Data Structures

Chaining

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Contacts Application

- Given list of (name, phone), we need to be able to insert/remove/check in some data-structure. Target be very fast!
- Let’s create a class that can hash its objects based on specific key(s)
  - Here we use the name as main entry to search/remove

```
13 struct PhoneEntry {
14     const static int INTERNAL_LIMIT = 2147483647;
15     string name;              // key
16     string phone_number;      // data
17
18     int hash() {
19         return hash_string(name, INTERNAL_LIMIT);
20     }
```

Data and Collisions

- Assume we use these entries and computed their final hash function

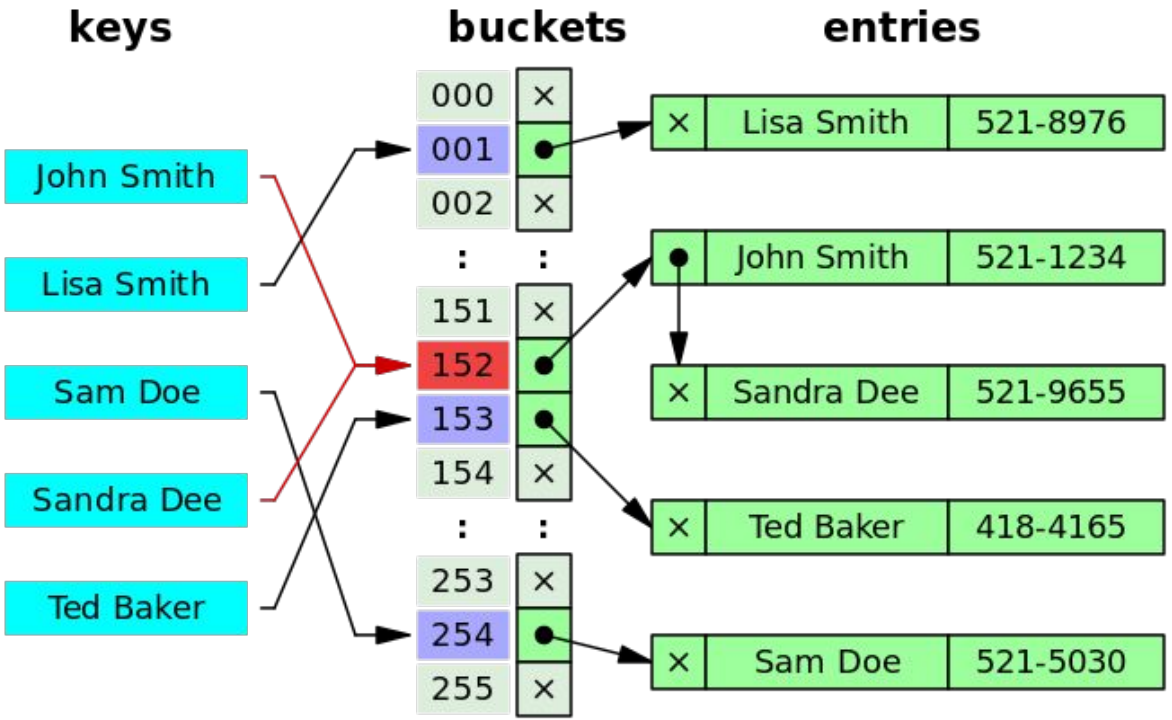
Name (as search key)	Attached data (phone #)	Hash Function code
John Smith	5211234	152
Lisa Smith	5218976	1
Sam Doe	5215030	254
Sandata Dee	5219655	152 (collision)
Ted Baker	4184165	153

Chaining

- Very simple idea to understand and implement
- Our array (aka table), will have in each index (aka bucket) another data structure that insert/remove/search th**e items with same** key
  - Array of linked-lists
  - Array of AVL tree
  - Array of vector
- Implementation can vary, but eventually this is the idea

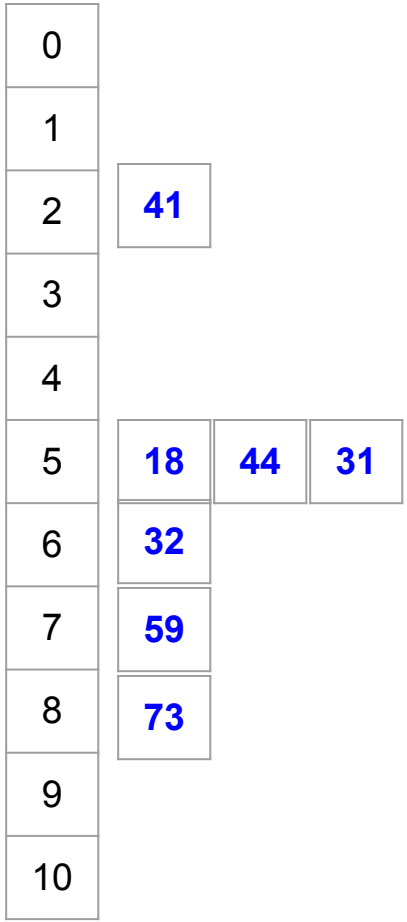
Chaining

- x (nullptr) means no elements so far at this key
- Otherwise, each key is a linked list
- Key 152: linked list of 2 items (John, Sandra)



Integers example

- Assume we have the following integers
- [**18, 41, 22**, 44, 59, 32, 31, 73] and their hash index:
  [5 , 2 , 9 , 5 , 7 , 6 , 5 , 8]
- Assume our table has 11 buckets



Side note: vector< vector<int> >

- vector<int> ⇒ dynamic 1D array of integers
- vector<vector<int> > ⇒ dynamic 1D array: each element is a vector of int.
- Below is vector of 3 items.
  - First element v[0] is a vector of 6 elements.
  - second element v[1] is a vector of 2 elements.
- As you see, if we have items with same key, we can have vector for them
  - **vector<vector<PhoneEntry>> table;**

1	10	2	5	13	6
7	25				
15	6	22	40		

Side note: vector< vector<int> >

- Please, practice creation and manipulation before next lecture

```
vector<vector<int>> v(3);
v[0] = {1, 10, 2, 5, 13, 6};
v[1].push_back(7);
v[1].push_back(25);
v[2] = {15, 6, 22, 40};
cout<<v[0][3]<<"\n"; // 5
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."