Democratic and Popular Republic of Algeria

Ministry of Higher Education and Scientific Research

Ecole supérieure en sciences et technologies de l'informatique et du numérique

Hashing methods

Presented by : Dr. Daoudi Meroua

Academic year: 2024/2025

Introduction

Problem: Storing data in an arbitrary order in an array

- **Maintaining an ordered array:**

In this case, inserting a new data item requires shifting elements to preserve the order.

- **Not maintaining an ordered array:**

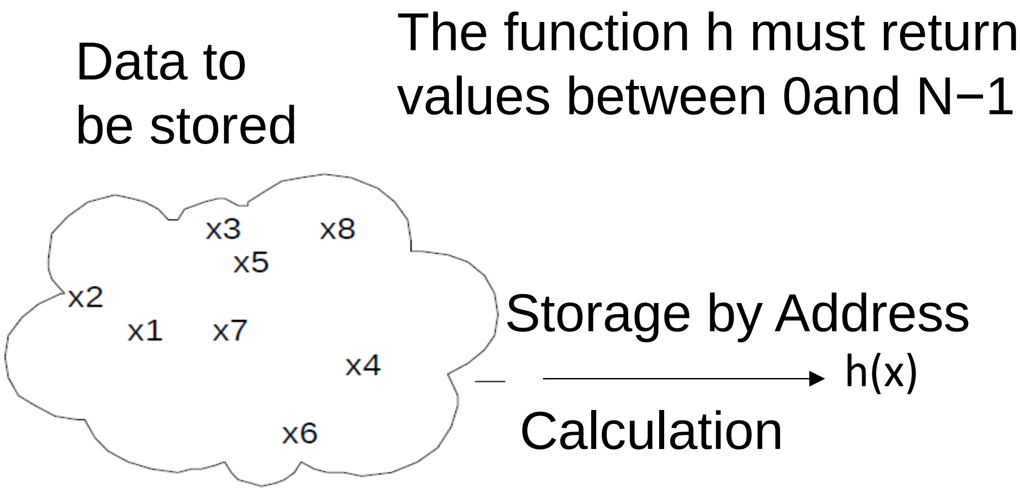
Here, inserting a new data item is fast because it is simply appended to the end.

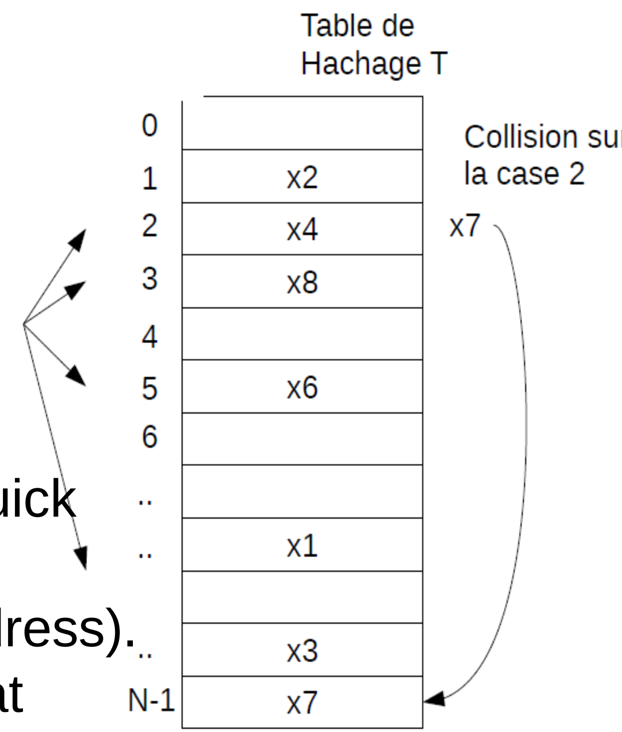
Hashing

A third solution:

- Assign each key xxx a position y in the array, calculated using a hash function h such that y=h(x)
- This is referred to as a scattered storage table or the Hashing technique.
- In this type of storage, whether for inserting or searching for data, the process is always fast (O(1))

Hashing





- Store data (x) in a table (T) using a function (h) for quick localization (address calculation).
- We try to store x in the cell at index h(x) (primary address).
- If the cell is already occupied (collision), we insert x at another location (secondary address) determined by a given algorithm (collision resolution method).
- h(x4)=h(x7)=2 x4 and x7 are synonyms. The primary address of x4 and x7 is 2.
- X7 is inserted in overflow.The secondary address of x7 is N−1

Terminology

- The function hhh is called a hash function.
- The primary address h(x)h(x)h(x) of a data item is the result returned by the hash function.
- Synonyms are data items that have the same primary address.
- Example:
- x1 and x2 are synonyms if h(x1)=h(x2). It is also said that x1 and x2 are in collision.
- Overflow occurs when a data item is not at its primary address. It is also said to be stored at a secondary address.
- The secondary address is determined by a given method. This is referred to as a collision resolution method.

Hashing

Combine: a hash function with a collision resolution method.

- A good property of a hash function:

  minimizes collisions while minimizing the range of the space ⇒ addressable.

- A collision resolution method allows for managing (searching, inserting, and deleting) data that caused collisions. Examples include Linear Probing, External Chaining, Double Hashing, etc.

Hashing function

- The goal is to find a hash function hhh such that $0 \leq h(x) < N$ that minimizes the number of collisions:

- The ideal case is to find a bijective hash function, which means a function that assigns a unique position in the array for each data item to be inserted.

- The worst-case scenario is when all data is hashed to the same address.

- An acceptable solution is one in which some data share the same address (the hash function is not injective).

Hashing function

There are several hash functions, with the most commonly used being:

- The Division Hash Function: Uses the modulo operation to determine the index, typically h(x)=xmod N, where N is the size of the table.
- The Squaring Method: Involves squaring the key and extracting a portion of the result to use as the hash value. This method helps in distributing the hash values more uniformly.
- The Radix Transformation Method: Converts the key into a different base (radix) and uses a portion of the transformed value to determine the index in the hash table.

Hashing function

- In conclusion, there is no universal hash function.
- However, a good hash function should be:
  - Fast to compute
  - Distribute elements uniformly
- It therefore depends on:
  - The machine
  - The elements
- But no function can completely avoid collisions, which will need to be handled.

Collision Resolution Methods

To resolve collisions, two strategies are available:

- Direct methods (hashing by calculating the location):
  - Linear Probing
  - Double Hashing
- Indirect methods or hashing by chaining:
  - Separate Chaining
  - Internal Chaining

Collision Resolution Methods : Linear Probing

1. If a collision occurs at the position h(x), we try the preceding positions: h(x)−1,h(x)−2,…,0,N−1,N−2,… until we find an empty slot.

2. Finding an empty slot indicates that the data does not exist in the table.

3. A free slot must be sacrificed in the hash table to ensure that the probing sequence is finite.

Collision Resolution Methods : Linear Probing

Enregistrement	a	b	c	d	e	f	g	j
h (x)	5	1	3	3	0	2	8	2

Indice	Vide	enregistrement
0	V	
1	V	
2	V	
3	V	
4	V	
5	V	
6	V	
7	V	
8	V	
9	V	

Après l'insertion de a, b et c

Indice	Vide	enregistrement
0	V	
1	F	b
2	V	
3	F	c
4	V	
5	F	a
6	V	
7	V	
8	V	
9	V	

Collision Resolution Methods : Linear Probing

Essai linéaire:

Enregistrement	a	b	c	d	e	f	g	j
h (x)	5	1	3	3	0	2	8	2

Indice	Vide	enregistrement
0	V	
1	F	b
2	F	d
3	F	c
4	V	
5	F	a
6	V	
7	V	
8	V	
9	V	

↑ collision

Calcul de h(d) -1 = 2 ➔ case vide

Collision Resolution Methods : Linear Probing

Essai linéaire:

Enregistrement	a	b	c	d	e	f	g	j
h (x)	5	1	3	3	0	2	8	2

↑ collision

Calcul de h(f) - 1 = 1 ➔ case pleine
Calcul de h(f) - 2 = 0 ➔ case pleine
Calcul de h(f) - 3 + 10 = 9 ➔ case vide

Après l'insertion de e

Indice	Vide	enregistrement
0	F	e
1	F	b
2	F	**d**
3	F	c
4	V	
5	F	a
6	V	
7	V	
8	V	
9	V	

Collision Resolution Methods : Linear Probing

Essai linéaire:

Enregistrement	a	b	c	d	e	f	g	j
h (x)	5	1	3	3	0	2	8	2

↑ collision

Calcul de h(j) - 1 = 1 ➔ case pleine
Calcul de h(j) - 2 = 0 ➔ case pleine
Calcul de h(j) - 3 + 10 = 9 ➔ case pleine
Calcul de h(j) - 4 + 10 = 8 ➔ case pleine
Calcul de h(j) - 5 + 10 = 7 ➔ case vide

Après l'insertion de f, g

Indice	Vide	enregistrement
0	F	e
1	F	b
2	F	**d**
3	F	c
4	V	
5	F	a
6	V	
7	V	
8	F	g
9	F	**f**

Collision Resolution Methods : Linear Probing

Essai linéaire:

Enregistrement	a	b	c	d	e	f	g	j
h (x)	5	1	3	3	0	2	8	2

Après l'insertion de j

Indice	Vide	enregistrement
0	F	e
1	F	b
2	F	**d**
3	F	c
4	V	
5	F	a
6	V	
7	F	**j**
8	F	g
9	F	**f**

Collision Resolution Methods : Linear Probing

Essai linéaire:

Enregistrement	a	b	c	d	e	f	g	j
h (x)	5	1	3	3	0	2	8	2
Adresse	Primaire	P	P	Secondaire	P	S	P	S

- The search for k such that h(k)=2 stops with a failure at the empty slot at index 6 → the test sequence is: 2, 1, 0, 9, 8, 7.
- If we were to insert k, the data would be placed at index 6 (if it is not the last empty slot).
- The table is considered full when the number of inserted elements equals N−1 leading to the sacrifice of one empty slot.

Indice	Vide	
0	F	e
1	F	b
2	F	d
3	F	c
4	V	
5	F	a
6	V	
7	F	j
8	F	g
9	F	f

Linear Probing : search

- The search for a data item proceeds as follows:
- Calculate the primary address of x(let i=h(x))
- If the slot i in the hash table contains the data x, the search is successful.
- Otherwise, search for x in the preceding slots: i−1,i−2,…,0,N−1,N−2, …until an empty slot is found.
- If the search stops at an empty slot, it means that x does not exist in the hash table.

Linear Probing : Insertion

The insertion of a value x proceeds as follows:

- Calculate the primary address of xxx (let i=h(x)i = h(x)i=h(x)).
- If the slot iii is empty, insert xxx into iii and mark the slot as occupied.
- Otherwise, traverse the preceding slots until an empty slot is found (let jjj).
- Insert xxx into the slot jjj and mark it as occupied.

Linear Probing : deletion

- The physical deletion of the data x creates an empty slot.
- This new empty slot can make other slots inaccessible.
- For example, if we want to delete b by clearing slot 1, we also lose access to the data f (where h(f)=2) because it is no longer accessible.
- Tests must be performed before clearing a slot to ensure that other data is not lost.

Indice	Vide	
0	F	e
1	V	b
2	F	d
3	F	c
4	V	
5	F	a
6	V	
7	F	j
8	F	g
9	F	f

Linear Probing : deletion

Indice	Vide	
0	F	e
1	V	b
2	F	d
3	F	c
4	V	
5	F	a
6	V	
7	F	j
8	F	g
9	F	f

Indice	Vide	
0	F	e
1	F	f
2	F	d
3	F	c
4	V	
5	F	a
6	V	
7	V	
8	F	g
9	F	j

Linear Probing : deletion

The principle of deleting the data xxx is as follows:

- Search for the address i of x.
- Traverse all the slots preceding i (let j) until an empty slot is found.
- For each slot j, verify that its data remains accessible if slot i is cleared.
- If all preceding slots remain accessible after clearing i, then clear i and stop.
- If not, move the data from slot j to slot i and attempt to clear its original slot by testing the remaining slots that have not been tested. The same principle is applied for slot i.