Data structures and algorithms

Tutorial 11

Amr Keleg

Faculty of Engineering, Ain Shams University

April 30, 2020

Contact: amr_mohamed@live.com

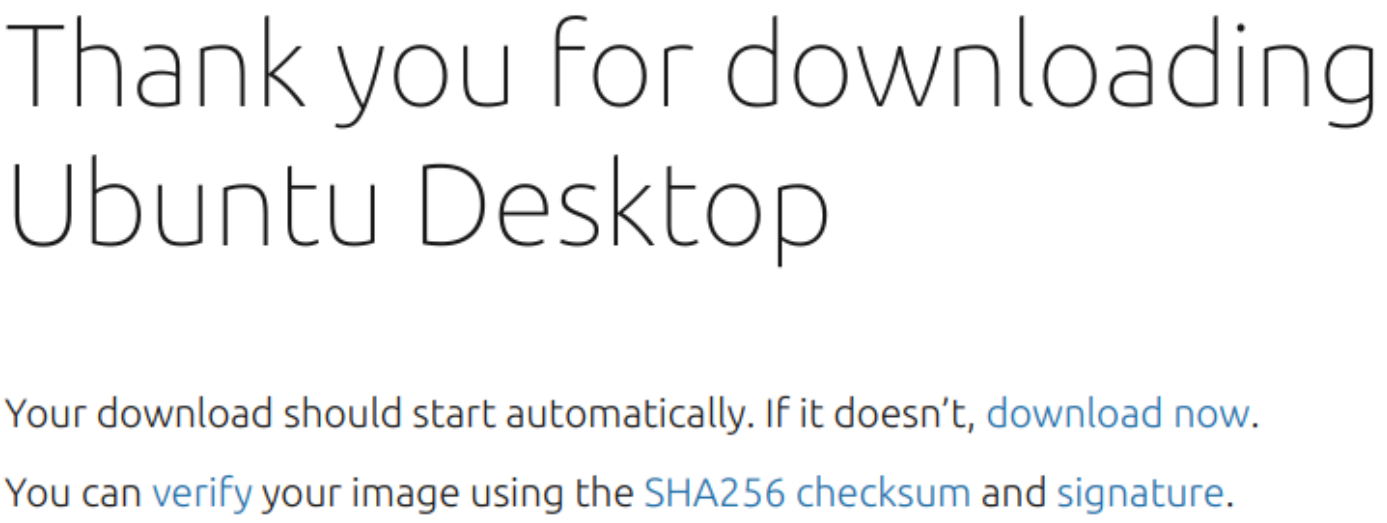
Outline

Outline

- 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"];
```

Complexity

Average case: constant.
Worst case: linear in container size.

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.

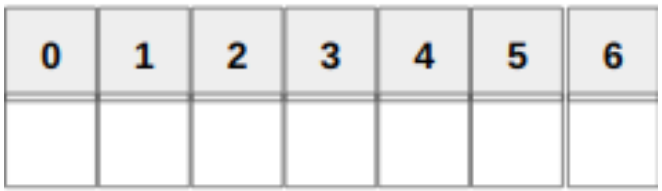
How to delete an item(key, value)?

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

Q5. Given the key values for some students 2341, 1234, 1189, 2829, 430, 422, 454, 597, 2920, students key ranges between 400 to 3000.
Show the resulting hash tables after inserting the student data in the given order with each of these collision strategies.
- Using 1:1 direct mapping, no hashing

0	1	...	400	..	430	..	1189	..	1234	..	2341	..	2839	..	3000
					v430		v1189		v1234		v2341		v2839		

Q5. Given the key values for some students 2341, 1234, 1189, 2829, 430, 422, 454, 597, 2920, students key ranges between 400 to 3000.
Show the resulting hash tables after inserting the student data in the given order with each of these collision strategies.
- Using open hashing with a linked list emerging from each bucket; bucket size = 7



ID	Hash(ID)
2341	3
1234	2
1189	6
2829	1
430	3
422	2
454	6
597	2
2920	1

Q5. Given the key values for some students 2341, 1234, 1189, 2829, 430, 422, 454, 597, 2920, students key ranges between 400 to 3000.
Show the resulting hash tables after inserting the student data in the given order with each of these collision strategies.
- Using open hashing with a linked list emerging from each bucket; bucket size = 11

0	1	2	3	4	5	6	7	8	9	10

ID	Hash(ID)
2341	9
1234	2
1189	1
2829	2
430	1
422	4
454	3
597	3
2920	5

Q5. Given the key values for some students 2341, 1234, 1189, 2829, 430, 422, 454, 597, 2920, students key ranges between 400 to 3000.
Show the resulting hash tables after inserting the student data in the given order with each of these collision strategies.
- Using closed hashing with a hash table of size 40 and second hashing function hi(x) = (x +i) % 40

0	1	2	3	4	5	6	7	8	9	10

11	12	13	14	15	16	17	18	19	20	21

22	23	24	25	26	27	28	29	30	31	32

33	34	35	36	37	38	39

ID	Hash(ID)
2341	21
1234	34
1189	29
2829	29
430	30
422	22
454	14
597	37
2920	0

Q5. Given the key values for some students 2341, 1234, 1189, 2829, 430, 422, 454, 597, 2920, students key ranges between 400 to 3000.
Show the resulting hash tables after inserting the student data in the given order with each of these collision strategies.
- Using closed hashing with a hash table of size 40 and second hashing function hi(x) = (x+ 3*i) % 40

0	1	2	3	4	5	6	7	8	9	10

11	12	13	14	15	16	17	18	19	20	21

22	23	24	25	26	27	28	29	30	31	32

33	34	35	36	37	38	39

ID	Hash(ID)
2341	21
1234	34
1189	29
2829	29
430	30
422	22
454	14
597	37
2920	0

Q6. Which hashing function is more uniform in the range 0:9 assume you have key values between 200 and 300: H= x %10, H= (x*x)%10, H= int(sqrt(x)) %10
Why choosing a uniform function during hashing is important?

Q6. Which hashing function is more uniform in the range 0:9 assume you have key values between 200 and 300: H= x %10, H= (x*x)%10, H= int(sqrt(x)) %10
Why choosing a uniform function during hashing is important?

H= x%10

Q6. Which hashing function is more uniform in the range 0:9 assume you have key values between 200 and 300: H= x %10, H= (x*x)%10, H= int(sqrt(x)) %10
Why choosing a uniform function during hashing is important?

H= x%10 H= int(sqrt(x))%10	10 items in each bucket

Q6. Which hashing function is more uniform in the range 0:9 assume you have key values between 200 and 300: H= x %10, H= (x*x)%10, H= int(sqrt(x)) %10
Why choosing a uniform function during hashing is important?

H= x%10 H= int(sqrt(x))%10 H= (x*x)%10	10 items in each bucket sqrt(200) = 14.1421, sqrt(300) = 17.3205

Q6. Which hashing function is more uniform in the range 0:9 assume you have key values between 200 and 300: H= x %10, H= (x*x)%10, H= int(sqrt(x)) %10
Why choosing a uniform function during hashing is important?

H= x%10	10 items in each bucket
H= int(sqrt(x))%10	sqrt(200) = 14.1421, sqrt(300) = 17.3205
H= (x*x)%10	How is this option actually distributed?

What is the value of (x*x) % 10?

What is the value of (x*x) % 10?

Option # 1: For x = 12: x*x = 144, x*x % 10 = 4

What is the value of (x*x) % 10?
Option # 1: For x = 12: x*x = 144, x*x % 10 = 4
Option # 2: What is the value of x % 10?

What is the value of (x*x) % 10?
Option # 1: For x = 12: x*x = 144, x*x % 10 = 4
Option # 2: What is the value of x % 10?
x = C1 * 10 + (x%10) ; C1 is the maximum possible value.

What is the value of (x*x) % 10?
Option # 1: For x = 12: x*x = 144, x*x % 10 = 4
Option # 2: What is the value of x % 10?
x = C1 * 10 + (x%10) ; C1 is the maximum possible value.
e.g: 63 %10 = 6 * 10 + **3**

What is the value of (x*x) % 10?
Option # 1: For x = 12: x*x = 144, x*x % 10 = 4
Option # 2: What is the value of x % 10?
x = C1 * 10 + (x%10) ; C1 is the maximum possible value.
e.g: 63 %10 = 6 * 10 + **3**

x * x = (10 C1 + x%10) * (10 C1 + x%10)

What is the value of (x*x) % 10?
Option # 1: For x = 12: x*x = 144, x*x % 10 = 4
Option # 2: What is the value of x % 10?
x = C1 * 10 + (x%10) ; C1 is the maximum possible value.
e.g: 63 %10 = 6 * 10 + **3**

x * x = (10 C1 + x%10) * (10 C1 + x%10)
x * x = 100 C1 * C1 + 20 * C1 * (x%10) + (x%10) * (x%10)

What is the value of (x*x) % 10?
Option # 1: For x = 12: x*x = 144, x*x % 10 = 4
Option # 2: What is the value of x % 10?
x = C1 * 10 + (x%10) ; C1 is the maximum possible value.
e.g: 63 %10 = 6 * 10 + **3**

x * x = (10 C1 + x%10) * (10 C1 + x%10)
x * x = 100 C1 * C1 + 20 * C1 * (x%10) + (x%10) * (x%10)
(x * x) %10 =

What is the value of (x*x) % 10?
Option # 1: For x = 12: x*x = 144, x*x % 10 = 4
Option # 2: What is the value of x % 10?
x = C1 * 10 + (x%10) ; C1 is the maximum possible value.
e.g: 63 %10 = 6 * 10 + **3**

x * x = (10 C1 + x%10) * (10 C1 + x%10)
x * x = 100 C1 * C1 + 20 * C1 * (x%10) + (x%10) * (x%10)
(x * x) %10 = 0 + 0 + ((x%10) * (x%10))%10

Val	Val%10	Val*Val%10
200	0	0
201	1	1
202	2	4
203	3	9
204	4	6
205	5	5
206	6	6
207	7	9
208	8	4
209	9	1
210	0	0
211	1	1
212	2	4
213	3	9
214	4	6
215	5	5

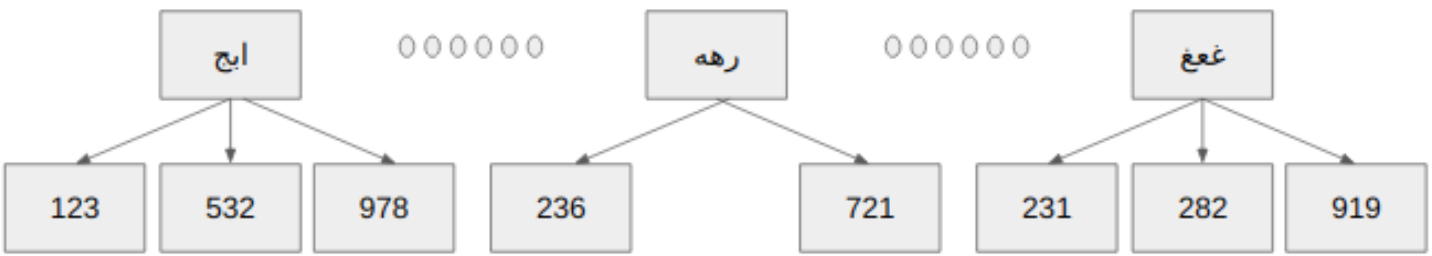
Q7. Design two level of hashing for storing records about the cars in Egypt. In the first level we use the first three letters in the car sign. In the second level we use the remaining three letters. Discuss the required data structures and hence the resulting space and big O time.

Q7. Design two level of hashing for storing records about the cars in Egypt. In the first level we use the first three letters in the car sign. In the second level we use the remaining three letters. Discuss the required data structures and hence the resulting space and big O time.

Answer:
Think of two level hashing as a way of organizing files on a computer.

Q7. Design two level of hashing for storing records about the cars in Egypt. In the first level we use the first three letters in the car sign. In the second level we use the remaining three letters. Discuss the required data structures and hence the resulting space and big O time.

Answer:
Think of two level hashing as a way of organizing files on a computer.



H1/H2	0	1	2	3	4	5
0						
1						
2						
3						

To find the correct location for plate no P:

- Compute H1(P[0:2])
- Compute H2(P[3:5])
- Insert it in the bucket at row H1(P[0:2]) and column H2(P[3:5])

Bonus Idea :D

What are the number of different plate numbers for cars in Egypt?

How to count it?

- What is the count if the plate number is a single Arabic alphabetic character?

- What is the count if the plate number is a single Arabic alphabetic character?
- 28 Different Characters

- What is the count if the plate number is a single Arabic alphabetic character?
- 28 Different Characters
- What is the count if the plate number is a string of two Arabic alphabetic characters?

- What is the count if the plate number is a single Arabic alphabetic character?
- 28 Different Characters
- What is the count if the plate number is a string of two Arabic alphabetic characters?
- For Alef, we have 28 different options for the second character. For Baa', we have 28 different options for the second character.
  and so on. So the total count is 28 * 28.

- What is the count if the plate number is a single Arabic alphabetic character?
- 28 Different Characters
- What is the count if the plate number is a string of two Arabic alphabetic characters?
- For Alef, we have 28 different options for the second character. For Baa', we have 28 different options for the second character.
  and so on. So the total count is 28 * 28.
- What is the count if the plate number is a string of three Arabic alphabetic characters?

- What is the count if the plate number is a single Arabic alphabetic character?
- 28 Different Characters
- What is the count if the plate number is a string of two Arabic alphabetic characters?
- For Alef, we have 28 different options for the second character. For Baa', we have 28 different options for the second character.
  and so on. So the total count is 28 * 28.
- What is the count if the plate number is a string of three Arabic alphabetic characters?
- 28 * 28 * 28.

- What is the count if the plate number is a single Arabic alphabetic character?
- 28 Different Characters
- What is the count if the plate number is a string of two Arabic alphabetic characters?
- For Alef, we have 28 different options for the second character. For Baa', we have 28 different options for the second character.
  and so on. So the total count is 28 * 28.
- What is the count if the plate number is a string of three Arabic alphabetic characters?
- 28 * 28 * 28.

Therefore, the total number of different plates
is

- What is the count if the plate number is a single Arabic alphabetic character?
- 28 Different Characters
- What is the count if the plate number is a string of two Arabic alphabetic characters?
- For Alef, we have 28 different options for the second character. For Baa', we have 28 different options for the second character.
  and so on. So the total count is 28 * 28.
- What is the count if the plate number is a string of three Arabic alphabetic characters?
- 28 * 28 * 28.

Therefore, the total number of different plates is28*28*28*10*10*10 = 21,952,000

Things to check in the lecture's slides:

- Two level hashing
- Different hashing functions for strings