CSE 332: Data Structures and Parallelism

Section 5: Hashing & Sorting

1. Hash... Browns?

For the following scenarios, insert the following elements in this order: 7, 9, 48, 8, 37, 57. For each table, TableSize = 10, and you should use the primary hash function *h(k) = k*.

a) Linear Probing - Insertion

0	8
1	37
2	57
3	
4	
5	
6	
7	7
8	48
9	9

Linear Probing - Delete 37, 7, 57

0	8
1	X
2	X
3	
4	
5	
6	
7	X
8	48
9	9

b) Quadratic Probing

0	
1	37
2	8
3	
4	
5	
6	57
7	7
8	48
9	9

c) Separate chaining hash table - Use an unsorted linked list for each slot.

0	
1	
2	
3	
4	
5	
6	
7	$7 \rightarrow 37 \rightarrow 57$
8	$48 \rightarrow 8$
9	9

2. Double Double Toil and Trouble

a) Describe double hashing.

The first hash function determines the location where initially try to place the item. If there is a collision, then the second hash function is used to determine the probing step distance as 1*h2(key), 2*h2(key), 3*h2(key) etc. away from the original location.

b) List 2 cons of quadratic probing and describe how one of those is fixed by using double hashing.

In quadratic probing,

1. if the table is more than half full (load factor = 0.5) then you are not guaranteed to be able to find a location to place the item,
2. suffers from secondary clustering (items that initially hash to the same location resolve the collision identically).

Assuming a good second hash function is used, double hashing does not suffer from 1). Assuming a good second hash function is used, double hashing avoids secondary clustering because items that initially hash to the same location resolve the collision differently, which decreases the likelihood that two elements will hash to the same index after initial collision.

c) Compare open addressing and separate chaining.

Open Addressing

- Deals with collisions by moving element to a different index
- Can use less memory
- Linear probing: easiest, but has primary clustering, need to resize a lot
- Quadratic probing: secondary clustering, will find a spot if $\lambda < ½$
- Double hashing: low chance of clustering, but need another hash function

Separate Chaining

- Deals with collisions by putting all the elements in a “bucket” at that index
- Easier to implement
- More memory because needs “bucket” data structure
- Average runtime for insert/delete/find: $O(1 + \lambda)$
    - Best: $O(1)$, Worst: $O(n)$

3. Clash of the Hashes!

For each of the following questions, choose the correct collision resolution option!

a) You are implementing a hash table on hardware with low memory and low computational power. Thankfully, your hash function almost always spreads keys out evenly and the items you are hashing take up a small amount of memory (e.g. integers or shorts). Which collision resolution is best?

Linear Probing Quadratic Probing Double Hashing Separate Chaining

Linear Probing is best as it tightly packs data together and primary clustering isn't as big of an issue since our hash function almost always spreads keys out evenly.

- Double Hashing is worse since the device has low computational power. So we can't use another hash function
- Separate Chaining is worse since data types are small and memory is limited, meaning that the LinkedList memory overhead is large. We want to save memory so we shouldn't use separate chaining.
- Quadratic Probing is worse since it can leave holes when doing collision resolution which could cause less space to be used. Moreover, quadratic probing requires a load factor < 0.5 meaning more frequent resizes.

b) Now you are working on creating a hash table specifically for Strings. The issue is these strings are all extremely long! The Strings are so long that the process of hashing them (which includes iterating through every char) affects the runtime. Which collision resolution is best?

Linear Probing Quadratic Probing Double Hashing Separate Chaining

We want to rehash the Strings as infrequently as possible.

Therefore Separate Chaining is best since we can keep the load factor high (e.g. >1.0) to avoid resizing and rehashing and still have the hash table work.

- Linear Probing is worse since to find an open spot the load factor must be < 1.0. Separate chaining, however, can keep a load factor > 1.0. This means that we must resize and as a result rehash items much more frequently if we use Linear Probing, something we want to avoid.
- Double Hashing is worse since we have to rehash the string upon collision. We also have the same issue as linear probing where the load factor must be < 1.0
-
- Quadratic Probing is worse since we must keep the load factor < 0.5 to work. So for the same reason as linear probing, separate chaining is better.

c) You are designing a hash table with your CSE 332 TA. However, the initial hash function that they designed causes items to cluster and they are adamant about keeping it. Which collision resolution is best?

Linear Probing   Quadratic Probing   Double Hashing   Separate Chaining

Double Hashing is best here. Since the initial hash function causes items to cluster, we know that

- Linear Probing will result in primary clustering.
- Quadratic Probing will result in secondary clustering
- Separate Chaining will grow really long chains in the cluster regions.

Therefore, Double hashing which provides the most even distribution would be much better for collision resolution.