Data structures and algorithms

Tutorial 11

Amr Keleg

Faculty of Engineering, Ain Shams University

May 14, 2019

Contact: amr_mohamed@live.com

Outline

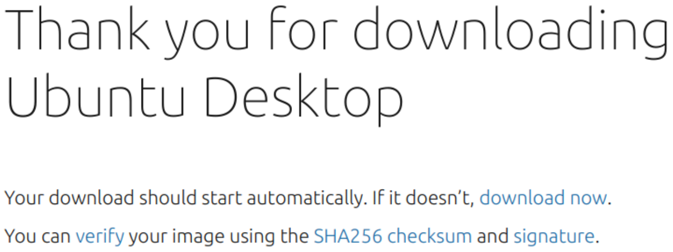
Outline

1 Hashing
- Definition
- Problem Details
- How does hashing work?
- How to solve collisions? (Collision Resolution Techniques)

- Hashing is a type of algorithm which takes any size of data and turns it into a fixed-length of data.
- Hashing is a one way mapping, e.g: You can find the hash for a certain value BUT You can't find the value given its hash.

Applications:

- Instead of storing passwords in a database, store the corresponding hashes.
- To ensure that a file has been downloaded correctly, `https://www.ubuntu.com/download/desktop/thank-you?version=18.04.2&architecture=amd64`



Outline

But hashing has another important application:
Given a set of key-value pairs, store these values such that the searching complexity is optimised.
Example:

- Name: Section
- Marawan: 3
- Chelsea: 1
- Nada: 3
- Mariam: 3
- Omar: 2
- Asim: 1

Options:

```
// Option 1
vector<string> names;
vector<int> section;

names.push_back("Chelsea");
section.push_back(1);

names.push_back("Marawan");
section.push_back(3);

.....

How to check to what section does Asim belong?
```

Options:

```
// Option 2 - A map is internally implemented as a BST
map<string, int> section;
section["Chelsea"] = 1;
section["Marawan"] = 3;
.....

How to check to what section does Asim belong?
if (section.find("Asim") != section.end())
  // Found
  cout<< section["Asim"];
```

Options:

```
// Option 3 - A hashed array
unordered_map<string, int> section;

section["Chelsea"] = 1;
section["Marawan"] = 3;
.....

How to check to what section does Asim belong?
if (section.find("Asim") != section.end())
  // Found
  cout<< section["Asim"];
```

How does option 3 really work??

Outline

- Create an array of size 11 (in practice the array should have a size bigger than the available data).
- We want to map each name (key) to a unique index using a **HASHING FUNCTION**
- The hashing function can be:
  - Map each character to an int (a - 0 , b - 1, c - 2, ...).
  - Add the values of the last two characters for each key(name).
  - Use the modulus (%11) to make the hash in range 0-10

```
int compute_hash(string s){
  int hash_value = 0;
  for (int index= s.size()-2; index < s.size(); index++)
  {
    hash_value += s[index] - 'a';
  }
  return hash_value % 11;
}
```

- Name: Hash(Name)
- Chelsea: 4
- Marawan: 2
- Omar: 6
- Mariam: 1
- Nada: 3
- Asim: 9

- What if we had a new name **Mohammad**.
- The hash value is 3 "The same as HASH(**Nada**)".
- A COLLISION !
- Can you explain why did it happen?

Outline

Open Hashing:

- Each bucket isn't just a single element, it's a container.
- For example, each bucket is a linked list instead of a single index in the array. (Array of linked lists)

Things that we need to do:

- Add a new item
- Search for an item
- Delete an item

What is the average/worse case complexity of searching for a key?
N items and D buckets, N is more than D.

How about having an array of Binary Search Trees?
What is the average/worse case complexity for searching for a key?

Closed Hashing:
If there is a collision, find another empty place for the key.

Linear Probing:
$Hi(x) = (H(x) + i) \% D$

- Compute $H0(x) = H(x) \% D$
- Is there a collision?
    - No, Insert the item in $H0(x)$
    - Yes, Find $H1(x) = (H(x) + 1) \%D$, Is there a collision?
        - No, Insert the item in $H2(x)$
        - Yes, Find $H2(x) = (H(x) + 2) \%D$, Is there a collision?
        - ....

Things that we need to do:

- Add a new item
- Search for an item
- Delete an item

How to search for an item (key, value)?

- Compute H0(key)
- Is there an item in index H0(key)?
  - No, The item doesn't exist - RETURN
  - Yes, Check whether the value stored at index H(key) is actually = value.
    - The Value at index H0(key) == value - Item exists
    - The Value at index H0(key) != value - Find H1(key) and do the same checks.

How to delete an item(key, value)?

- Search for the item first.
- Delete it.
- Any problems here?

Solve sheet's questions.

Things to check in the lecture's slides:

- Two level hashing
- Different hashing functions for strings

Feedback form: `https://forms.gle/BZ76rh8hfth3Pxfu5`