding a hash table once the load factor reaches a pre-defined threshold value."

- **Example:** a load factor > 0.5 triggers a rehash operation

Secondary Clustering

- **Note:** Secondary clustering is a consequence of quadratic probing
- Since items probe the same alternative cells during collision resolution
- How do we resolve this?

"Approach characteristics to quadratic probing whereby elements that hash out to the same position probe the same alternative cells"

0	10
1	
2	
3	
→ 4	337
5	617
→ 6	123
→ 7	93
8	17
9	
→ 10	63

Secondary cluster Formation due to probing of the Same alternative cells to resolve collision

Methods of Collision Resolution?

- **Recall # 1**: **Linear probing**
  - Probe alternative locations successively (H+1, H+2, H+3,....)
  - Primary clustering (**problem - expensive**)
- **Recall # 2**: **Quadratic probing**
  - Probe alternative locations away from original probe point H...(H+1, H+4, H+9...)
  - **Resolves primary clustering**
  - BUT!!! Results in secondary clustering

How do we resolve secondary clustering?