course_content

top

Hash Tables

Week 3

Time:
5 min

Hashing

- A technique used to identify a specific object from other similar objects e.g.
  - Each student at a university assigned a unique Student ID
  - ISBN (International Standard Book Number)
- Can you identify other real life examples of hashing?
- Is the hashing always unique?

Time:
5 min

Hash Functions

- A hash function converts a key to a specific index.
- This is a method of converting a long & complex key (e.g. string, tuple etc) to a single integer key.
- The data is actually stored in an array with the location of the value being the index corresponding to the hashed key.
- The value can now be accessed in O(1) time (hopefully!)

Time:
15 min

Hashes - Exercise

- Groups of 3 - discuss the following
- Do not use Google. If you know how hash functions work, be creative!
- There are 25 people in the class. I want the keys to be each person's name.
- Come up with a hashing function that converts each person's name to an index between 0 and 1000
- What if I want the index to now be between 0 and 100?
- Or now 0 and 30?

Time:
5 min

3 Components of a Hash Function

- **Initial Hashing Mechanism**
  - Easy to compute (O(1) computation time - not another lookup)
  - Distribute results evenly (don’t cluster around specific numbers)
  - Use the order to help you (otherwise “abc” might be same as “cab”)
- **Modulo (remainder) into your allocated space**
  - large_index % size_of_array = actual_index
- **Collision Resolution Techniques** - they will still happen. How to avoid them?

Time:
5 min

Hashing Mechanism

- A huge variety of hash functions exist for different purposes
  - Basic ones might use ASCII codes for characters
  - Faster ones might use just first and last characters to save on iterating long pieces of text
  - **‘Folding’** - a process (e.g. multiplying the result by a prime) before moving onto the next character helps avoid collisions
  - XOR functions are often used in hashing
- There is a **huge amount of research conducted** into finding good hash functions

Time:
5 min

Modulo / Remainder

- It is necessary to take the answer and find the remainder after division by the size of the array you want to fit it into.
- If you don’t, your array will be unnecessarily long
- The size of the array will depend on how much space you want to take.
- These will be dynamic arrays generally so the more space you leave, the less likely you’ll have to adjust the hash functions (and the collision avoidances that we will discuss!)

Time:
5 min

Collision Resolution

- Once you have your hash function and modulo, you might still get some form of collision because by chance two or more keys indexed to the same place.
- There are 4 key techniques we will look at:
  - Separate Chaining (Open Hashing)
  - Linear Probing (Closed Hashing)
  - Quadratic Probing (Closed Hashing)
  - Double Hashing (Closed Hashing)

Time:
15 min

Collision Resolution - Exercise

- Groups of 4
- Use Google
- Split the 4 techniques amongst yourselves.
- Explore your technique on Google understanding how it works and what are the advantages/disadvantages
- Share it amongst each other asking questions to improve understanding

Time:
5 min

Collision Resolution

- **Open Hashing** - Solving collisions outside of the Hash Table structure
- **Closed Hashing** - Solving collisions inside the Hash Table structure
- **Open Addressing** - Solving collisions outside the index/address defined by the hash function (the same as closed hashing)
- **Hash Table** - a store of indexes and keys, which hash to those indexes. A closed hash table has only one key in each index. Associative Arrays reference these hash tables (using the hash function first) before storing values in the right index.

Time:
5 min

Resolution Techniques

- **Linear Probing** - looking a the next available index linearly, one-by-one until an available space is found (looking 1 after, 2 after, 3 after etc)
- **Quadratic Probing** - looking at the next available index quadratically (looking 1 after, 4 after, 9 after, 16 after etc)
- **Double Hashing** - referring back to another hash function hash2(key) and jumping by multiples of the new hash function (hash2(key), 2*hash2(key) etc)
- **Separate Chaining** - creating linked lists at the point in the hash table (and the corresponding value)

Time:
1 min

Questions