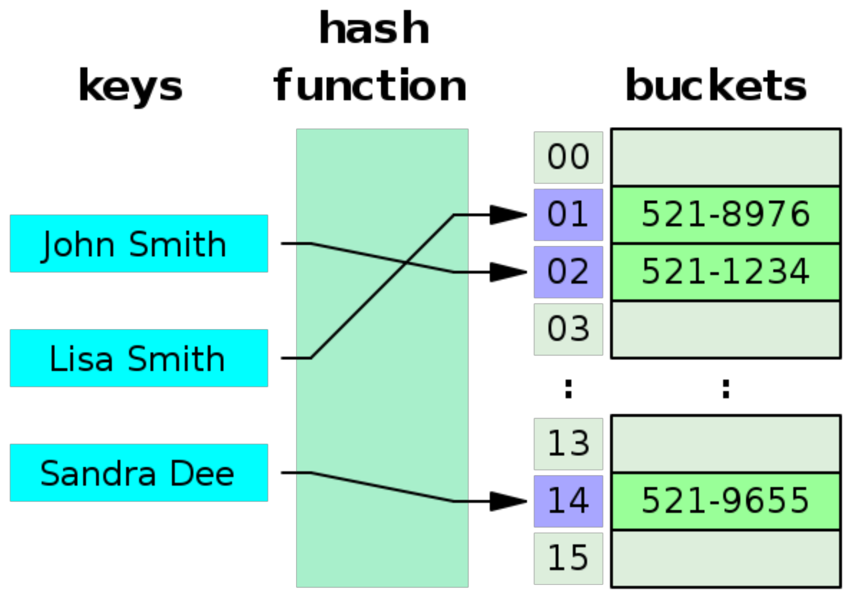
Lecture 5: Hash Table

Dr. Tsung-Wei Huang
Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, UT

Hash Table

- A data structure that maintains a mapping between "keys" and "values"
  - A hash function to compute the key (index of array)
  - A structure that maps the associated value



Why do we Need Hash Table?

- **Main reason is memory storage efficiency**
  - N numbers within the range 1~N
    - easy to declare an array
  - N numbers within the rage 1~2147483647
    - uneasy to declare an array
  - N numbers within the range -65536.0 ~ 65536.0
    - uneasy to declare an array
- **We uses hash table to “discretize” items**
  - So we can easily store them in a linear storage

std::map and std::unordered_map

- **Both are 1-1 mapping**
  - map implements O(logN) red-black tree
  - unordered_map implements O(1) hash function

```
#include <map>
#include <unordered_map>

//declaration
std::map<int, int>QQ; // std::unordered_map<int, int>
QQ[1234567] = 123456;
QQ[8] = -1;
printf("%d\n", QQ[8]);          //-1
printf("%d\n", QQ[123456]);     //0

//clear
QQ.clear();
```

std::set and std::unordered_set

- **Set is a collection of unique items**
  - std::set implements O(logN) red-black tree
  - std::unordered_set implements O(1) hash function

```
#include <set>
#include <unordered_set>
std::set<int>my; // or std::unordered_set<int>

my.insert(1);
my.insert(5);
my.insert(3);

for(auto data : my)
    cout<< data <<" "<<endl; //1, 3, 5

my.clear();
```

Example 1: Dictionary Set

You are asked to write a program that lists all the different words in the input text. In this problem, a word is defined as a consecutive sequence of alphabets, in upper and/or lower case. Words with only one letter are also to be considered. Furthermore, your program must be CaSe InSeNsItIvE. For example, words like "Apple", "apple" or "APPLE" must be considered the same.

Input: The input file is a text with no more than 5000 lines. An input line has at most 200 characters. Input is terminated by EOF.

Output: Your output should give a list of different words that appears in the input text, one in a line. The words should all be in lower case, sorted in alphabetical order. You can be sure that he number of distinct words in the text does not exceed 5000.

Example 1: Dictionary Set

Sample Input

Adventures in Disneyland Two blondes were going to Disneyland when they came to a fork in the road. The sign read: "Disneyland Left." So they went home.

Sample Output

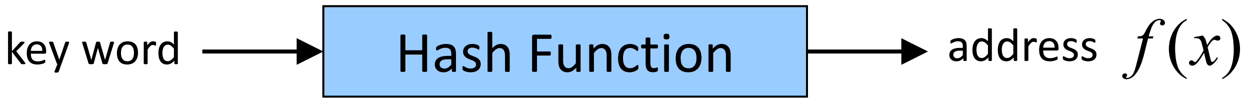
```
a
adventures
blondes
came
disneyland
fork
going
home
in
left
read
road
sign
so
the
…
```

Please refer to CPP reference

- https://en.cppreference.com/w/cpp/container/set
- https://en.cppreference.com/w/cpp/container/map
- https://en.cppreference.com/w/cpp/container/unordered_set
- https://en.cppreference.com/w/cpp/container/unordered_map

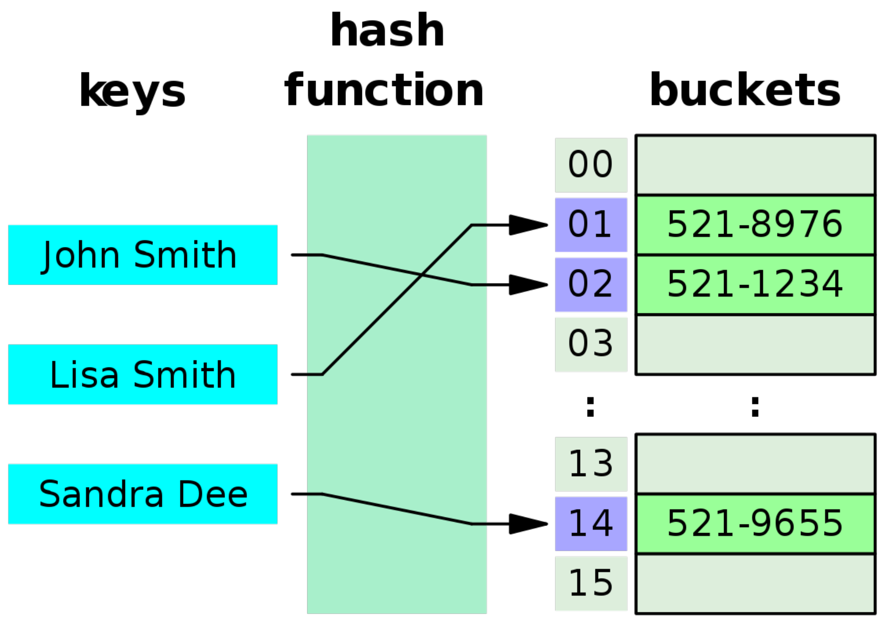
Hash Table

- **Hash**
  - A table with static or dynamic size
  - hash represents the procedure that uses a function f(x) to transform a key word x into a new address



Hash Table Terminology

- Bucket: The size of f(x) for the hash table
- Slot: The size of each bucket
- Collision: x_1 != x_2, while $f(x_1)=f(x_2)$
- Overflow: The size of bucket is full



Hash Function

- ❑ A hash function is a well-defined procedure or mathematical function which converts a large, possibly variable-sized amount of data into a small datum, usually a single integer that may serve as an index to an array. The values returned by a hash function are called hash values, hash codes, hash sums, or simply hashes.
  - ❑ CFGA → 3671 (turn to a number) → 13476241 (square the number) → 762 (take the mid 3 numbers)

Shift Folding Hash Function

- Shift folding
- Ex.

  key word X

  X=167329421598 12

  group for each three nubmers

P1	167
P2	329
P3	421
P4	598
P5	12

summation 1527

Shift Folding and Reverse Hash Function

- Shift folding
- Ex.

  key word X

  X=16732942159812

  group for each three numbers

P1	167	
P2	~~329~~	← reverse 923
P3	421	
P4	~~598~~	← reverse 895
P5	12	

summation $\boxed{2418}$

Modulo Hash

- **Example of Hash Function (Division)**
  - Divide the key word by a value M and use the modulo value as the address

$$f(x) = x\%M$$

Modulo Hash Example

- push **345, 728, 251, 490, 15** into 12 buckets, M is 12, **f(x)=x%12**

  f(345)=9

  f(728)=8

  f(251)=11

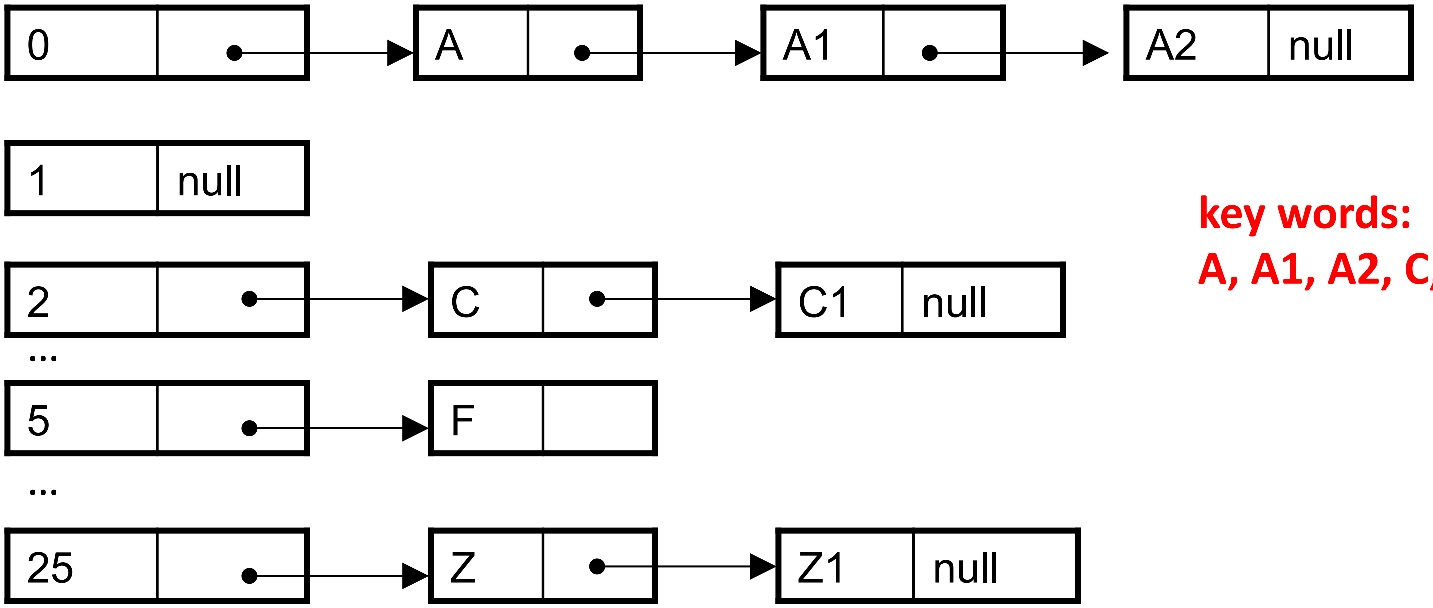
  f(490)=10

  f(15)=3

	X		X
0		6	
1		7	
2		8	728
3	15	9	345
4		10	490
5		11	251

Handling Overflow

- ❑ The size of bucket is overflow due to multiple key words that access the same bucket
- ❑ Maintains a std::list or a std::vector to store multiple items that map to the same key

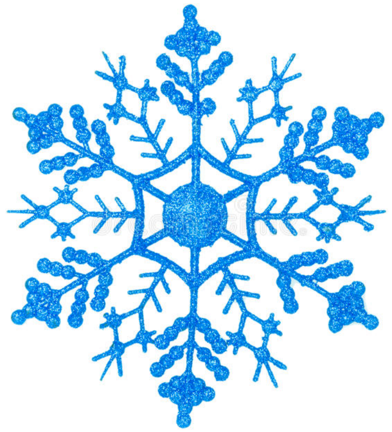


key words:
A, A1, A2, C, C1, F, Z, Z1

Example 2 - Snowflake

Problem Description

You may have heard that no two snowflakes are alike. Your task is to write a program to determine whether this is really true. Your program will read information about a collection of snowflakes, and search for a pair that may be identical. Each snowflake has six arms. For each snowflake, your program will be provided with a measurement of the length of each of the six arms. Any pair of snowflakes which have the same lengths of corresponding arms should be flagged by your program as possibly identical.



Example 2: Snowflake

I/O Description

Input

The first line of input will contain a single integer *n*, $0 < n \leq 100000$, the number of snowflakes to follow. This will be followed by *n* lines, each describing a snowflake. Each snowflake will be described by a line containing six integers (each integer is at least 0 and less than 10000000), the lengths of the arms of the snow ake. The lengths of the arms will be given in order around the snowflake (either clockwise or counterclockwise), but they may begin with any of the six arms. For example, the same snowflake could be described as 1 2 3 4 5 6 or 4 3 2 1 6 5.

Output

If all of the snowflakes are distinct, your program should print the message:

No two snowflakes are alike.

If there is a pair of possibly identical snow akes, your program should print the message:

Twin snowflakes found.

Example 2: Snowflake

Sample I/O

Input

```
5
1 2 3 4 5 6
5 6 1 2 3 4
6 5 4 3 1 2
1 4 6 2 3 5
4 3 2 1 6 5
```

Output

```
Twin snowflakes found.
```

Brute Force?

- **Enumerate all pairs**
  - For each pair check if both are identical for
    - Six clockwise directions
    - Six counter-clockwise directions
- **Time complexity**
  - O(N*N*12*6)

Symmetry

Six rotations					
1	2	3	4	5	6
6	1	2	3	4	5
5	6	1	2	3	4
4	5	6	1	2	3
3	4	5	6	1	2
2	3	4	5	6	1

How do we Decide a Hash Function?

Six rotations					
1	2	3	4	5	6
6	1	2	3	4	5
5	6	1	2	3	4
4	5	6	1	2	3
3	4	5	6	1	2
2	3	4	5	6	1

Example 3: Balanced Binary Code

Problem Description

John’s *N* items (1 ≤ *N* ≤ 100,000) share many similarities. In fact, John has been able to narrow down the list of features shared by these items to a list of only *K* different features (1 ≤ *K* ≤ 30). For example, items exhibiting feature #1 might have red color, items exhibiting feature #2 might have black color, and so on.

John has devised a concise way to describe each item in terms of its "feature ID", a single K-bit integer whose binary representation tells us the set of features exhibited by the item. As an example, suppose an item has feature ID = 13. Since 13 written in binary is 1101, this means our item exhibits features 1, 3, and 4 (reading right to left), but not feature 2. More generally, we find a 1 in the 2^(*i*-1) place if an item exhibits feature *i*.

Items are lined up in order 1..*N* along a long row. Certain ranges of items are somewhat "balanced" in terms of the features the exhibit. A contiguous range of items *i*..*j* is balanced if each of the *K* possible features is exhibited by the same number of items in the range. We need to find the size of the largest balanced range of items.

Example 3: Balanced Binary Code

I/O Description

Input

Line 1: Two space-separated integers, *N* and *K*.

Lines 2..*N*+1: Line *i*+1 contains a single *K*-bit integer specifying the features present in item *i*. The least-significant bit of this integer is 1 if the cow exhibits feature #1, and the most-significant bit is 1 if the item exhibits feature #*K*.

Output

Line 1: A single integer giving the size of the largest contiguous balanced group of cows.

Example 3: Balanced Binary Code

Input

```
7 3
7
6
7
2
1
4
2
```

Output

```
4
```

Brute Force?

- **Enumerate all pairs of ranges**
    - Check if the range is balanced
- **Time complexity**
    - O(N*N*k)

Clever Method

Binary numbers

7	111
6	110
7	111
2	010
1	001
4	100
2	010

Clever Method

Binary numbers

7	111
6	110
7	111
2	010
1	001
4	100
2	010

Lump sum

111	111
110	221
111	332
010	342
001	343
100	443
010	453

Clever Method

Binary numbers

7	111
6	110
7	111
2	010
1	001
4	100
2	010

Lump sum

111	111
110	221
111	332
010	342
001	343
100	443
010	453

Offset

111	000
110	110
111	110
010	120
001	010
100	110
010	120

Clever Method

Offset

0	7	111	000
1	6	110	110
2	7	111	110
3	2	010	120
4	1	001	010
5	4	100	110
6	2	010	120

Clever Method

Offset

0	7	111	000
1	6	110	110
2	7	111	110
3	2	010	120
4	1	001	010
5	4	100	110
6	2	010	120

000	0	7,	0
110	1	6, 7, 4	1, 2, 5
120	2	2, 2	3, 6
010	3	1	4