Analysis of Algorithm

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010

Week 12

❖ Rabin-Karp Algorithm

Definition: It is a string-searching algorithm that uses hashing to efficiently locate a pattern within a text. It is particularly useful when searching for multiple patterns or handling large texts.

Steps:

- **Hashing**: Calculate a hash value for the pattern and the first segment of the text of the same length as the pattern.
- **Sliding Window**: Slide through the text one character at a time, recalculating the hash for each new segment.
- **Comparison:** If the hash of the current segment matches the hash of the pattern, check the characters to confirm the match.
- **Output**: Continue this process to find all occurrences of the pattern in the text.

Time Complexity:

- **Best case: O(n+m)** (hashing avoids direct character comparisons for mismatched windows).
- **Worst case: O(nm)** (due to hash collisions requiring character-by-character verification).

Example:

Text: abedabcabc

Pattern: abc

- Compute hash of **abc** and the first three characters of the text **abc**.
- Slide the window, recompute the hash, and check matches:
    1. No match for **abc** bed, etc., until **abc**.

- Return the starting index of matches.

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010

Example:

Text: ccacedba

Pattern: dba

- Compute hash of **dba** and the first three characters of the text **dba**.
- Slide the window, recompute the hash, and check matches:
  - No match for **dba**, bed, etc., until **dba**
- Return the starting index of matches.

❖ Hashing

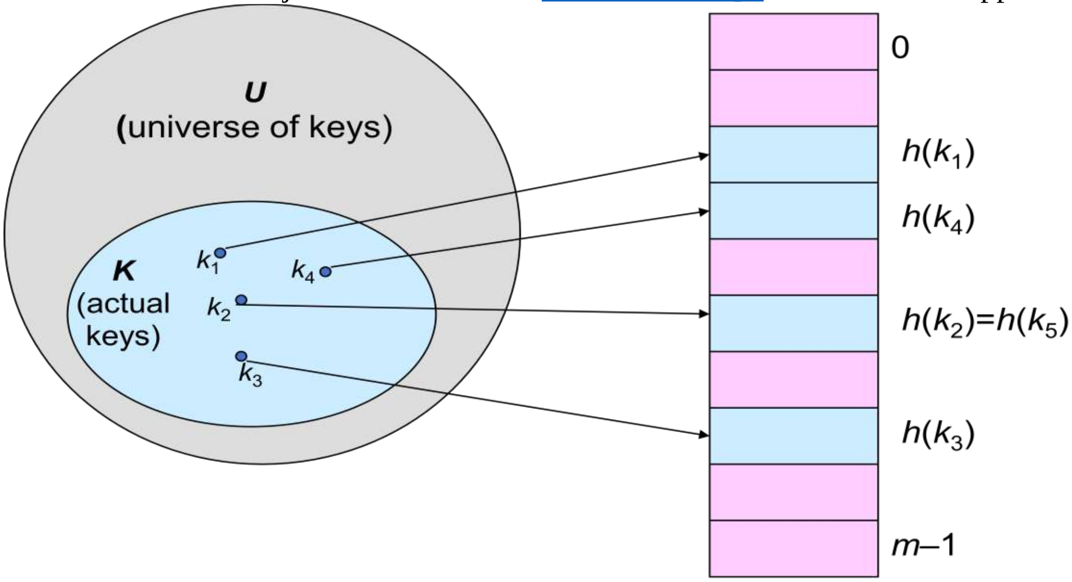
- Hash function *h*: Mapping from *U* to the slots of a hash table *T*[0..*m*–1].

  $h : U \rightarrow \{0,1,\ldots, m-1\}$
- With arrays, key *k* maps to slot *A*[*k*].
- With hash tables, key *k* maps or "hashes" to slot *T*[*h*[*k*]].
- *h*[*k*] is the *hash value* of key *k*.

<u>Hash Function</u>

- Distribute keys among cells of the hash table as evenly as possible
- A hash function has to be easy to compute

A Hashing process is depicted in the given diagram in which keys are going to mapped on hash table….that particular keys will be calculated by hash functions.



A Simple Example to understand Hashing concept

Example

A, FOOL, AND, HIS, MONEY, SOON, PARTED

Hash function: Assume taking mod by 13.

(19+15+15+14)%13=11 (SOON)

Keys	A	FOOL	AND	HIS	MONEY	SOON	PARTED	
add	1	9	6	10	7	11	12	

In above example values for alphabet is assigned according to the position of alphabet in English language… like A has value 1, B has 2 and so on. If we will take mod by 13; there will be maximum 13 values calculated by hash function.

<u>Issues with Hashing</u>

- Multiple keys can hash to the same slot – collisions are possible.
  - Design hash functions such that collisions are minimized.
  - But avoiding collisions is impossible.

- Design collision-resolution techniques.

- Search will cost $\Theta(n)$ time in the worst case.
  - However, all operations can be made to have an expected complexity of $\Theta(1)$.

Collision

Collision means one hash value for two are more than two key words. As illustrated by the example below.

A, FOOL, AND, HIS, MONEY, ARE, SOON, PARTED

Hash function: Assume taking mod by 13.

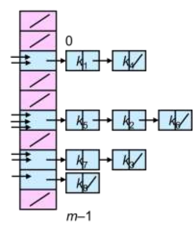
Collision between SOON and ARE

(19+15+15+14)%13=11 (SOON)

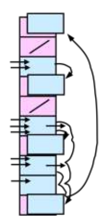
(1+18+5)%13=11 (ARE)

Methods to Resolve Collision

- Chaining:
  - Store all elements that hash to the same slot in a linked list.
  - Store a pointer to the head of the linked list in the hash table slot.



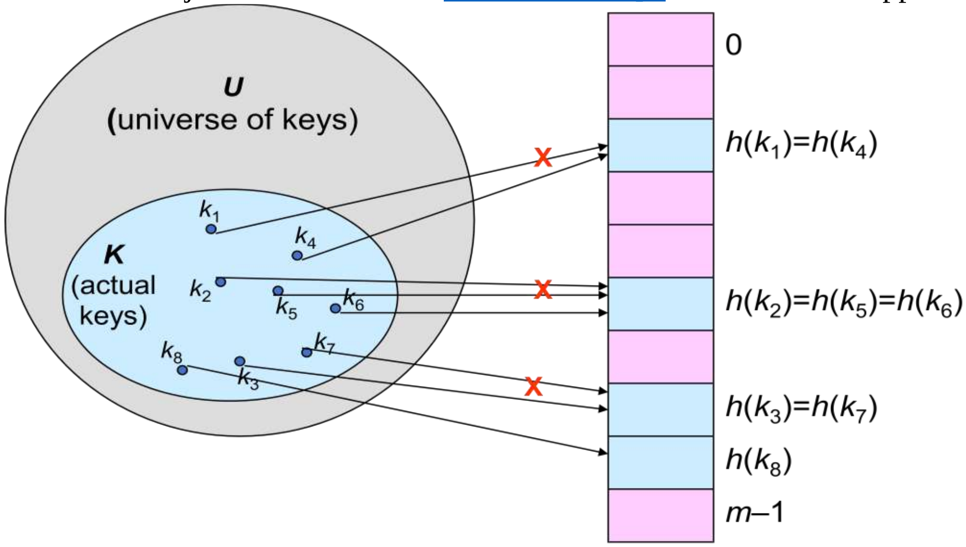
- Open Addressing:
  - All elements stored in hash table itself.
  - When collisions occur, use a systematic (consistent) procedure to store elements in free slots of the table.



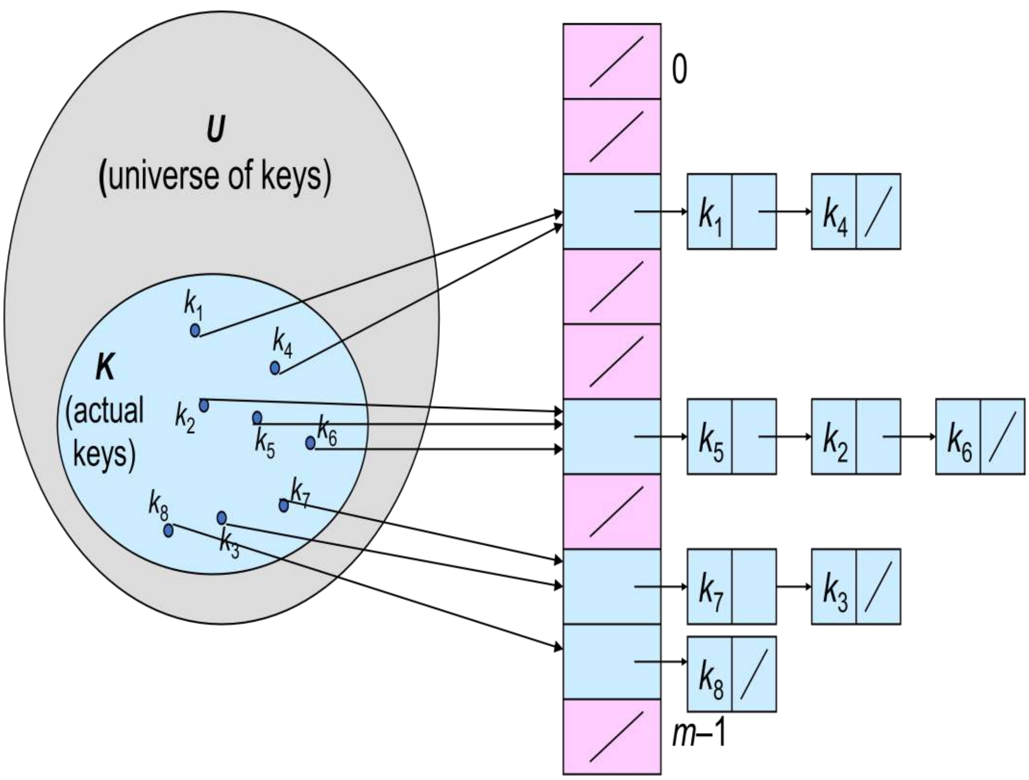
Collision Resolution by Chaining

Collision problem is depicted in following diagram

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010



In solution a chain is made to store multiple key words against one hash value as depicted in the following diagram



Insertion in Hash Table

HASH_INSERT (T,K)

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010


```
i ← 0
Repeat j ← h ( k, i )
   IF  T[ j ] = NILL
      THEN  T[ j ]=k
       Return  j
   ELSE i ← i + 1
UNTIL  i = m
ERROR " Hash Table Over Flow"
```

Searching from Hash Table

HASH_SEARCH (T, k)

```
i ← 0
Repeat j ← h ( k, i )
   IF  T[ j ] = k
      THEN   Return  j
   ELSE i ← i + 1
UNTIL  i = m  OR  T[ j ] = NILL
RETURN  NILL
```

Analysis

- Worst case for inserting a key is $\theta(n)$
- Worst case for searching is $\theta(n)$
- Algorithm assumes that keys are not deleted once they are inserted

Deleting a key from an open addressing table is difficult, instead we can mark them in the table as removed (introduced a new class of entries, full, empty and removed).

Example

Given data; **Asim, Tina, Nida, Saim, Amna, Dina, Isma, Tara, Maha and Moiz**.

Analysis of Algorithm

Dr. Naseer Ahmed Sajid   email id: naseer@biit.edu.pk   WhatsApp# 0346-5100010

Hash Function

The hash function sums up each alphabet of a string and then takes its mod (remainder) by 10.

For Collision

This hash function returns the hash key to store into the hash table that may create the same keys.

Apply the following strategies to resolve collision.

(i) **Linear Probing**   **(ii) Chaining**

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
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Solution

Asim = 1+19+9+13 = 42%10 = 2

Tina = 20+9+14+1 = 44%10 =4

Nida= 14+9+4+1= 28 %10 = 8

Saim=19+1+9+13= 42%10 = 2

Amna=1+13+14+1 = 29%10 = 9

Dina= 4+9+14+1= 28 %10 = 8

Isma= 9+19+13+1= 42%10 = 2

Tara=20+1+18+1= 40%10 = 0

Maha=13+1+8+1 = 23 %10 = 3

Moiz=13+15+9+26= 63 % 10 = 3

Linear Probing

0	Dina
1	Tara
2	Asim
3	Saim
4	Tina
5	Isma
6	Maha
7	Moiz
8	Nida

Chaining

		Chain
0	Tara	
1		
2	Asim	Saim, Isma
3	Maha	Moiz
4	Tina	
5		
6		
7		
8	Nida	Dina

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010

9	Amna		9	Amna

Recurrence Relation of Hashing:

- **Linear probing:**

  $C(n)=((m-n+1)(m)*1+((n-1)(m)*(1+c(n-1))$

  $C(n)=1+((n-1)/m)*c(n-1)$

- **Chaining:**

  $C(n)=1+p*c(n-1)$