13. HASHING

13.1 Giới thiệu về Hashing

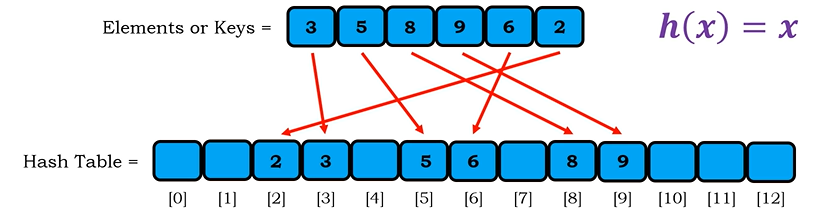
- Hashing là một kĩ thuật dùng cho việc: Tìm kiếm, Chèn, Xóa phần tử từ một Collection

- Độ phức tạp của Hashing: O(1)

13.2 Ý tưởng của Hashing:

- Hash Table dùng để lưu trữ các data hoặc các elements

- Hash Function dùng để truy vấn từng phần tử với vị trí index, trị số tương ứng



- Compression Hashing: Thực hiện giống như Radix Sort

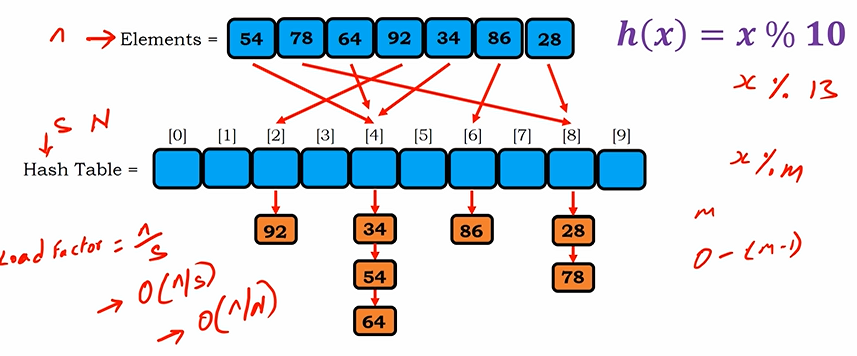
- Collision: Khi mà có nhiều hơn 1 key để truy vấn đến cùng 1 index trong hash table

Bao gồm các kĩ thuật như sau:

* Chaining
* Open Addressing
* Linear Probing
* Quadratic Probing
* Double Hashing

13.2 Chaning

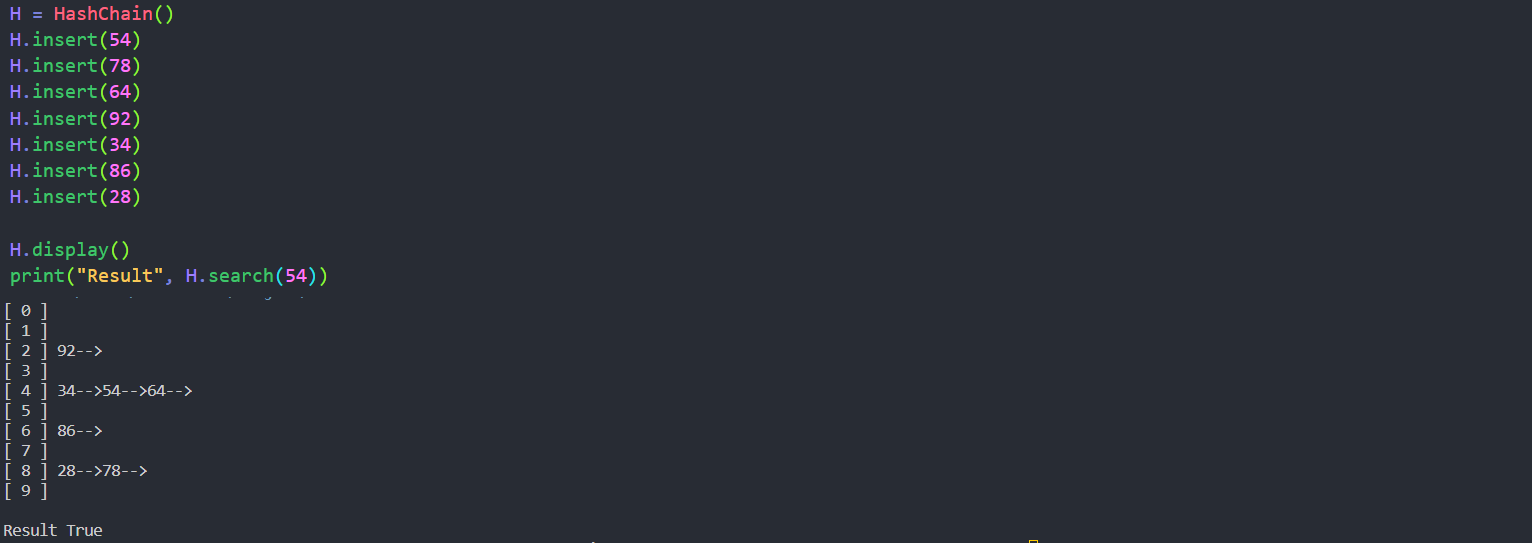
* Chaning dùng cho các auxiliary lists
* Cú pháp đơn giản, dễ hiểu, hiệu quả trong việc handling collisions
* Thực Hiện dựa trên ý tưởng giống với radix sort



Thực thi thuật toán (Python):



Triển Khai và Kết Qủa:



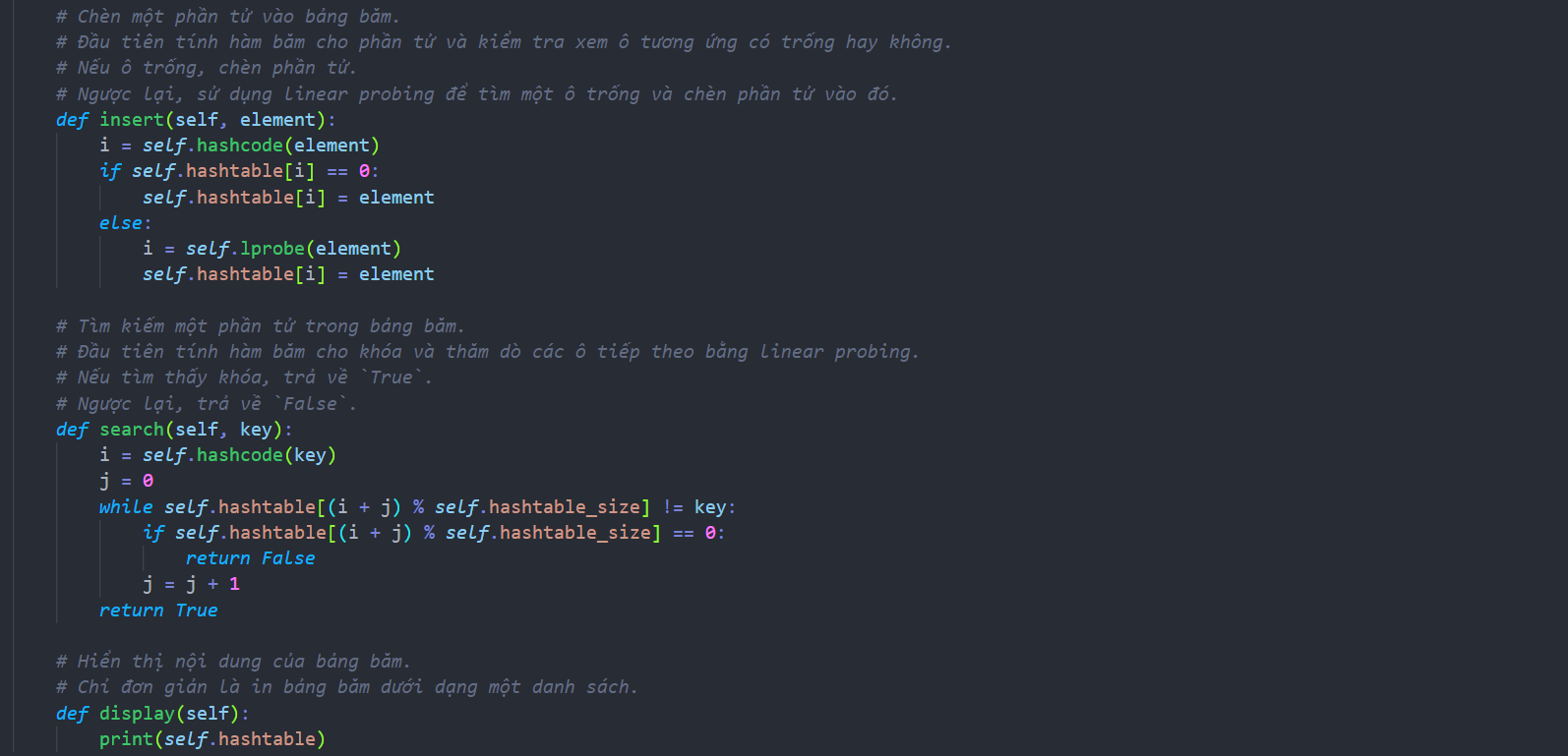
13.4 Linear Probing

Nằm trong phương pháp open addressing

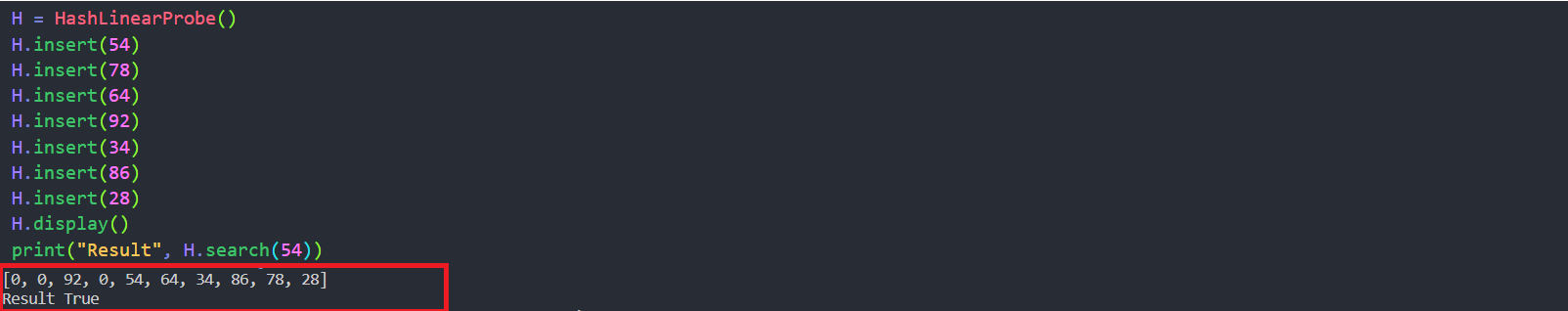
Chèn phần tử vào index kế bên, nếu cell đã thực sự được thực thi

Thực thi thuật toán (Python):





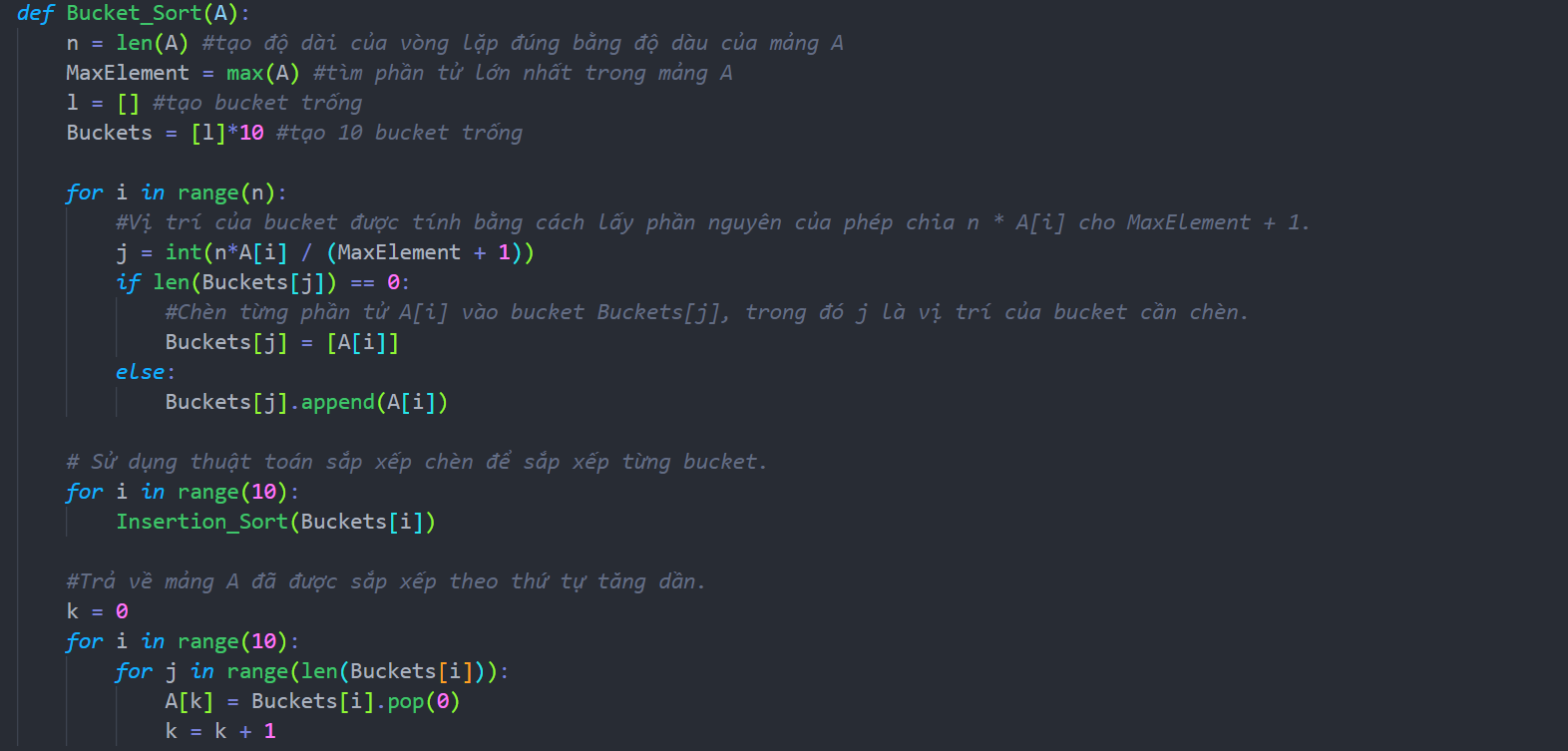
Triển Khai và Kết Qủa:

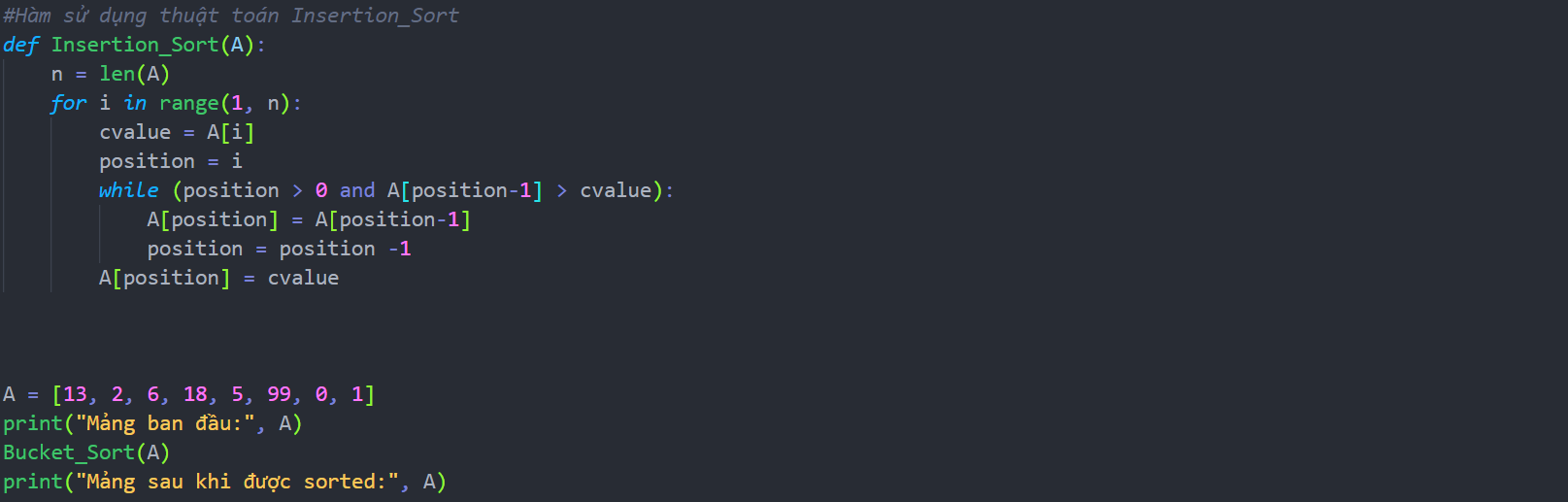


* Quadratic Probing
* Double Hashing

13.5 Bucket Short

* Bucket sort là một kỹ thuật sắp xếp liên quan đến việc chia các phần tử thành nhiều nhóm, hoặc các thùng. Các thùng này được hình thành bằng cách phân phối đều các phần tử. Sau khi các phần tử được chia vào các thùng, chúng có thể được sắp xếp bằng bất kỳ thuật toán sắp xếp nào khác. Cuối cùng, các phần tử đã sắp xếp được thu thập lại theo thứ tự.
* Thuật toán của Bucket Sort:
* Tạo n thùng rỗng (Hoặc danh sách) và thực hiện các bước sau cho mỗi phần tử mảng arr[i].
  + Chèn arr[i] vào bucket[n\*array[i]]
  + Sắp xếp từng thùng riêng lẻ bằng cách sử dụng insertion sort.
  + Ghép nối tất cả các thùng đã sắp xếp.
* Độ Khó thuật toán và Không Gian Bổ Trợ:
* Time Complexity: O(n2): Nếu chúng ta giả sử rằng việc chèn vào một thùng mất O(1) thời gian thì rõ ràng các bước 1 và 2 của thuật toán trên mất O(n) thời gian. O(1) có thể dễ dàng thực hiện nếu chúng ta sử dụng danh sách liên kết để biểu thị một thùng. Bước 4 cũng mất O(n) thời gian vì sẽ có n phần tử trong tất cả các thùng. Bước chính để phân tích là bước 3. Bước này cũng mất O(n) thời gian trung bình nếu tất cả các số được phân bố đều.
* Space Aulixiary: O(n+k)
* Thực Thi Thuật Toán (Python):





Thực Thi và Kết Quả:

