University of Science, VNU-HCM
Faculty of Information Technology

Data Structure and Algorithm

Search Algorithms

Lecturer: Le Ngoc Thanh
Email: lnthanh@fit.hcmus.edu.vn

HCM City

Outline

- What is search?
- Why need to search?
- Search application
- Basic search algorithms
  - Sequential search
  - Binary search
  - Search using hash table
- String match

What is search?

- ***Search*** is the process of determining the position of an element with a specified feature in a set.

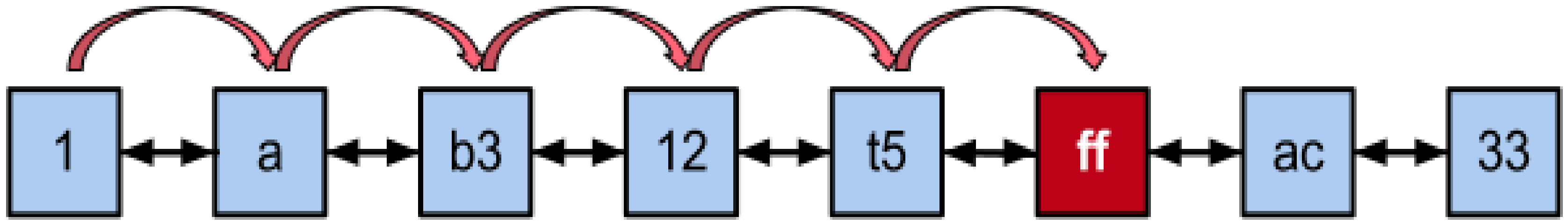
Easy or difficult to search?

- Find the number 12 in the following array?

	0	1	2	3	4	5	6	7	8	9
Array	23	17	97	44	35	10	12	8	5	78

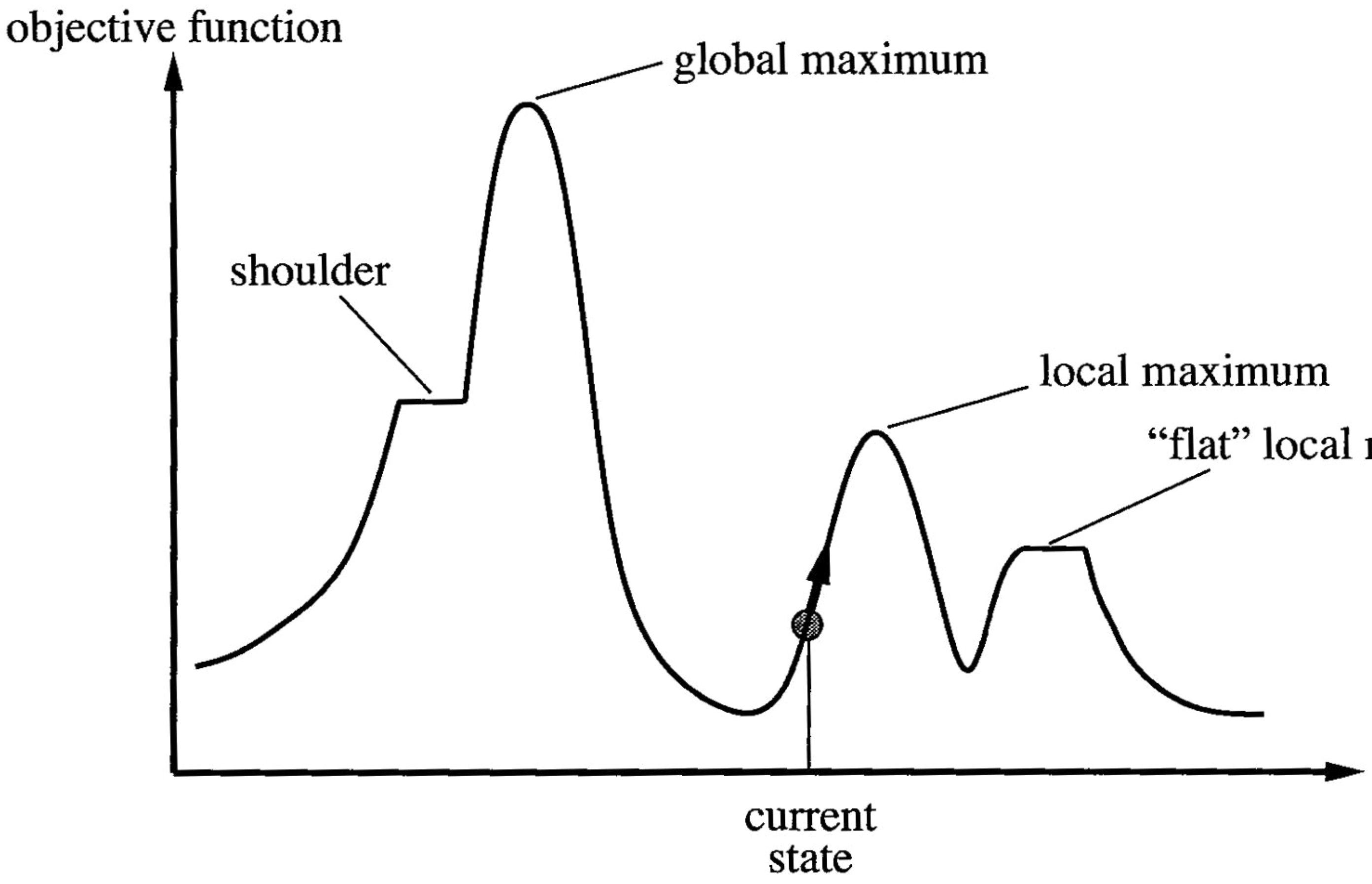
Easy or difficult to search?

- Find the string ff in the following linked list?



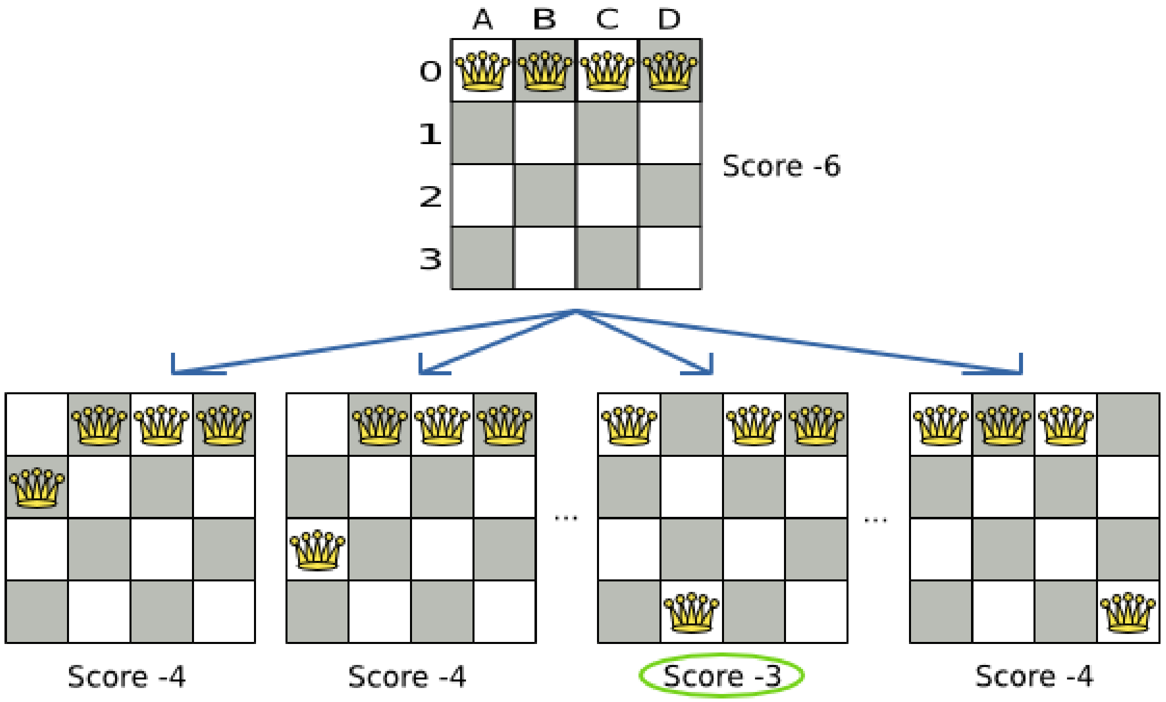
Easy or difficult to search?

- Determining the highest mountain peak when you are blindfolded?



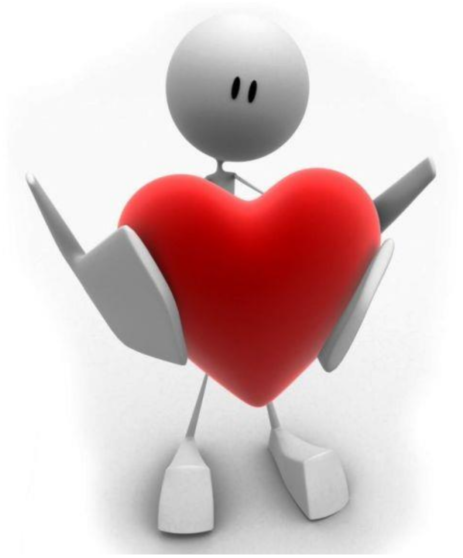
Easy or difficult to search?

- Placing eight chess queens on an 8×8 chessboard so that no two queens threaten each other?



Easy or difficult to search?

- If all of the above is easy then consider this question:
  - Find someone to love "forever"?



Why?

- Easy because:
  - – The power of the computer
  - – The power of storage
  - – The power of the algorithm
- Difficult because:
  - – Big data
  - – Time consuming
  - – Depends on how to store it
  - – The feature is difficult to find or describe
  - – Lack of search knowledge: how the brain works to recognize images, ...

Search application

- Search for the website
  - Google, Yahoo, Bing, …
- Database
  - Search for a record
- …

Simple search algorithm structure

- Input:
  - List which consists of n elements
  - Describe the element to look for
- Output:
  - Returns whether or not the element is present
  - Returns the element to be found.
- Basic operation:
  - Compare

Basic search algorithms

- Linear/sequential search
- Binary search
- Search tree
- Hashing

Linear/sequential search

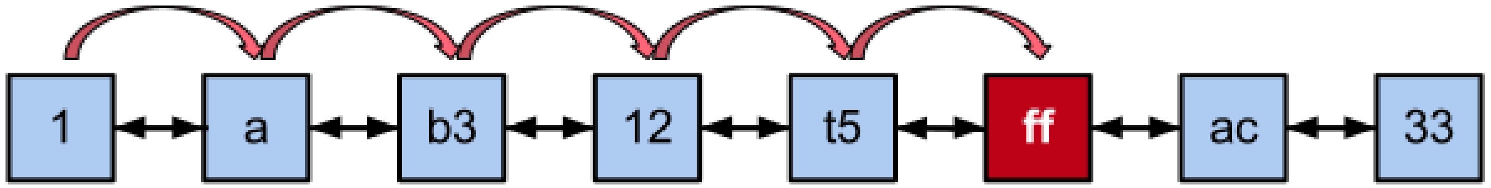
- How do I find a house when I know its address?

Linear/sequential search

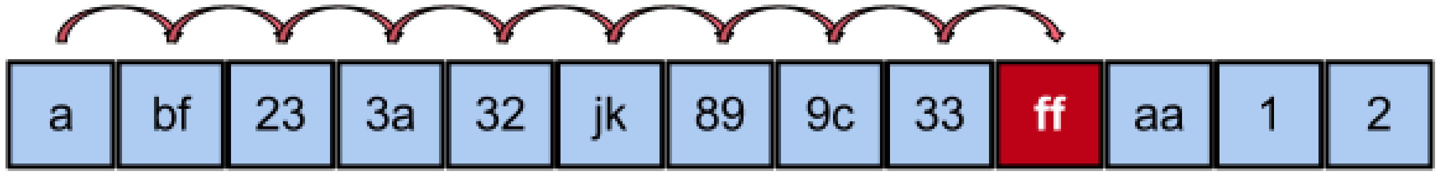
Idea:

- Compare x with the elements of the array in turn until it meets the desired element or runs out of the array

1. Linked Lists search for "ff"

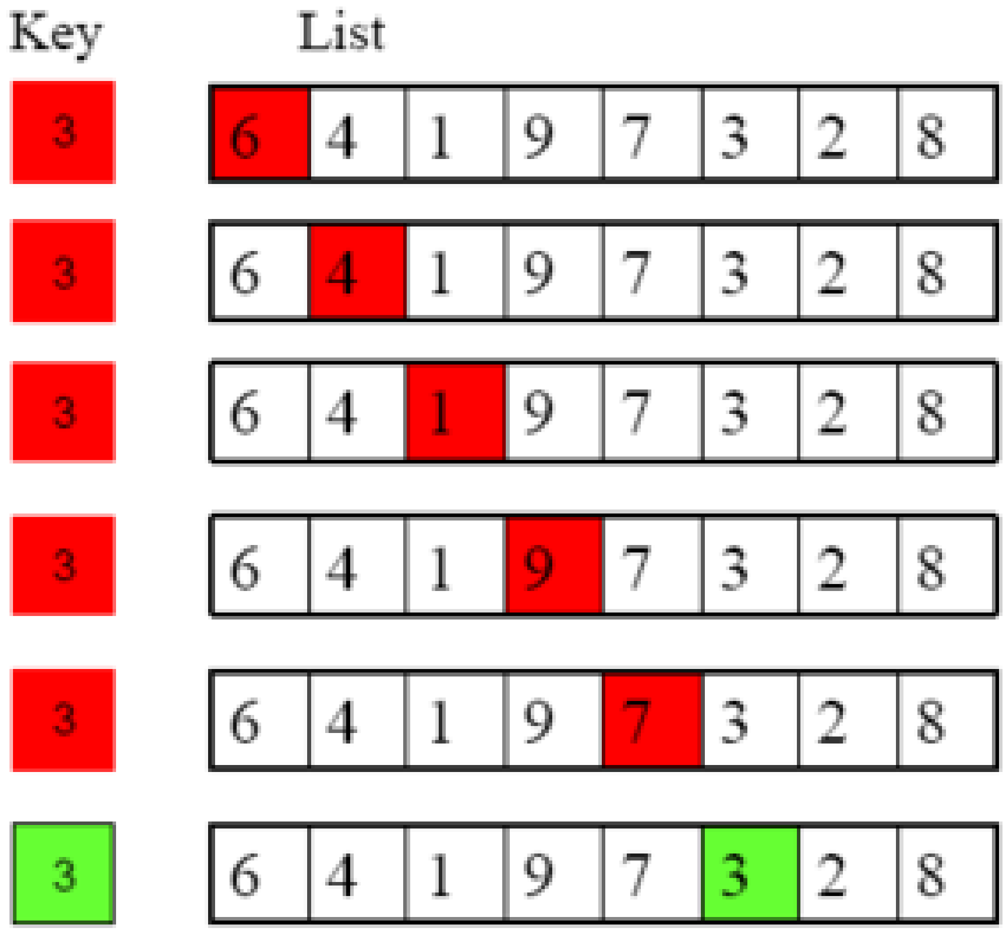


2. Arrays sequential search



Linear/sequential search

- Example:



Linear/sequential search

Pseduo code

```
for (i ←0 to n) do
  if (x = a[i]) then
      return true;
end for
```

Evaluate sequential search

- Advantage:
  - Ease of implementation
- Disadvantage:
  - Long searching time: O(n)

Sentinel Linear Search

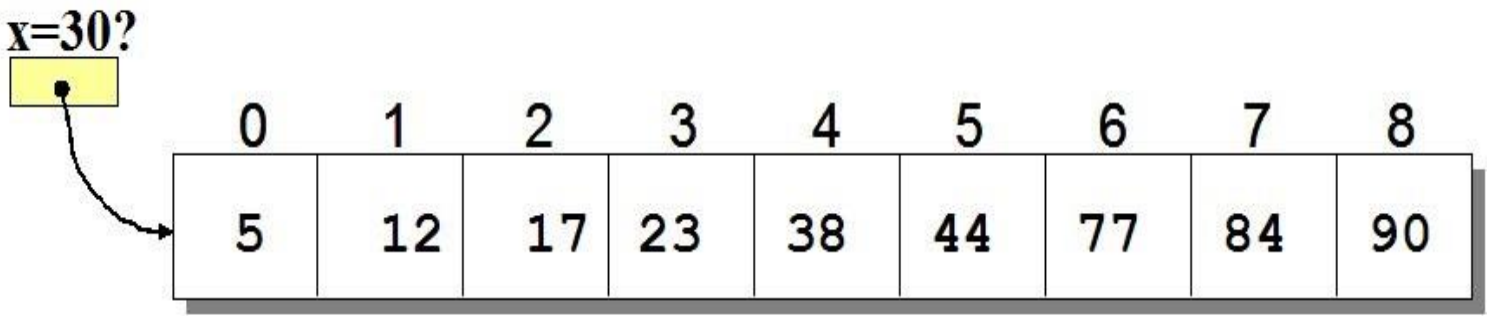
- How many comparisons / tests are there in each sequential search loop?
  - – Checks if the element is x
  - – Checks out of bounds of the array.
- Sentinel element: places the element x at the end of the array → No need to check out of bounds of the array.
  - – Example: A = {1, 25, 5, 2, 37} and x = 6

1	25	5	2	37	6

  - – Return: *n* if not found.

Sequential search on ordinal array

- *If the list is sorted, how good is it for sequential search?*
  - With an ordered array, instead of finding the entire array, we escape when we encounter an element that is larger than the desired one.



Binary search

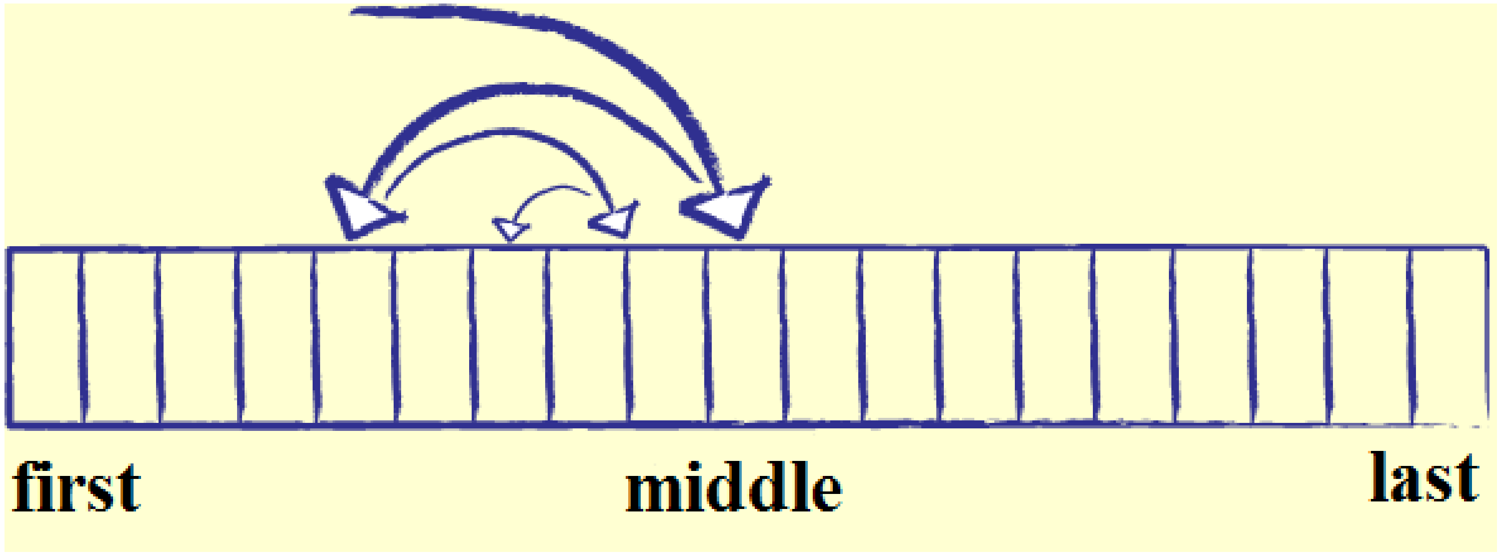
- Once the array is ordered, we take advantage of this to reduce the number of operations.
  - How do you find a new word in a dictionary?

Binary Search

Binary search

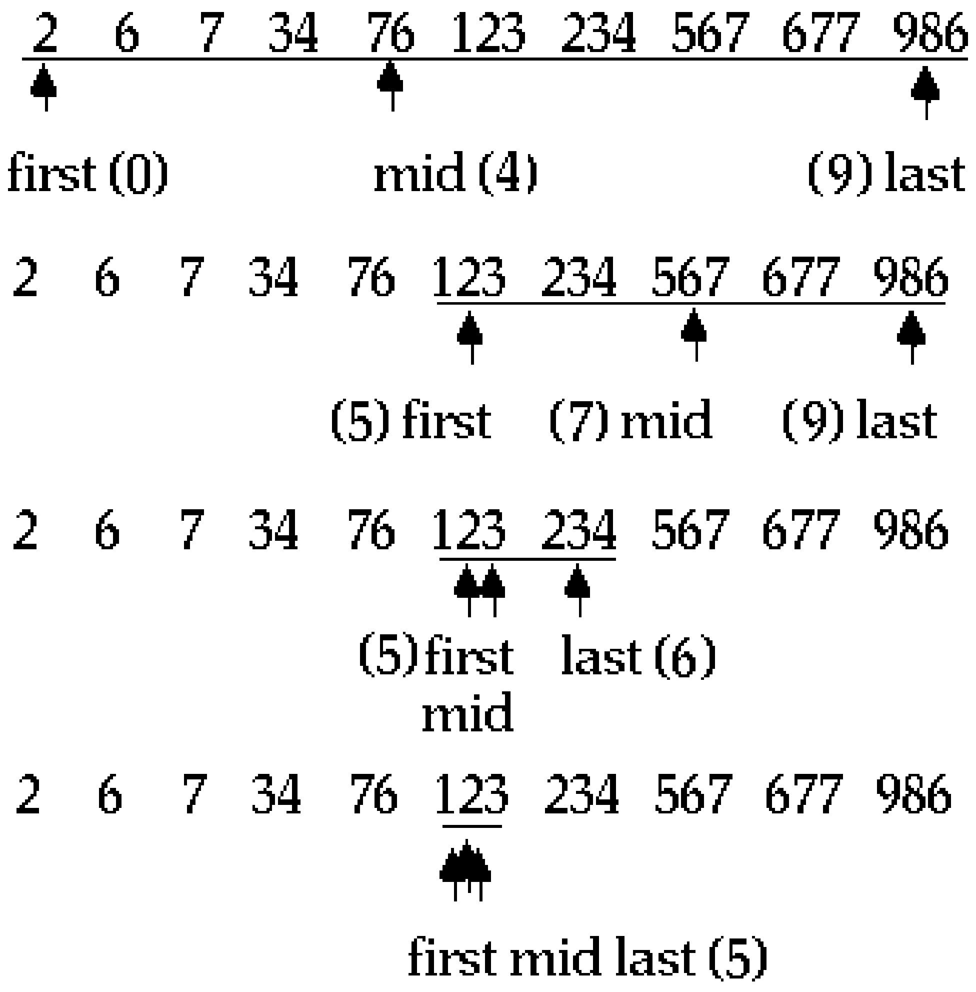
Idea: devide and conquer

- Check the middle element a[m]
- If a[m] = x, return x
- If a[m] > x, find x in the same way in the left sub-sequence
- If a[m] < x, find x in the same way in the right sub-sequence



Binary search

- Find the position of the element x = 123 in the array.



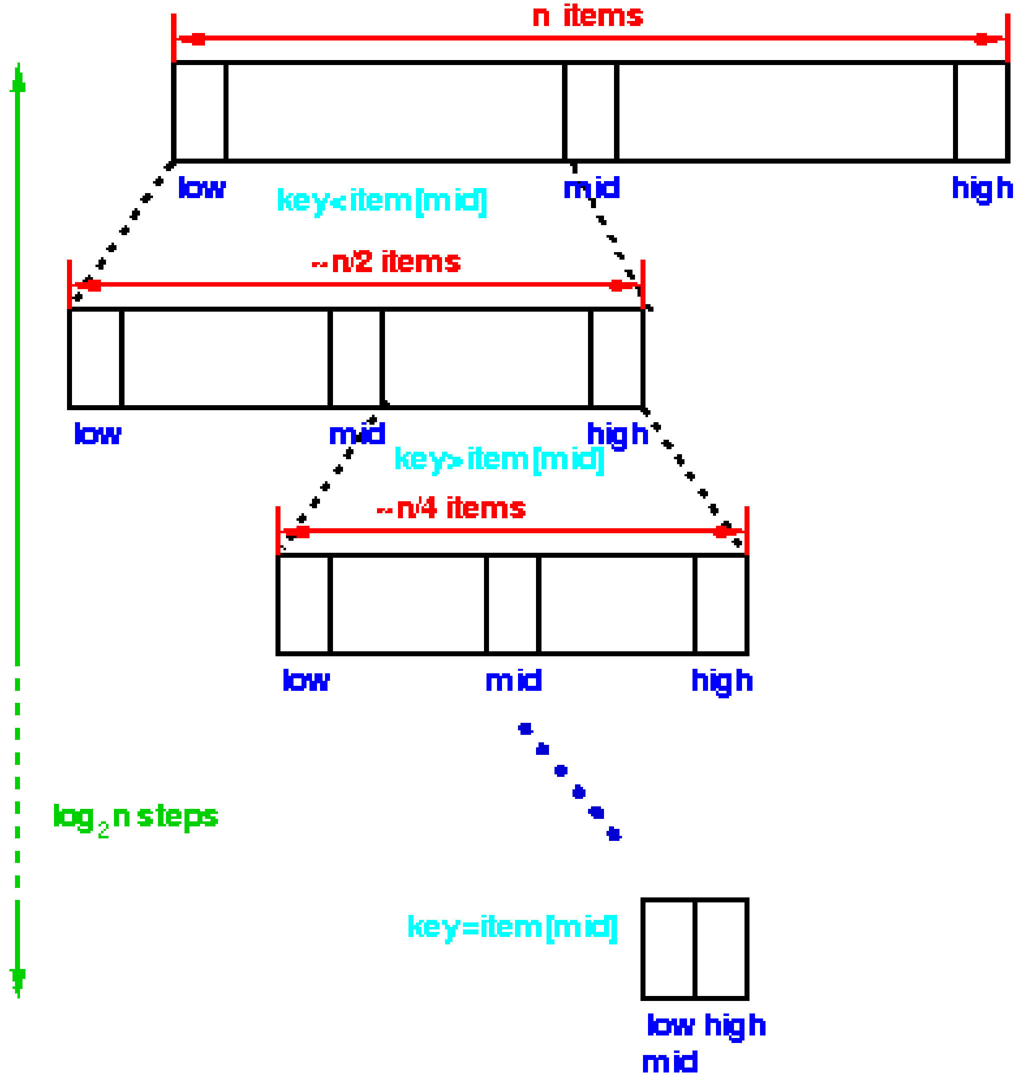
Binary search

Pseudo Code

```
left =0; right=n-1;
while (l ≤ r) do
   m = (left+right)/2;
   if (x = a[m]) then return true; //found
   else
          if (x < a[m]) then
                    right = m-1;    //find on left side
          else
                    left = m+1;     //find on right side
          end if
   end if
end while
return -1;        //not found
```

Evaluate a binary search

- Advantage:
  - Tìm kiếm nhanh hơn tuần tự: $O(\log_2 n)$
- Disadvantage:
  - Arrays must be sorted first.

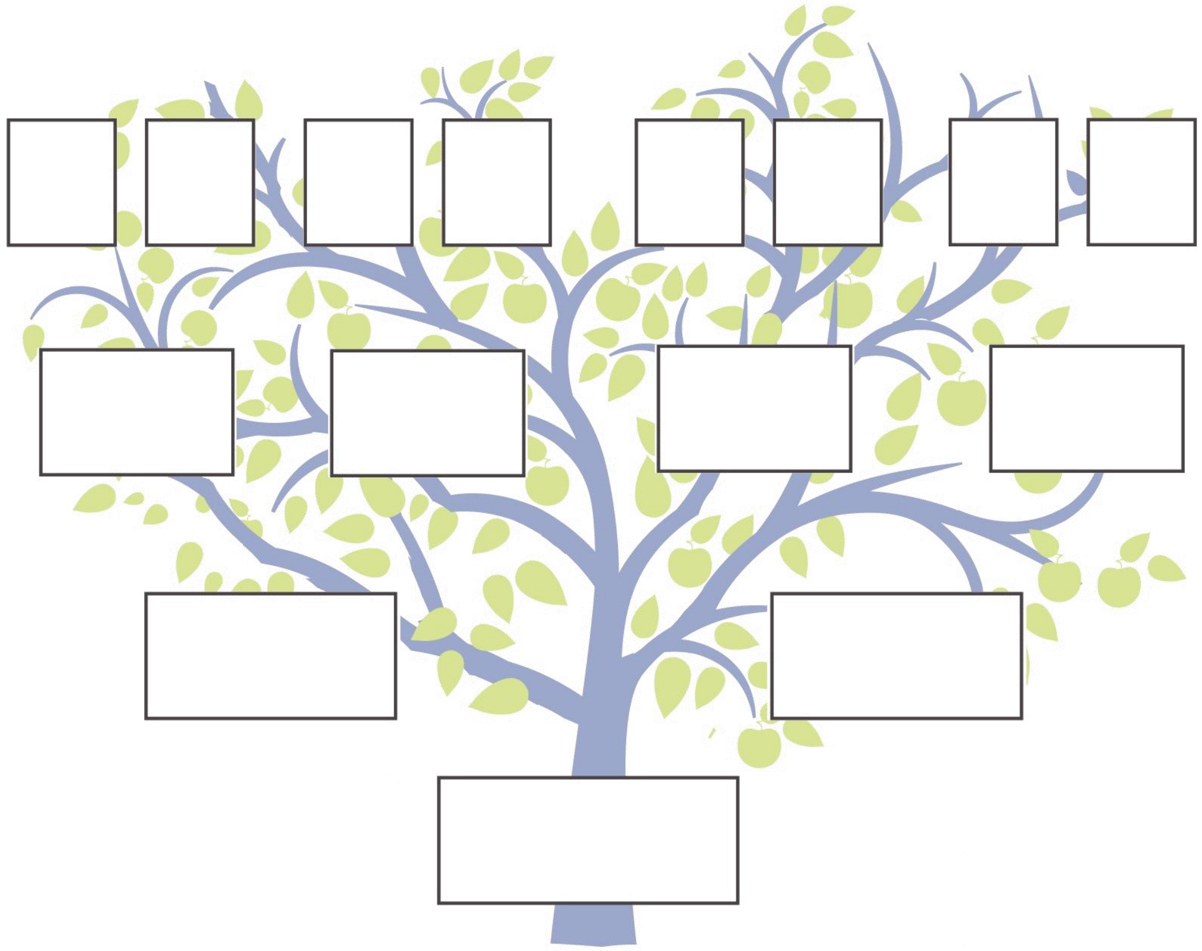


Basic search algorithms

- Linear/sequential search
- Binary search
- Search tree
- Hashing

Search tree

Will be introduced in the next lesson

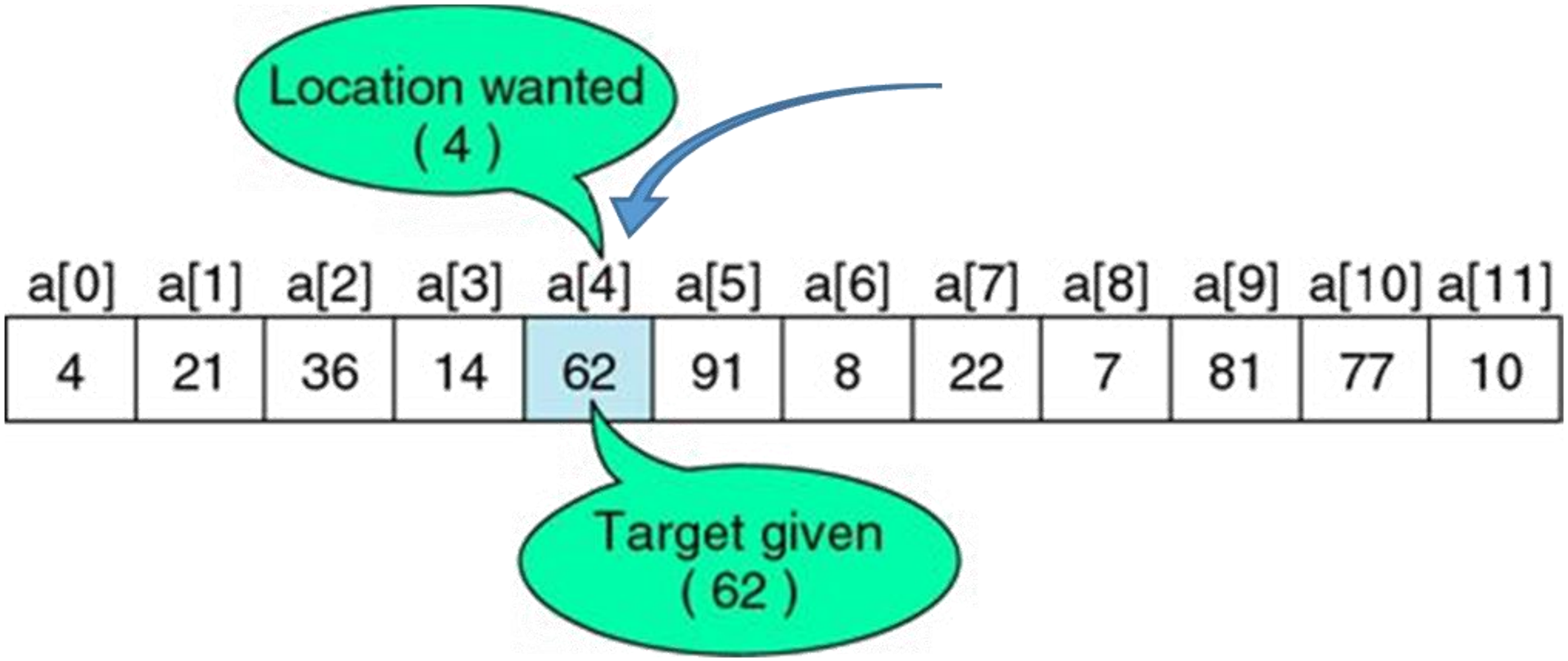


Basic search algorithms

- Linear/sequential search
- Binary search
- Search tree
- Hashing

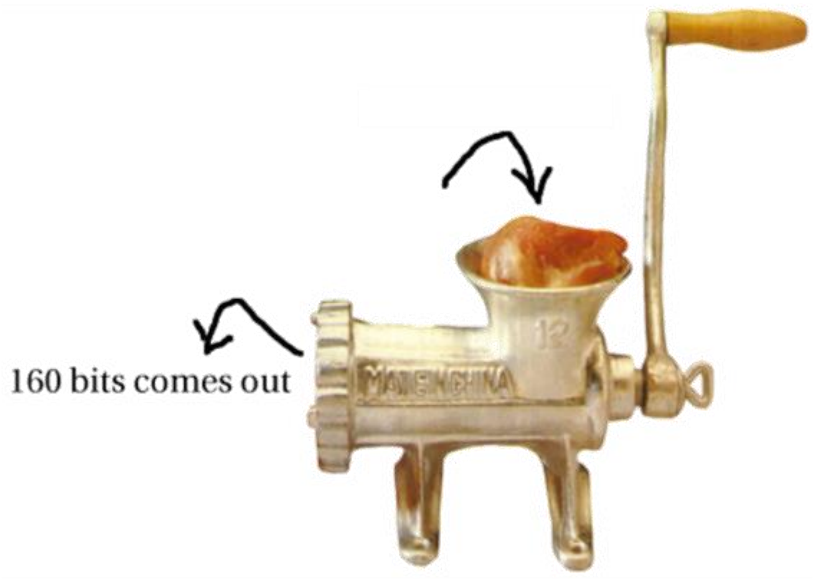
Hashing

- Is there a way to find the element in just one step?



Hashing

- ***Hashing*** is a technique of converting an element into another form to serve a specific purpose such as address, key, …



Process of meat transformation → minced meat

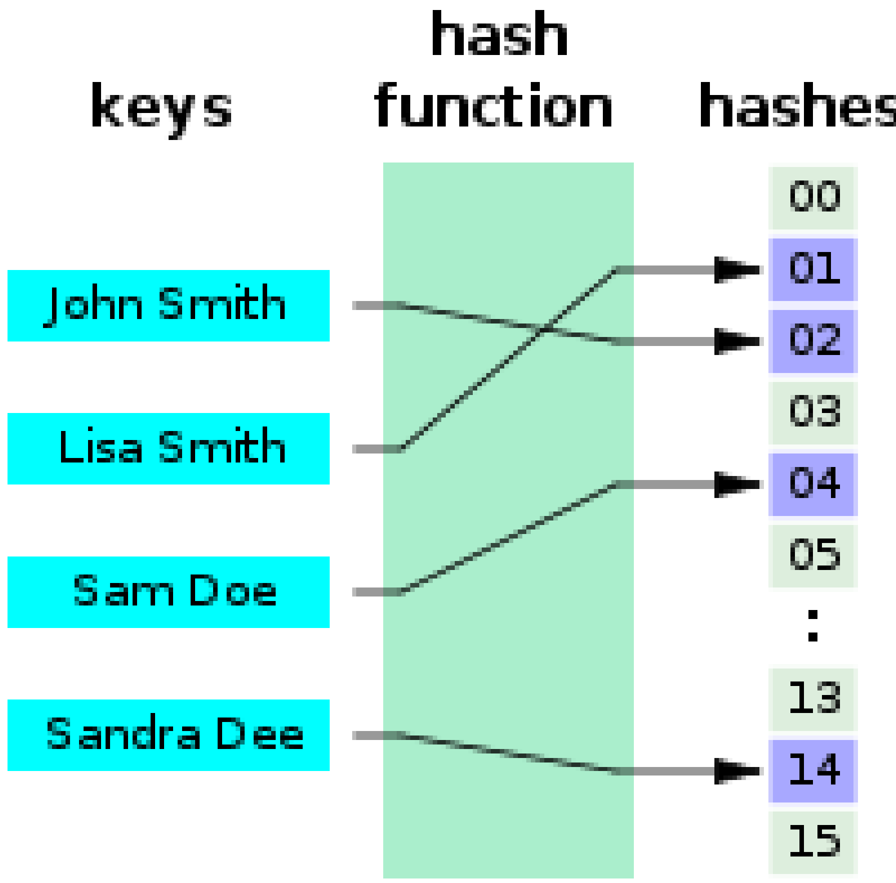
Hashing

- <span style="color:red">Hash function</span> is the mapping from one key to another.

  Example: H(k) = k mod M

$$\begin{tikzcd} H: U \arrow[r] & A \\ k \arrow[r] & a = H(k) \end{tikzcd}$$

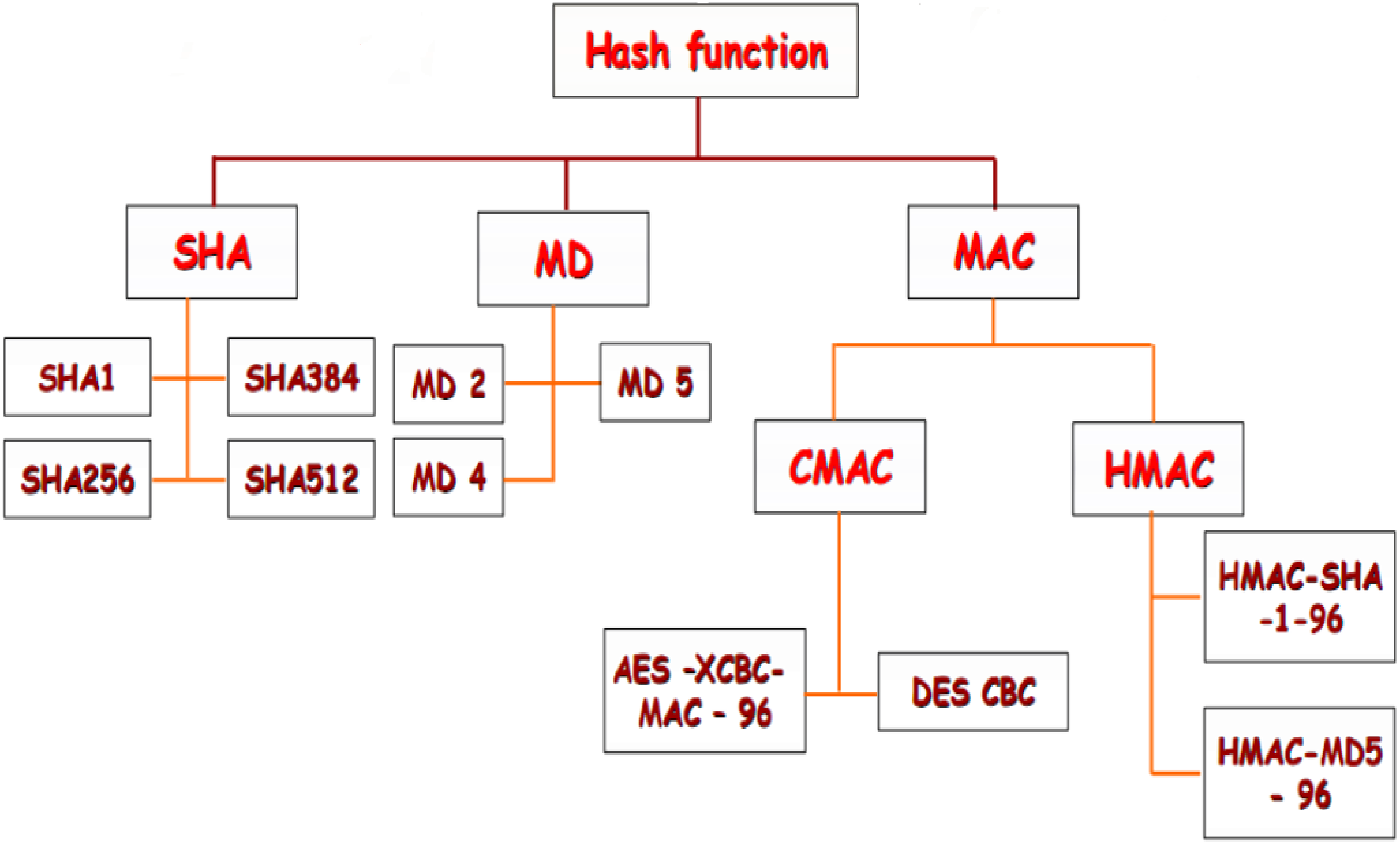
For example, suppose the characters are encoded as follows:
c = 3; a = 1; t=20; s=19.
The word "cats" will be hashed to:
3 + 1 + 20 + 19 = 43



Requirements for hash functions

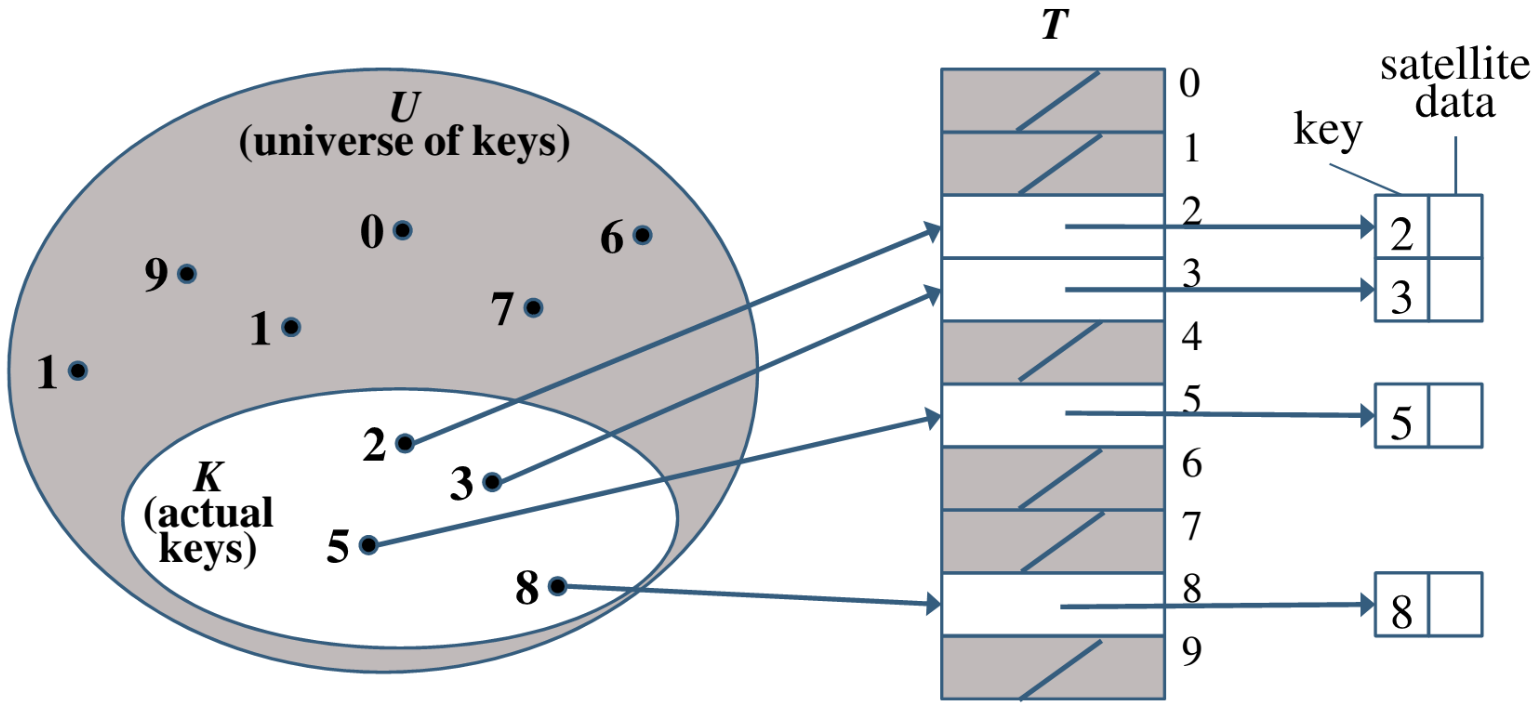
- The hash must run fast, take up less memory
- Each input produces only a single hash value or low collision constraint.
- Computing a inverse (one-way function) is very difficult, time consuming or impossible to calculate.
- There are many different hashing algorithms: SHA-1, SHA-256, MD5, MD2 ... Each algorithm differs in string length after hashing, speed, collision, ...

Some famous hash functions



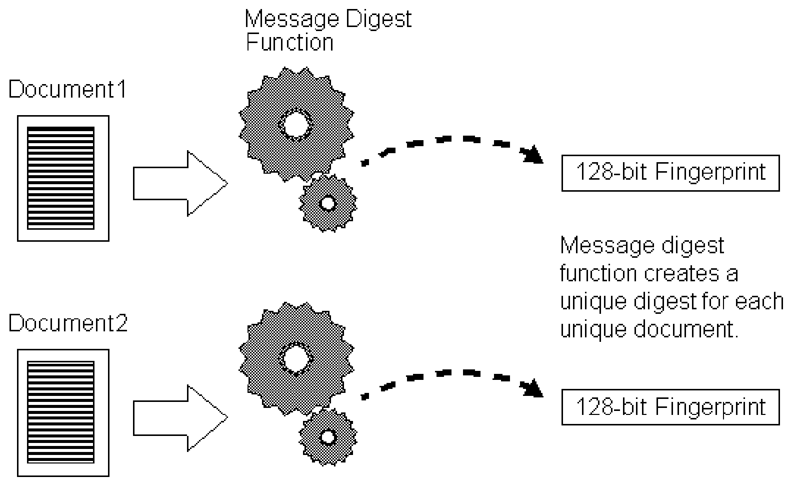
Application of hashing

- Search with O(1)
  - Each element's key / value in the array will be hashing to serve as the address to store the element.
  - When looking for an element, just hashing back, we will get the address and jump to its location.



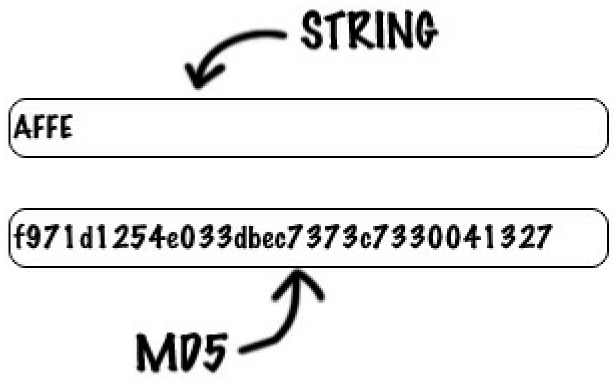
Application of hashing

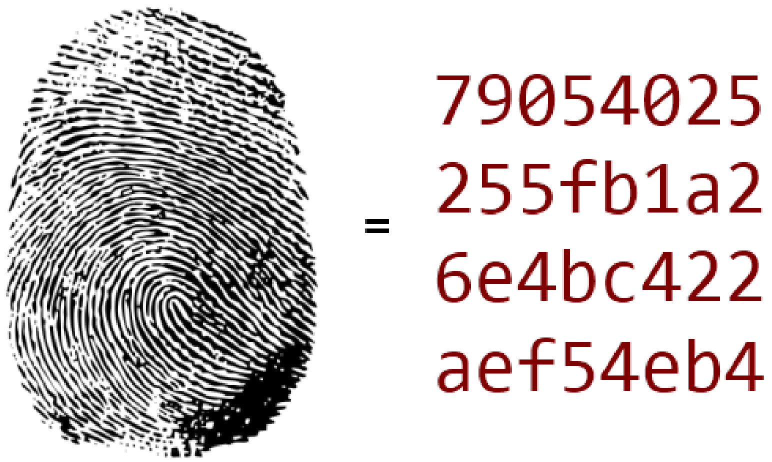
- Check file integrity
  - The two files are hashed into 2 strings. If the strings are identical, the two files are identical.
  - Application:
    - Check if the file transferred on the network is missing any bits
    - Whether the text transmitted over the network has been edited



Application of hashing

- Encode/Decode:
  - – Secure password storage.
  - – Electronic Signature
  - – Use on encryption algorithms





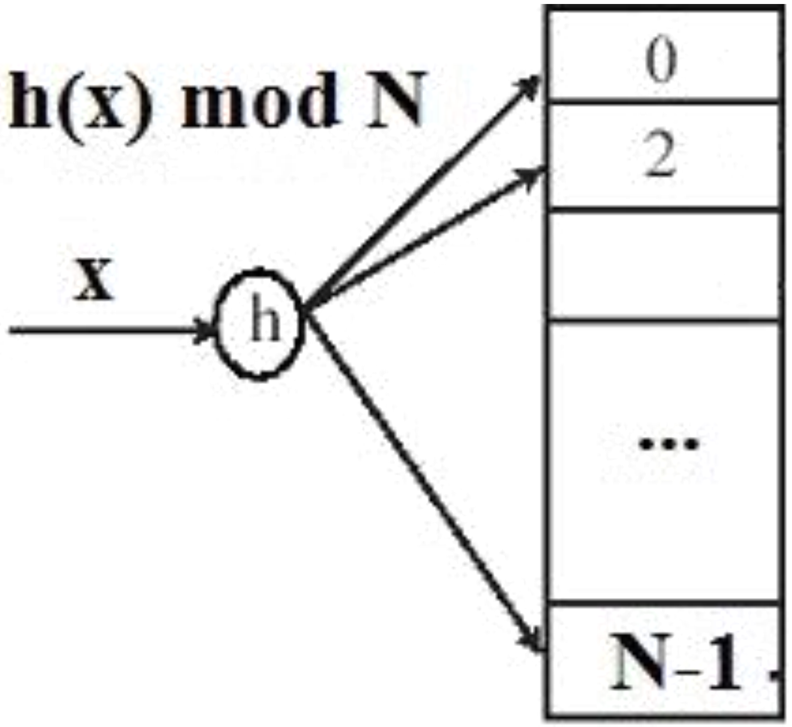
Implementing hash function

- Choose the Hash function :

$$H(x) = x \bmod N$$

where x is the element to add / find

N is the array size



Implementing hash function

- N = 15:

if x =	25	129	35	2501	47	36
H(x) =	10	9	5	11	2	6

- The element will be contained in the array:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
_	_	47	_	_	35	36	_	_	129	25	2501	_	_	_

- Adding, deleting, and editing only costs O(1) but …

Collision

- What happens if x = 65?

x = 65

H(x) = 5

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
_	_	47	_	_	35	36	_	_	129	25	2501	_	_	_

$\downarrow$

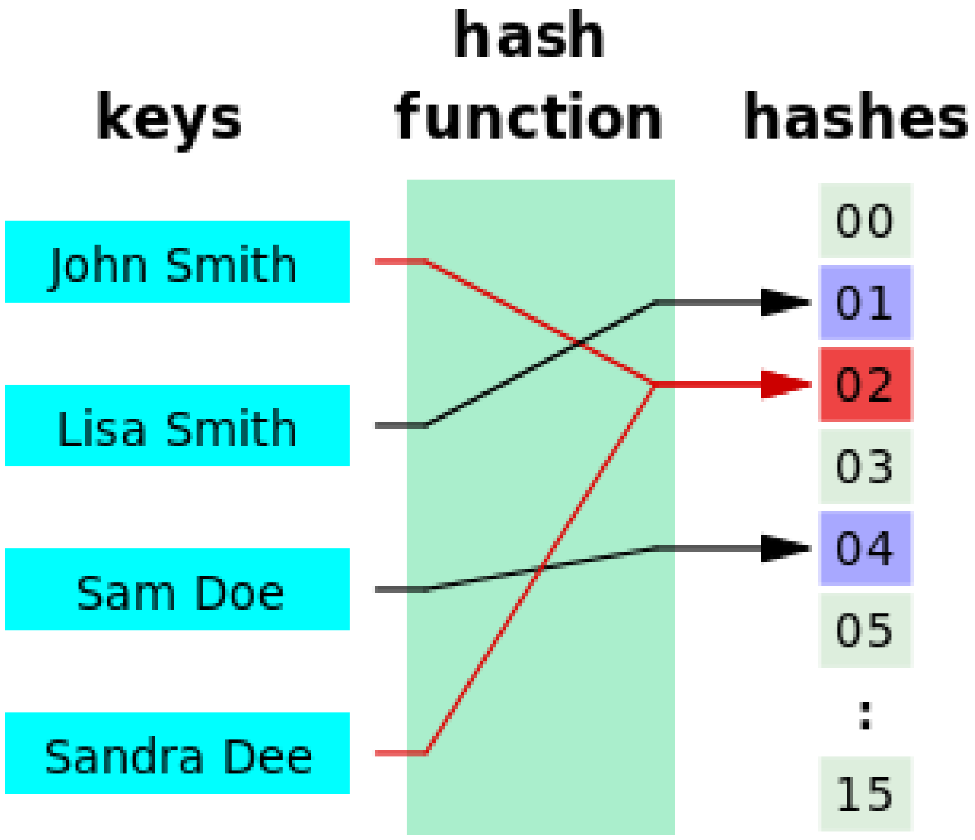
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
_	_	47	_	_	35	36	_	_	129	25	2501	_	_	_
					65(?)									

→ Collision

Collision

- *Collision* is the phenomenon of two or more keys being hashed into the same place.

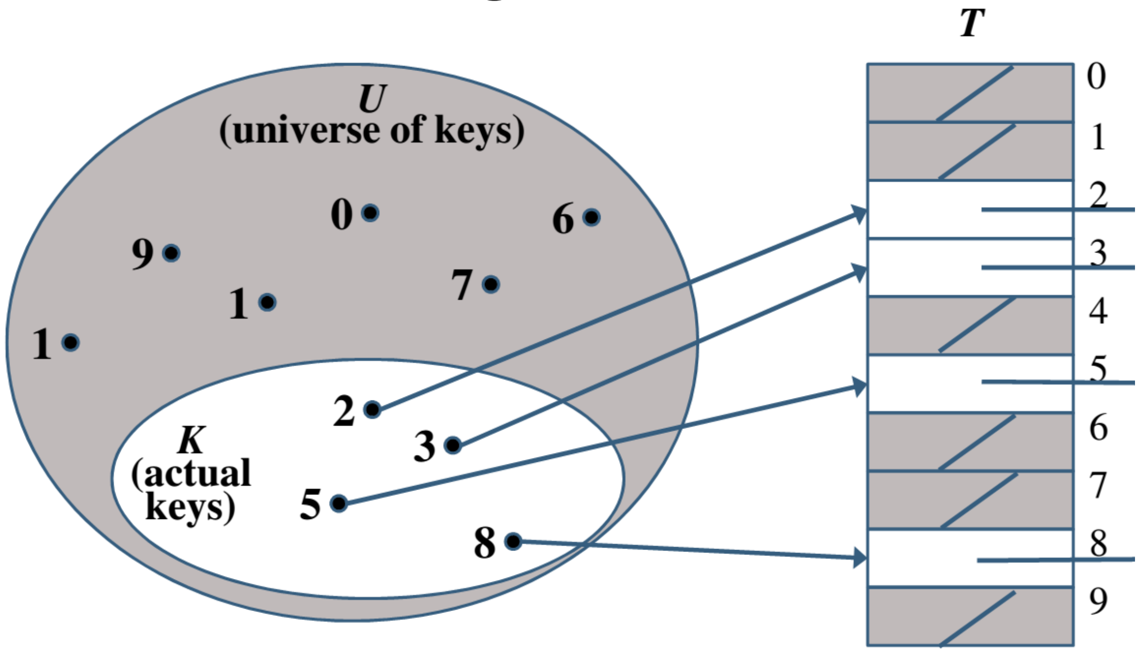
$$\exists k_1, k_2 \in K\colon\ k_1 \neq k_2, H(k_1) = H(k_2)$$



The cause of the collision

Given a key set K to be stored:

- If $|K| \leq N$, whether the collision can happen depends on the hash function.
- If $|K| > N$, the collision is sure to happen no matter how good a hash function is.



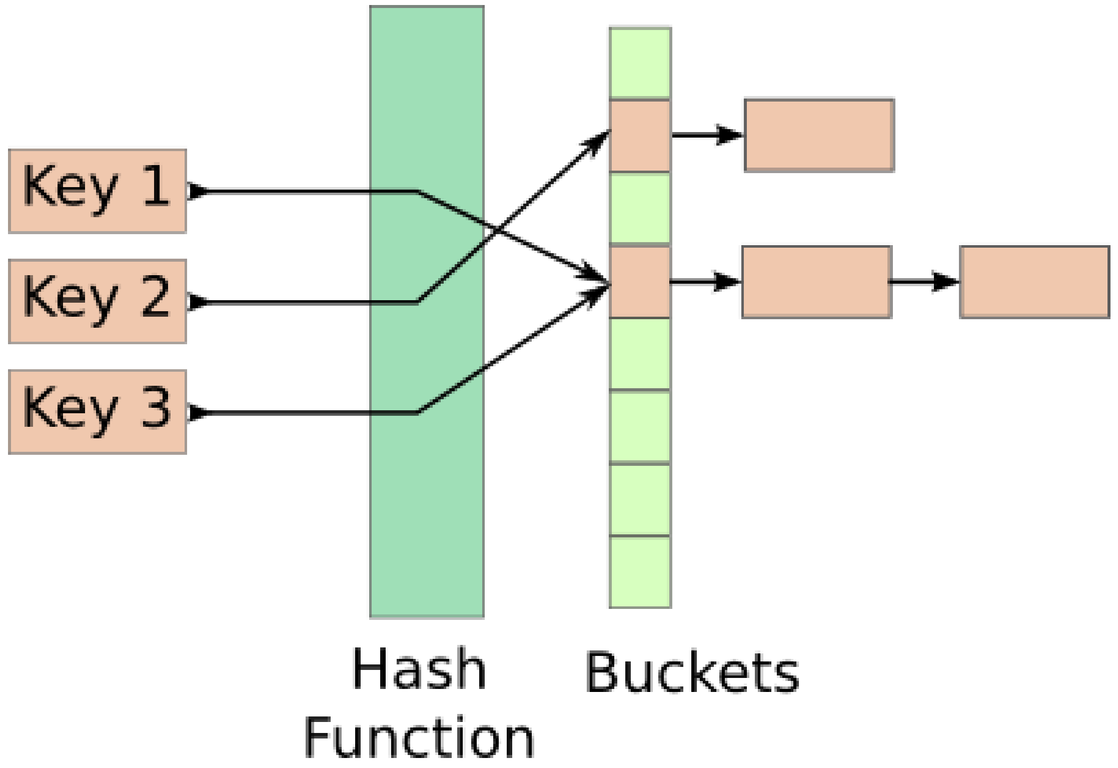
The set of possible keys (U) is often much larger than the actual key (K).

Collision Handling

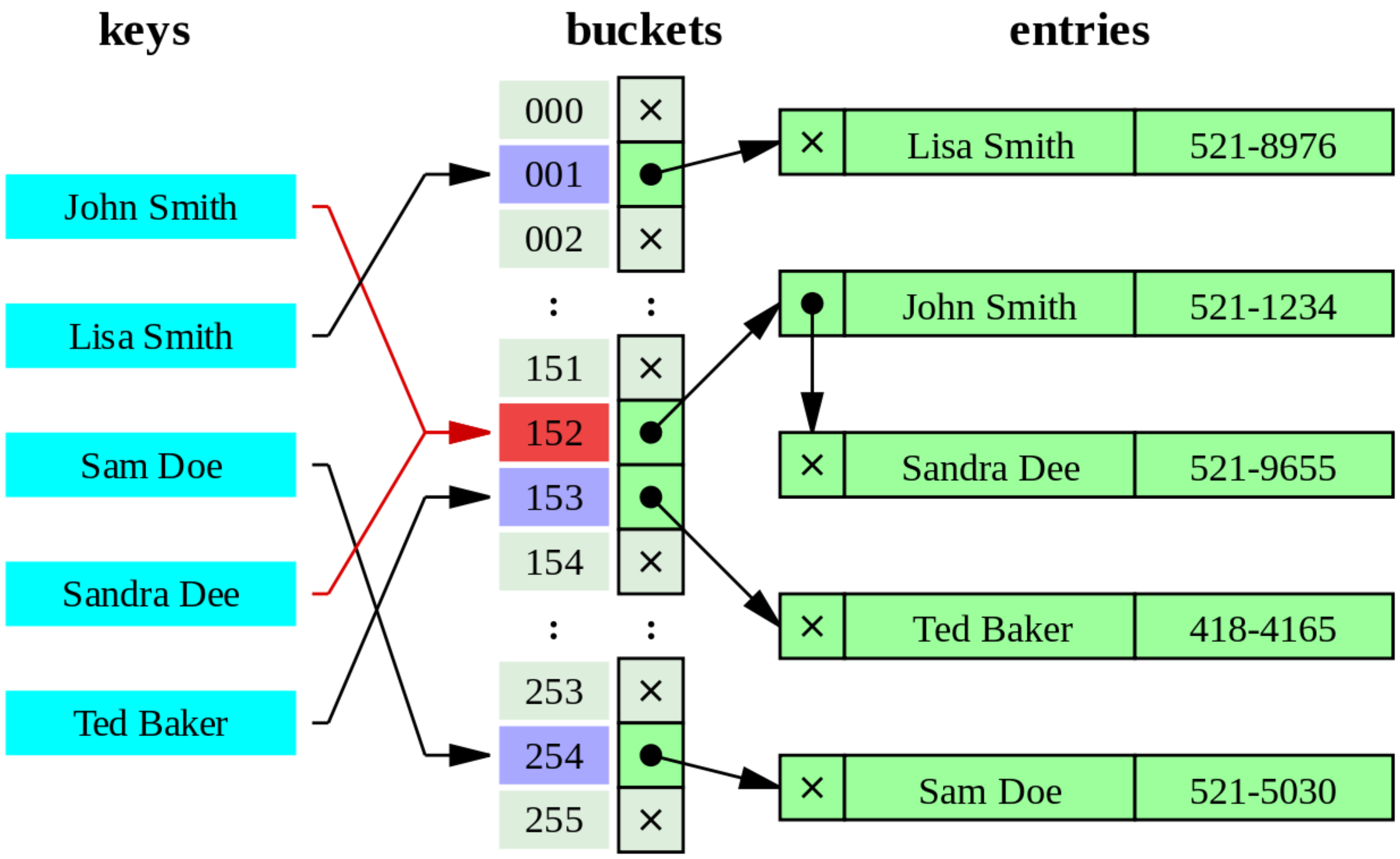
- Chaining approach
- Open-Addressing approach
  - – Linear Probing
  - – Quadratic Probing
  - – Double Hashing

Collision Handling - Chaining

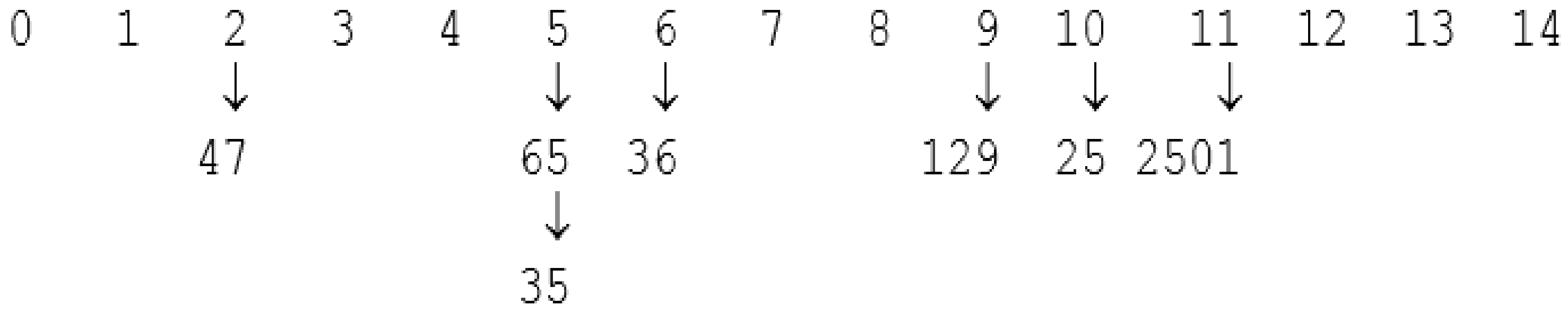
- For each address of the table, we use the linked list to contain the collision key.
- From there, we have a hash table containing the headers of the linked lists.



Chaining

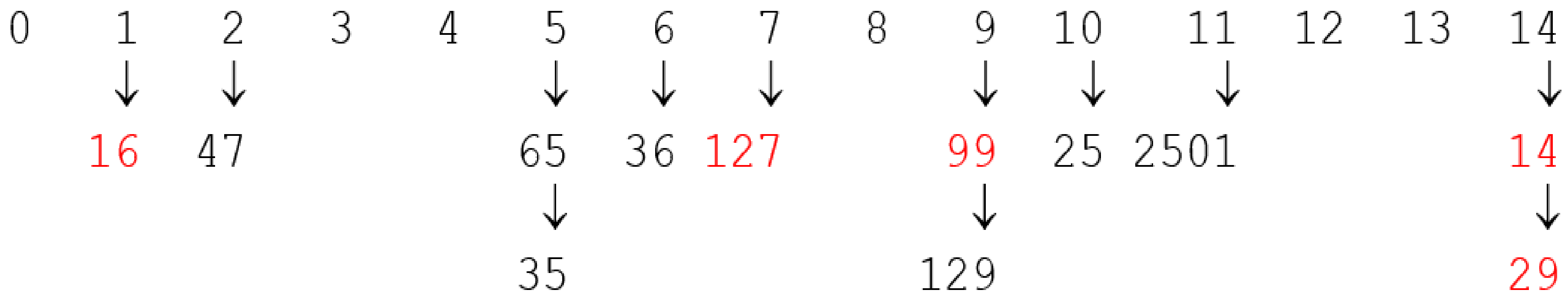

keys
buckets
entries
John Smith
Lisa Smith
Sam Doe
Sandra Dee
Ted Baker
000
001
002
151
152
153
154
253
254
255
Lisa Smith
521-8976
John Smith
521-1234
Sandra Dee
521-9655
Ted Baker
418-4165
Sam Doe
521-5030

Example



- Let's add the following keys to the above array using the hash table:

  29, 16, 14, 99, 127



Comments

- Other array elements may not be used.
- The longer the linked list, the increased search time, is possible O(n).

→ Is there any method of taking advantage of the remaining "empty" space in the array?

Collision Handling

- Chaining approach
- Open-Addressing approach
  - Linear Probing
  - Quadratic Probing
  - Double Hashing

Linear Probing

- Idea:
  - If the current position has been taken, we will find another position in the empty array.
- Linear probing:

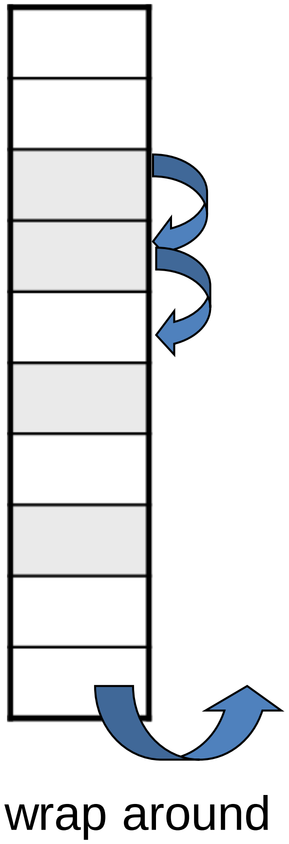
  h(k,i) = (H(k) + i) mod N

  where i: is the order of the attempt (i=0, 1, 2, …)

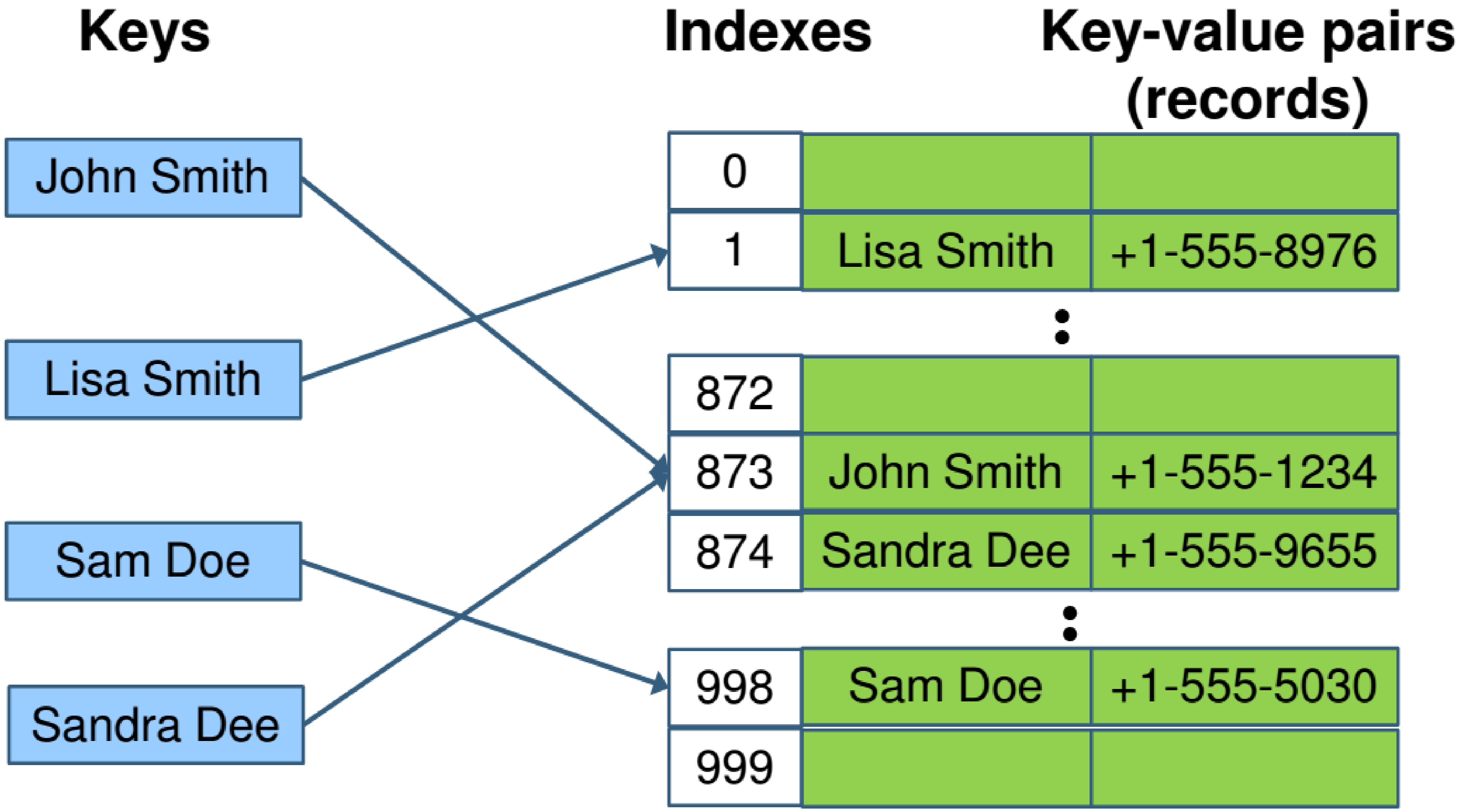
  H(k): hash function

  N: the number of elements in the array.

  Example: H(k), H(k) + 1, H(k) + 2, …



Linear Probing



Example

- Please use the open address method (linear probing) to resolve the following collision:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
		47			35	36			129	25	2501			
					65(?)									

- Answer:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
		47			35	36	65		129	25	2501			
					↑	↑	↑							

attempts

Example

- Please insert the following elements: 29, 16, 14, 99, 127 into the array.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
		47			35	36	65		129	25	2501			

- Answer:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	16	47			35	36	65	127	129	25	2501	29	99	14

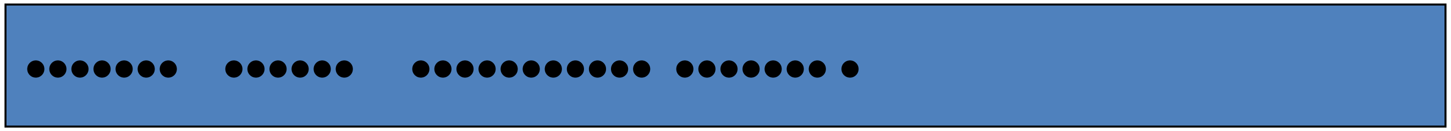
Example

- Hash function: H = k mod 10
- Please perform the following operations:
  - Add 47, 57, 68, 18, 67
  - Search 68
  - Search 10
  - Delete 47
  - Search 57

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Comments

- Advantages:
  - There is no cost of creating a node like the chaining method.
  - Use of free space.
- Disadvantages:
  - The search will cost
  - Deletion will be problematic.
  - Distribution of elements is often in the form of groups, leading to heavy usage and unused space.



Collision Handling

- Chaining approach
- Open-Addressing approach
  - Linear Probing
  - Quadratic Probing
  - Double Hashing

Quadratic Probing

- Similar to linear probing but use the quadratic function to hope the distribution of the elements will be more evenly distributed:

$$h(k,i) = (H(k) + i^2) \bmod N$$

where i: is the order of the attempt (i=0, 1, 2, …)

H(k): hash function

N: number of elements.

Example: $H(k)$, $H(k) + 1^2$, $H(k) + 2^2$, …

Example

- Collision:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
		47			35	36			129	25	2501			
					65(?)									

- Answer:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
		47			35	36			129	25	2501			65
					↑	↑			↑					↑
					t	t+1			t+4					t+9

attempts

Example

- Continue adding the elements: 16, 14, 99, and 127 to the array.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
		47			35	36			129	25	2501			65

- Answer:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
29	16	47	14		35	36	127		129	25	2501		99	65

Comments

- Advantage:
  - Element distribution is more even than linear probing (more evenly spread rather than concentrated in one place, but not too good either).
- Disadvantage:
  - It is possible to fall into the endless loop if precautions are not taken.
    - Example: Try adding key 16 to the array of size 16. Know that positions 0, 1, 4, 9 already contain elements
    - The array size should be prime.

Collision Handling

- Chaining approach
- Open-Addressing approach
  - Linear Probing
  - Quadratic Probing
  - Double Hashing

Double Hashing

- Use two hash functions:
  - The first function to locate the jump to.
  - The second function determines the jump if there is a collision.
- Double hash method:

$$h(k,i) = [H_1(k) + i*H_2(k)] \bmod N$$

where i: is the order of the attempt (i=0, 1, 2, …)

$H_1(k)$: first hash function

$H_2(k)$: second hash function

N: number of elements.

Example: $H(k)$, $H(k) + 1^2$, $H(k) + 2^2$, …

Double Hashing

- Suppose the hash functions are as follows:

  $H_1(k) = k \bmod 13$

  $H_2(k) = 1 + (k \bmod 11)$

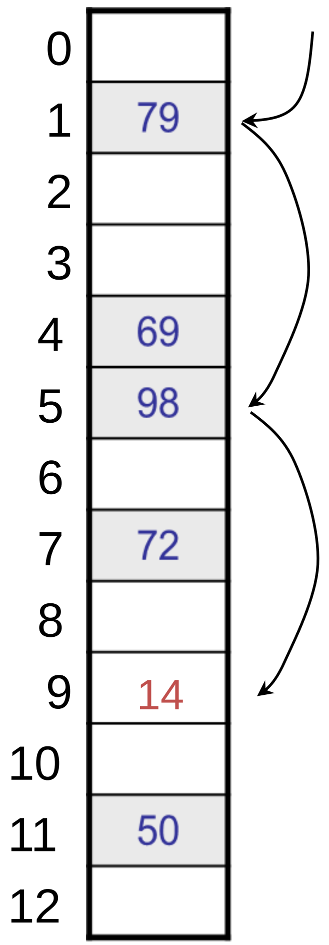
  $h(k,i) = [H_1(k) + i*H_2(k)] \bmod 13$

- Adds element 14 to the array:

  $h_1(14,0) = 14 \bmod 13 = 1$

  $h(14,1) = (h_1(14) + h_2(14)) \bmod 13$
  $= (1 + 4) \bmod 13 = 5$

  $h(14,2) = (h_1(14) + 2\ h_2(14)) \bmod 13$
  $= (1 + 8) \bmod 13 = 9$



The second hash function

- Usually the second hash function has the form :

$$H_2(k) = R + (k \bmod R)$$

where R is prime, $R < N$

Example

- Let the following hash functions:

  $H_1(k) = k \bmod 15$

  $H_2(k) = 11 - (k \bmod 11)$

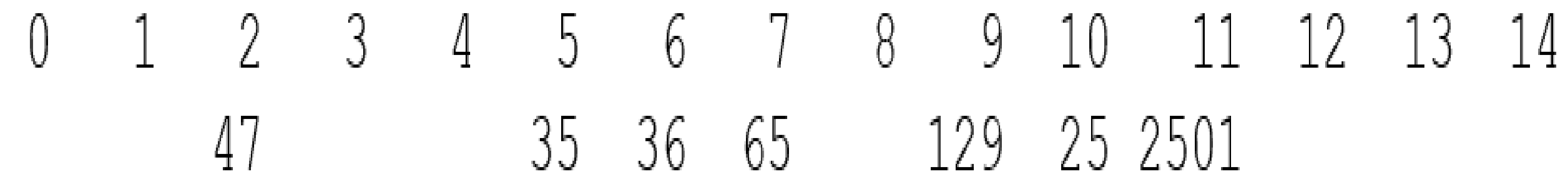
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
		47			35	36			129	25	2501			
					65(?)									

- Answer:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
		47			35	36	65		129	25	2501			
					↑	↑	↑							
					t	t+1	t+2							

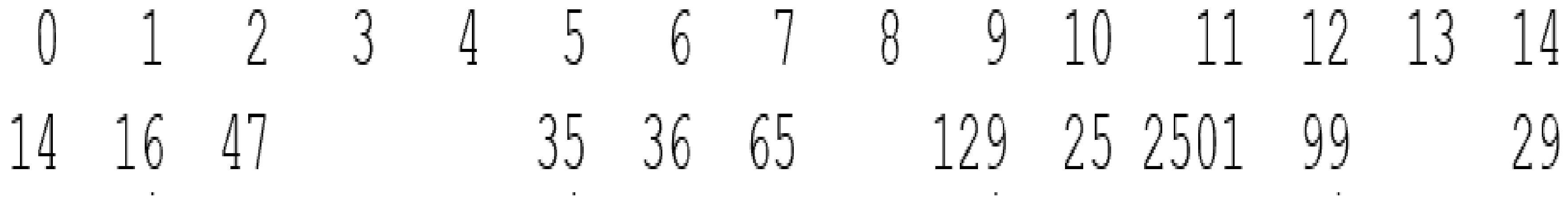
Example

- Let's continue adding the following elements to the array: 16, 14, 99, 127 (!)



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
		47			35	36	65		129	25	2501			

- Answer:



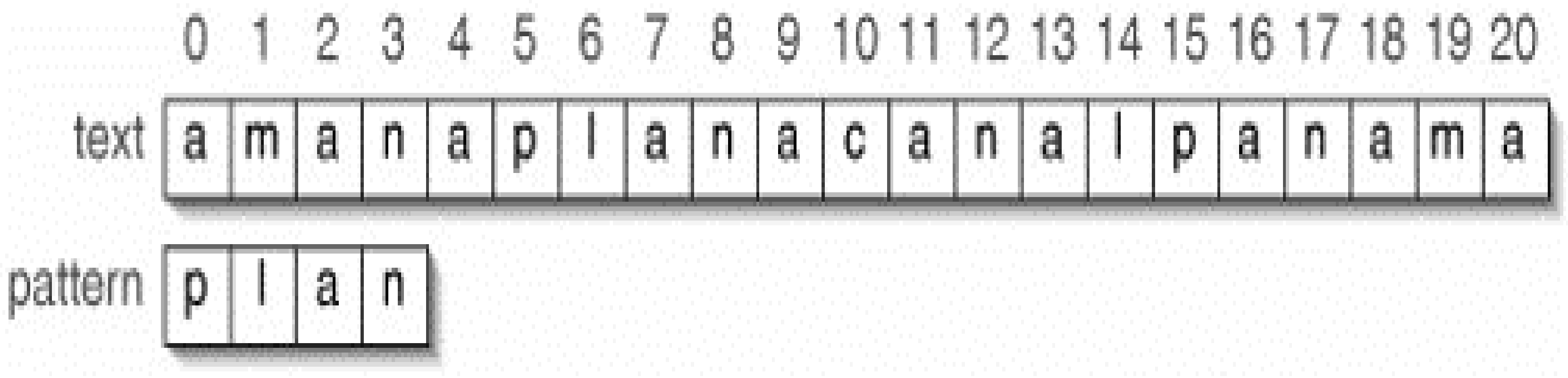
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
14	16	47			35	36	65		129	25	2501	99		29

Element 127: infinite loop

Comments

- Advantages:
  - Avoid focusing multiple elements in one place
  - The positions are defined differently for each element.
- Disadvantages:
  - Execution is slower than quadratic probing because the second hash is computed.

String Match



to be continued...

Advanced topics

- Other search algorithms:
  - Search algorithms on graphs
  - Search with Heuristic
  - Hill climbing
  - …

The End.