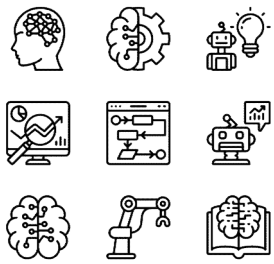
Computer Science for Practicing Engineers

Bảng băm (hash table)



TS. Huỳnh Bá Diệu
Email: dieuhb@gmail.com
Phone: 0914146868

Bảng băm

Nội dung:

1. **Tìm kiếm trên bảng băm**
2. **Các thành phần của bảng băm**
3. **Các loại hàm băm**
4. **Xử lý đụng độ**
5. **Ứng dụng của bảng băm**

Bảng băm (hash table)

Cho biết độ phức tạp của thuật toán khi tìm giá trị trên các cấu trúc dữ liệu dưới đây:

Tìm kiếm trên mảng một chiều: ???

Tìm kiếm trên mảng một chiều có thứ tự:???

Tìm kiếm nội suy trên dãy có thứ tự: ???

Tìm kiếm trên danh sách liên kết đơn: ???

Tìm kiếm trên cây nhị phân: ???

Tìm kiếm trên cây nhị phân tìm kiếm ???

Tìm kiếm trên cây AVL: ???

Bảng băm (hash table)

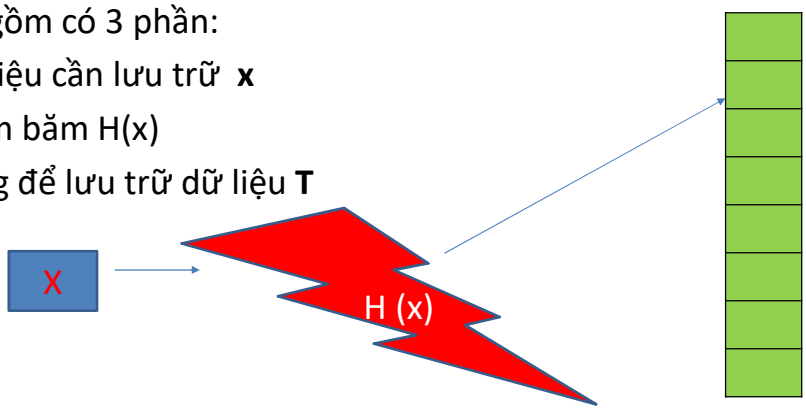
Tìm kiếm trên mảng một chiều:	độ phức tạp là O(n)
Tìm kiếm trên mảng một chiều có thứ tự	độ phức tạp là $O(lg_2 n)$
Tìm kiếm nội suy trên dãy có thứ tự:	độ phức tạp trung bình O(lglgn)
Tìm kiếm trên danh sách liên kết đơn:	độ phức tạp O(n)
Tìm kiếm trên cây nhị phân:	độ phức tạp O(n)
Tìm kiếm trên cây nhị phân tìm kiếm	độ phức tạp O(h), trong đó h là chiều cao
Tìm kiếm trên cây AVL:	độ phức tạp O(h), trong đó h là chiều cao và xấp xĩ bằng O(lgn)

Các thành phần của Bảng băm (hash table)

Bảng băm là cấu trúc lưu trữ với mục đích hỗ trợ cho việc tìm kiếm. Khi tìm kiếm trên bảng băm, độ phức tạp là O(1) hoặc xấp xỉ O(1).

Bảng băm gồm có 3 phần:

- Dữ liệu cần lưu trữ **x**
- Hàm băm H(x)
- Bảng để lưu trữ dữ liệu **T**



Thao tác trên Bảng băm (hash table)

Quá trình chèn một phần tử vào bảng băm:

Một phần tử x khi chèn vào bảng, sẽ được xác định vị trí chèn dựa vào hàm băm, sau đó sẽ được chèn vào vị trí trên bảng.

Quá trình tìm một phần tử x trên bảng băm:

Hàm băm sẽ xác định vị trí phần tử x trên bảng băm, sau đó so sánh giá trị tại vị trí trên bảng với giá trị x để cho kết luận là có hay không có.

Ví dụ tổ chức lưu trữ bằng Bảng băm (hash table)

Giả sử có các giá trị sau: 12, 9, 15, 26, 17, 31, 8, 11, 23

Cần lưu trữ các giá trị này và tìm kiếm liên tục.

- **Dùng mảng 1 chiều có 9 phần tử**
- **Dùng bảng băm**
- **Dùng bảng có 13 ô được đánh số từ 0- 12**
- **Dùng hàm băm hf = x mod 13**

0	1	2	3	4	5	6	7	8	9	10	11	12
26		15		17	31			8	9	23	11	12

Bảng băm (hash table)

Giả sử có các giá trị sau: 12, 9, 15, 26, 17, 31, 8, 11, 23

Cần lưu trữ các giá trị này và tìm kiếm liên tục.

0	1	2	3	4	5	6	7	8	9	10	11	12
26		15		17	31			8	9	23	11	12

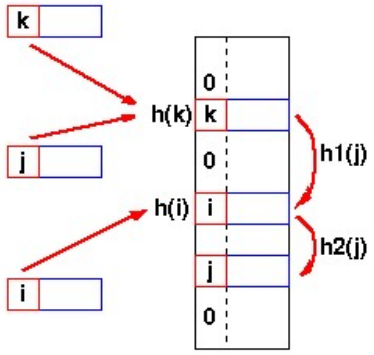
Tìm có x=55 trong bảng không???

Tìm x =9 có trong bảng không???

Đụng độ trong Bảng băm (hash table collision)

Nếu hai giá trị có cùng một vị trí trên bảng băm thì xảy ra hiện tượng đụng độ.

Ví dụ cần thêm giá trị 21 vào trong dãy thì như thế nào???

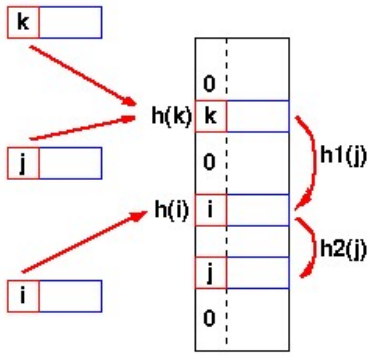


26		15		17	31			8	9	23	11	12	

Đụng độ trong Bảng băm (hash table collision)

Nếu hai giá trị có cùng một vị trí trên bảng băm thì xảy ra hiện tượng đụng độ.

Khi thêm giá trị 21 vào trong dãy thì bị trùng ở vị trí 8, không chèn được.



26		15		17	31			8	9	23	11	12	

Đụng độ trong Bảng băm (hash table collision)

Do hàm băm là không hoàn hảo và không biết trước được dữ liệu được chèn vào như thế nào nên chắc chắn xảy ra hiện tượng đụng độ.

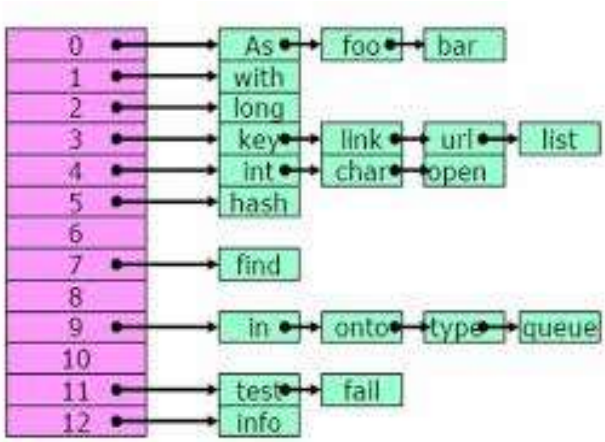
Hiện tượng đụng độ: hai khoá cùng chung một vị trí trên bảng băm.

Các cách xử lý đụng độ (hash table collision)

- Chaining (dùng danh sách liên kết)
- overflow areas (dùng các vùng nhớ phụ)
- re-hashing (sử dụng các hàm băm lại)
- using neighbour slots (linear probing): dùng phương pháp dịch chuyển tuyến tính
- quadratic probing: dùng phương pháp dịch chuyển bình phương
- random probing, ...

Bảng băm (hash table): Chaining List

- Dùng mảng các danh sách liên kết. Nếu hai phần tử có giá trị được xác định bằng hàm băm giống nhau thì cùng 1 danh sách liên kết.
- Nếu hàm băm không tốt, cũng xảy ra hiện tượng các giá trị chỉ tập trung vào 1 danh sách liên kết.



Ví dụ về xử lý đụng độ bằng chaining list

Bảng băm dùng kỹ thuật chaining list để lưu trữ các giá trị: 120, 34, 1, 28, 25, 23 , 45, 12, 54, 9,12, 26, 34,2, 4, 7,32, 21, 99, 27,67, 44, 59, 125, 86, 30, 20, 127,33, 19 .

Hàm băm để xác định vị trí của phần tử x trong bảng băm là:

hf = x mod 17

Bảng gồm 17 phần tử đánh số từ 0 đến 16.

Hãy vẽ hình bảng băm???

Bảng băm (hash table)

x	120	34	1	28	25	23	45	12	54	9	12	26	34	2	4
h(x)	1	0	1	11	8	6	11	12	3	9	12	9	0	2	4

x	7	32	21	99	27	67	44	59	125	86	30	20	127	33	19
H(x)	7	15	4	14	10	16	10	8	6	1	13	3	8	16	2

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	

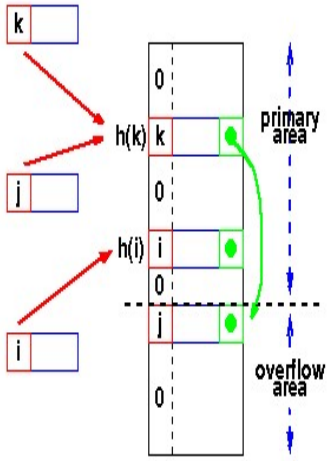
Bảng băm (hash table)

Hãy thử nghiệm tại địa chỉ sau và cho nhận xét???

https://www.cs.usfca.edu/~galles/visualization/OpenHash.html

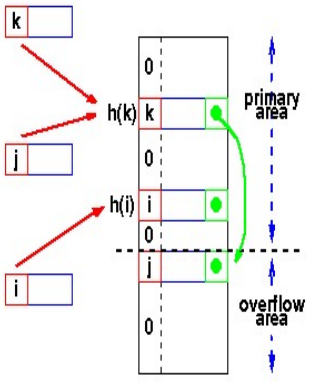
Bảng băm (hash table): **Overflow areas**

Dùng thêm vùng nhớ phụ. Khi một vị trí ô nhớ trong vùng nhớ chính đã có dữ liệu thì nó được chèn vào vị trí trong vùng nhớ phụ.



Bảng băm (hash table): **Overflow areas**

Phương pháp này cũng giống chaining list nhưng có điều khác là vùng nhớ này được cấp phát trước.



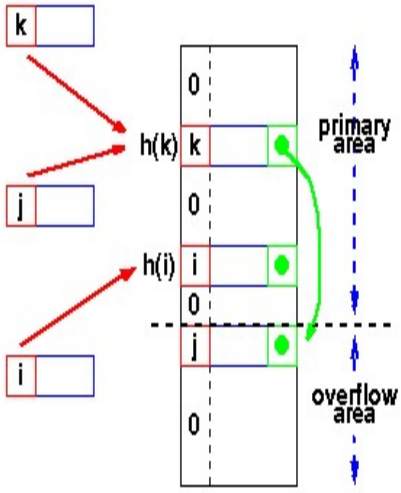
Thảo luận theo nhóm về hạn chế của phương pháp này???

Bảng băm (hash table)

Overflow areas

Ưu điểm: truy cập nhanh

Hạn chế: Tốn nhiều vùng nhớ hơn (nếu như rất ít phần tử lưu ở vùng nhớ phụ)

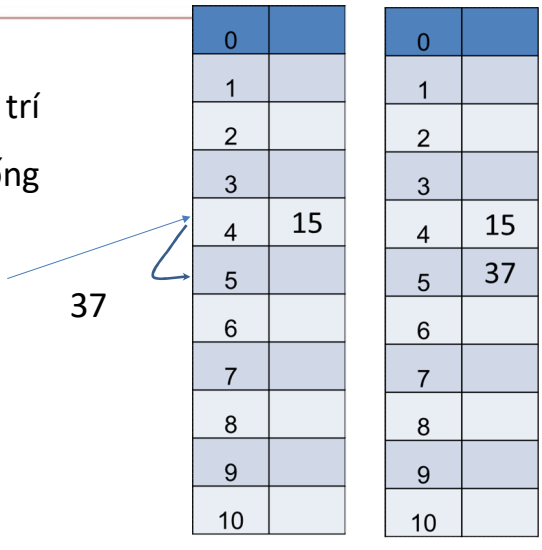


Bảng băm: **Dịch chuyển tuyến tính (**linear probing)

Dùng hàm băm xác định vị trí, nếu vị trí đó có rồi thì kiểm tra ô liền kề có trống không để chèn vào.

Ví dụ hf = x mod 11

Cần chèn 37 vào sau khi đã chèn 15

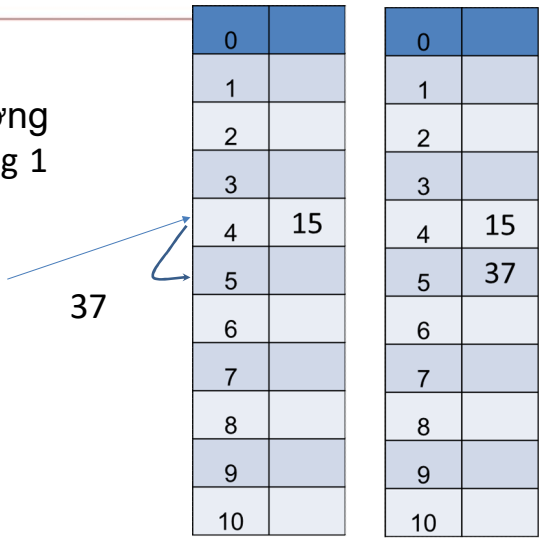


Bảng băm: **Dịch chuyển tuyến tính (**linear probing)

Ưu điểm: nhanh

Nhược điểm: có thể gây hiện tượng tụ tập [nhóm nhiều phần tử vào cùng 1 vị trí]

Ví dụ: chèn thêm 5 →???



Bảng băm (hash table)

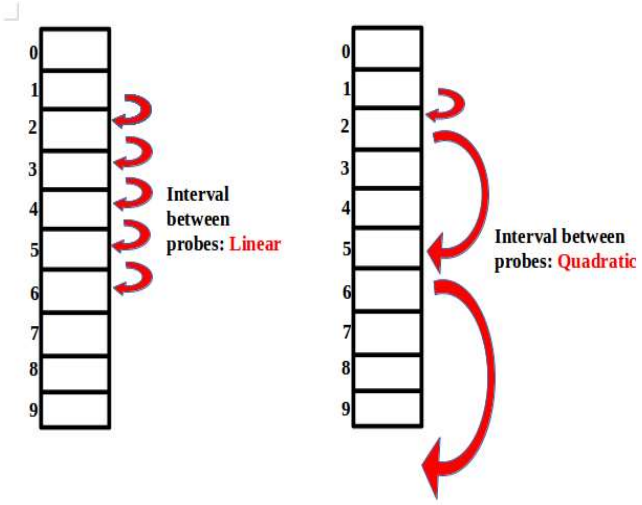
Chia theo nhóm, các bạn thử nghiệm các phương pháp dưới đây, thảo luận và cho nhận xét!!!

https://visualgo.net/en/hashtable?slide=1

http://www.cs.armstrong.edu/liang/animation/web/LinearProbing.html

Bảng băm: **Dịch chuyển bình phương** (quadratic probing)

Thay vì dịch chuyển 1 ô như phương pháp dịch chuyển tuyến tính, phương pháp dịch chuyển bình phương sẽ dịch chuyển i^2 ô, với i là lần thứ lặp lại .

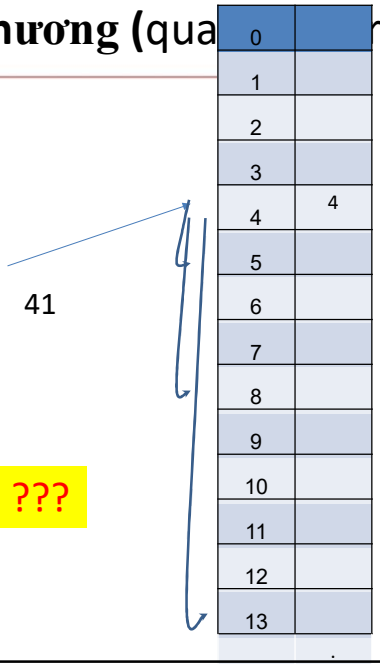


Bảng băm: **Dịch chuyển bình phương** (qua robing)

Ví dụ: Lần thứ nhất xác định hàm băm ở vị trí 4 thì lần sau sẽ chèn vào 5, 8, 13 ..

So với phương pháp dịch chuyển tuyến tính thì phương pháp này ít gây ra hiện tượng tụ tập dữ liệu hơn.

Hãy thử nghiệm: 0, 16, 32, 15, 31 ???

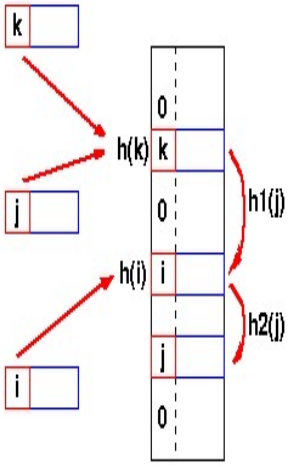


Bảng băm (hash table): **Phương pháp băm lại**

Phương pháp này sẽ dùng hai (hoặc nhiều hơn) hàm băm để xác định vị trí của một phần tử trong bảng băm.

Lần thứ nhất sẽ dùng hàm băm h1 để xác định. Nếu vị trí chèn đã có thì sẽ dùng hàm hăm h2 để xác định.

Hình bên dùng 3 hàm băm, h, h1, h2



Bài tập dung Bảng băm 1

Hash Table Ht have *tablesize size = 8* is used to store string. We use chaining with list heads to manage collision. The hash function to locate a word in the hash table is given as follows:

```
int hf( String t) {
    int s=0;
    for(int i=0; i<t.length(); i++) s=s + (int) t.charAt(i);
    return s % tablesize;
}
```

The hash table will use to store all words in string S="chao mung ngay giai phong hoan toan mien nam".

Bài tập dung Bảng băm 1 (tt)

Map the position of each word in string using hash function above.

Draw the hash table if we insert into the hash table all words in string S, one by one from left to right.

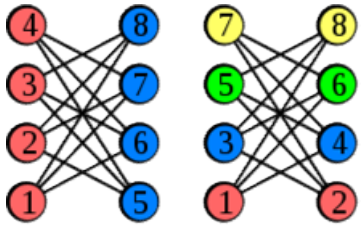
97	98	99	100	101	102	103	104	105	106	107	108	109
a	b	c	d	e	f	g	h	i	j	k	l	m

110	111	112	113	114	115	116	117	118	119	120	121	122
n	o	p	q	r	s	t	u	v	w	x	y	z

Bài tập nhóm

Mỗi nhóm 5 người, chọn 1 trong 2 bài sau (sử dụng thuật toán tham):

1. Có n thành phố và giữa một số các thành phố có tuyến bay nối với nhau, kèm theo chi phí chuyến bay. Một người từ thành phố A muốn đi qua tất cả các thành phố sau đó quay lại nơi xuất phát sao cho chi phí là ít nhất có thể. Hãy cài đặt thuật toán tìm đường.
2. Cho đồ thị có n đỉnh. Hãy tô màu đồ thị sao cho hai đỉnh kề nhau có màu khác nhau.



Bài tập Bảng băm (hash table)

Cho file HASH.inp chứa các số nguyên.

Hãy chèn các phần tử trong file vào bảng băm(dùng kỹ thuật Chaining List để tránh đụng độ). Kích thước bảng băm là 113.

Hàm băm sử dụng phương pháp Folding

https://www.cise.ufl.edu/~sahni/dsaaj/enrich/c11/function.htm#Folding

In ra file HASH.out thông tin bảng băm gồm 113 dòng, mỗi dòng chứa các phần tử trong bảng ở vị trí thứ i (i=0 đến 112).

Các bạn đưa ra giải pháp của các bạn???

Bảng băm (hash table)

```
import java.util.LinkedList;
public class CSPE_MyHashTable1 {
   LinkedList []T;
   final int size=113;
   CSPE_MyHashTable1 ()    {
      T= new LinkedList[size];
      for(int i=0; i<size; i++)          T[i] = new LinkedList <>();
   }
   int hf(int x) {return x% size;}
 }
```

Bảng băm (hash table)

```
void insert(int x)   {
        int vt= hf(x);
        if (T[vt].contains(x)==false) T[vt].add(x);
}
void xuly(){
         // mở file ra
         // đọc lần lượt từng phần tử trong file HASH.inp và chèn vào bảng
         // duyệt bang băm và ghi lại ra file HASH.out
}
```

Phương pháp thiết kế hàm băm (hash table)

Phương pháp thiết kế hàm băm

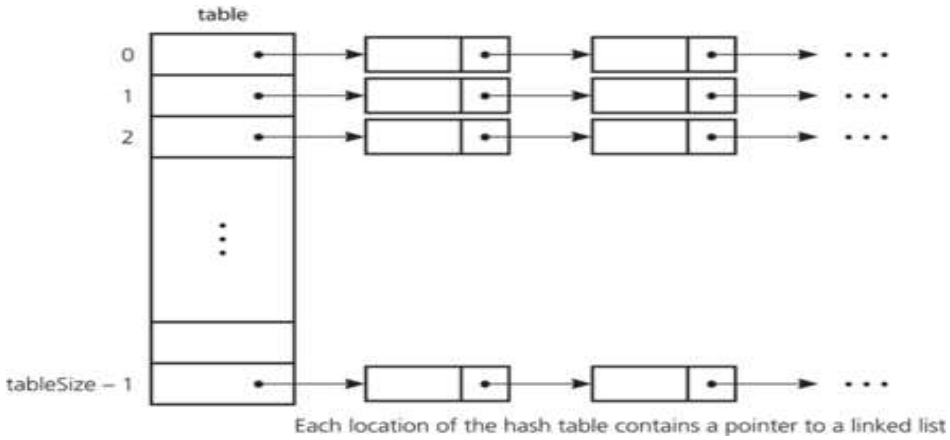
- Dùng hàm mod
- Dùng phương pháp nhóm (nhóm vào thùng: binning)
- Phương pháp **Mid-Square :**

http://research.cs.vt.edu/AVresearch/hashing/midsquare.php

Xử lý đụng độ trên bảng băm

- Chaining (dùng danh sách liên kết)
- overflow areas (dùng các vùng nhớ phụ)
- re-hashing (sử dụng các hàm băm lại)
- using neighbouring slots (linear probing): dùng phương pháp dịch chuyển tuyến tính
- quadratic probing: dùng phương pháp dịch chuyển bình phương
- random probing, ...

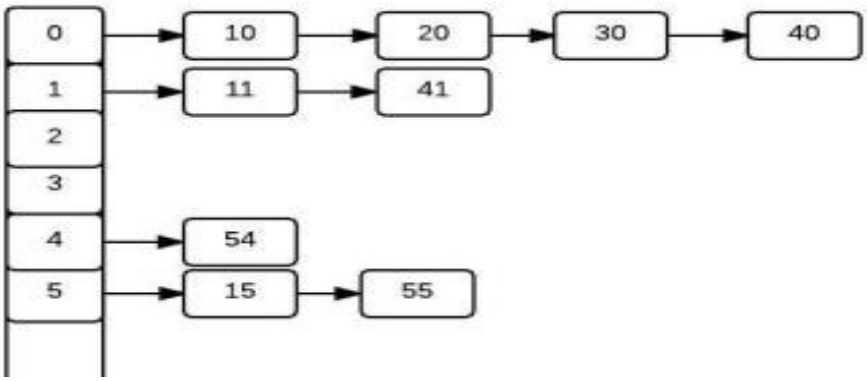
Xử lý đụng độ trên bảng băm: **Chaining (danh sách liên kết)**



Each location of the hash table contains a pointer to a linked list

Xử lý đụng độ trên bảng băm

Chaining (dùng danh sách liên kết)



Xử lý đụng độ trên bảng băm

- Chaining (dùng danh sách liên kết)
- overflow areas (dùng các vùng nhớ phụ)
- re-hashing (sử dụng các hàm băm lại)
- using neighbouring slots (linear probing): dùng phương pháp dịch chuyển tuyến tính
- quadratic probing: dùng phương pháp dịch chuyển bình phương
- random probing, ...

Sử dụng bảng băm của java

```
import java.util.*;
public class HashTableDemo {
  public static void main(String args[]) {
    // Tao bang
    Hashtable balance = new Hashtable();
    Enumeration names;
    String str;
    double bal;
    balance.put(“An", new Double(3434.34));
    balance.put(“Binh", new Double(123.22));
    balance.put(“Cong", new Double(1378.00));
    balance.put("Dung", new Double(99.22));
    balance.put(“Hanh", new Double(-19.08));
```

Sử dụng bảng băm của java

```
import java.util.*;
public class HashTableDemo {
  public static void main(String args[]) {
    // Tao bang

    // Hien thi noi dung trong hash table.
    names = balance.keys();
    while(names.hasMoreElements()) {
      str = (String) names.nextElement();
      System.out.println(str + ": " +
balance.get(str));
    }
    System.out.println();
    // tang tai khoan cho Binh
    bal =
((Double)balance.get(“Binh")).doubleValue();
    balance.put(“Binh", new Double(bal +
1000));
    System.out.println(“Tai khoan Binh : " +
balance.get(“Binh"));
  }
}
```

Bài toán xử lý bằng bảng băm: **Perfect Pair**

Rajiv and Nitish had a fight because Rajiv was annoying Nitish with his question. Rajiv being a genius in arrays gave Nitish an array of natural numbers ***A*** of length ***N*** with elements $A_1, A_2, \ldots, A_N$. Nitish has to find the total amount of perfect pairs in the array.

A perfect pair (A_i, A_j) is a pair where $(A_i + A_j)$ is a perfect square or a perfect cube and $i \neq j$.

Bài toán xử lý bằng bảng băm: **Perfect Pair**

Since Rajiv and Nitish are not talking with each other after the fight you have been given the question to solve and inturn make both of them a perfect pair again.

NOTE :- A pair (A_i, A_j) and (A_j, A_i) are same and not to be counted twice.

Bài toán xử lý bằng bảng băm: **Perfect Pair**

Input

The first line on the input contains the a single integer ***T*** denoting the number of test cases. The first line of each test case contains a single integer ***N***. The second line contains ***N*** space-separated integers $A_1, A_2, \ldots, A_N$.

Output

For each test case, print a single line containing a single integer denoting the total number of perfect pairs.

Constraints

$1 \le T \le 10$
$1 \le N \le 10^5$
$1 \le A_i \le 10^3$

Bài toán xử lý bằng bảng băm: **Perfect Pair**

SAMPLE INPUT

2
5
1 2 3 4 5
4
1 4 5 8

SAMPLE OUTPUT

3
2

Giải pháp của các bạn???

Tài liệu đọc thêm về bảng băm

https://en.wikipedia.org/wiki/Hash_table

https://www.cs.auckland.ac.nz/software/AlgAnim/hash_tables.html

Link YouTube

https://www.youtube.com/watch?v=MfhjkfocRR0

https://www.youtube.com/watch?v=0M_kIqhwbFo