GeekBand 极客班

互联网人才 + 油站!

GeekBand 极客班

www.geekband.com

1. Array & String

大纲

1. 入门题看string match
2. Array中HashTable应用
3. C/C++中的string
4. 例题分析

看一道入门题

/* Returns the position of the first occurrence of string target in string source, or -1 if target is not part of source.*/

```
int strStr(String source, String target) {
        //…
}
```

字符串匹配

两种比较易于实现的字符串比较算法

假设在长度为n的母串中匹配长度为m的子串。
（http://en.wikipedia.org/wiki/String_searching_algorithm）。

Brute-Force算法： 顺序遍历母串，将每个字符作为匹配的起始字符，判断是否匹配子串。时间复杂度 O(m*n)

Brute-Force

```
1  char* StrStr(const char *str, const char *target) {
2    if (!*target) return str;
3    char *p1 = (char*)str;
4    while (*p1) {
5      char *p1Begin = p1, *p2 = (char*)target;
6      while (*p1 && *p2 && *p1 == *p2) {
7        p1++;
8        p2++;
9      }
10     if (!*p2)
11       return p1Begin;
12     p1 = p1Begin + 1;
13   }
14   return NULL;
15 }
```

Rabin-Karp

将每一个匹配子串映射为一个hash值。例如，将子串看做一个多进制数，比较它的值与母串中相同长度子串的hash值，如果相同，再细致地按字符确认字符串是否确实相同。顺序计算母串hash值的过程中，使用增量计算的方法：扣除最高位的hash值，增加最低位的hash值。因此能在平均情况下做到O(m+n)。

raw: “1234”

match: “23”

```
int RabinKarp(string s[1..n], string pattern[1..m])
    hpattern = hash(pattern[1..m]);  hs = hash(s[1..m]);
    for i from 1 to n-m+1
        if hs = hpattern
            if s[i..i+m-1] = pattern[1..m]
                return i
        hs = hash(s[i+1..i+m])
       return not found
```

Array

int array[arraySize]; //在Stack上定义长度为arraySize的整型数组

int *array = new int[arraySize]; //在Heap上定义长度为arraySize的整型数组
使用完后需要释放内存:
delete[] array;

注意，在旧的编译器中，不能在Stack上定义一个长度不确定的数组，即只能定义如下:
int array[10];
新的编译器没有如上限制。但是如果数组长度不定，则不能初始化数组:
int array[arraySize] = {0}; //把不定长度的数组初始化为零，编译报错。

工具箱：Stack Vs. Heap

Stack主要是指由操作系统自动管理的内存空间。当进入一个函数，操作系统会为该函数中的局部变量分配储存空间。事实上，系统会分配一个内存块，叠加在当前的stack上，并且利用指针指向前一个内存块的地址。

函数的局部变量就存储在当前的内存块上。当该函数返回时，系统“弹出”内存块，并且根据指针回到前一个内存块。所以，Stack总是以后进先出(LIFO)的方式工作

Heap是用来储存动态分配变量的空间。对于heap而言，并没有像stack那样后进先出的规则，程序员可以选择随时分配或回收内存。这就意味着需要程序员自己用命令回收内存，否则会产生内存泄露(memory leak)。

在C/C++中，程序员需要调用free/delete来释放动态分配的内存。在JAVA，Objective-C (with Automatic Reference Count)中，语言本身引入垃圾回收和计数规则帮助用户决定在什么时候自动释放内存。

二维数组

在Stack上创建：
int array[M][N];
传递给子函数：
void func(int arr[M][N]){ // M可以省略，但N必须存在，以便编译器确定移动内存的间距

在Heap上创建：
int **array = new int*[M]; // 或者 (int**)malloc(M * sizeof(int*));
for(int i = 0; i < M; i++)
    array [i] = new int[N]; // 或者 (int*)malloc(N * sizeof(int));
传递给子函数：
void func(int **arr, int M, int N){}
使用完后需要释放内存：
for(int i = 0; i < M; i++)
    delete[] array[i];
delete[] array;

工具箱：Vector

vector可以用运算符[]直接访问元素。请参考http://www.cplusplus.com/reference/vector/vector/

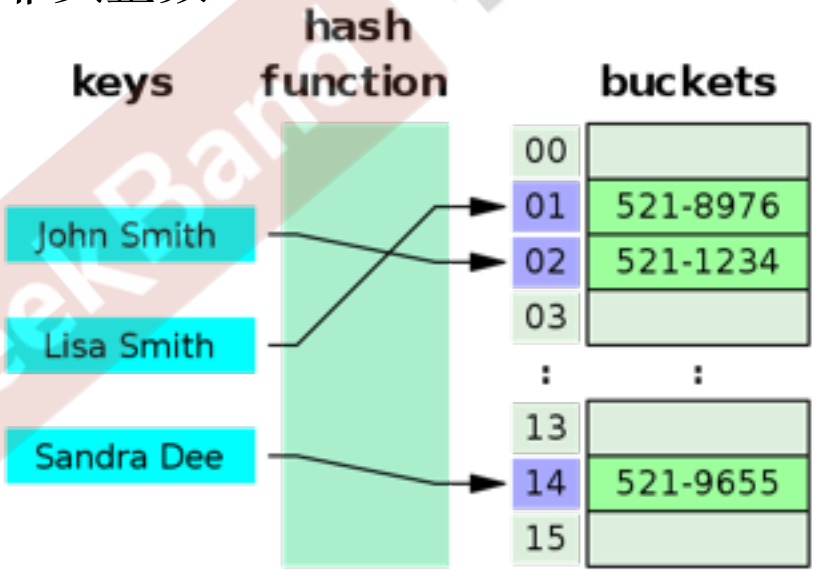
```
size_type size() const;       // Returns the number of elements in the vector.
void push_back (const value_type& val);
void pop_back();

iterator erase (iterator first, iterator last); // Removes from the vector either a single
element (position) or a range of elements ([first,last)).

for (vector<int>::iterator it = v.begin(); it != v.end(); ) {
        if (condition) {
            it = v.erase(it);
        } else {
            ++it;
        }
}
```

Hash Table

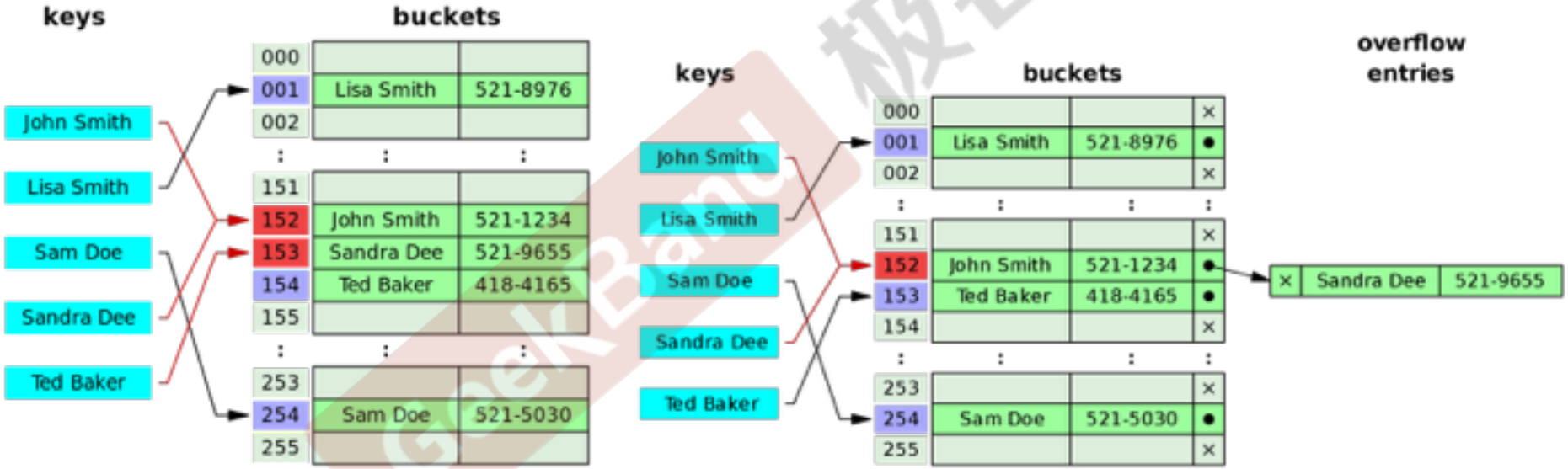
Hash table几乎是最为重要的数据结构，主要用于基于“key”的查找，存储的基本元素是key-value的pair。逻辑上，数组可以作为Hash table 的一个特例：key是一个非负整数。

Operations

- Insert - O(1)
- Delete - O(1)
- Find - O(1)



碰撞 - Open Hashing vs Closed Hashing



知识点：

What’s Differences of:

- HashTable
- HashSet
- HashMap

C++标准库

提供map容器，可以插入，删除，查找key-value pair，

底层以平衡二叉搜索树的方式实现，根据key进行了排序。

在C++11中，标准库添加了unordered_map，更符合Hash table的传统定义，平均查找时间O(1)

String

在C语言中，字符串指的是一个以’\0’结尾的char数组。关于字符串的函数通常需要传入一个字符型指针。

在C++中，String是一个类，并且可以通过调用类函数实现判断字符串长度，字串等等操作。

工具箱：C语言中String常用函数

char *strcpy (char *destination, const char *source); //copy source string to destination string
char *strcat (char *destination, const char *source); //Appends a copy of the *source* string to the *destination* string.
int strcmp (const char *str1, const char *str2);
char *strstr (char *str1, const char *str2); // Returns a pointer to the first occurrence of *str2* in *str1*, or a NULL pointer if *str2* is not part of *str1*.
size_t strlen (const char *str); // Returns the length of the C string *str*.
double atof (const char *str); // convert char string to a double
int atoi (const char *str); // convert char string to an int

工具箱：C++中String类常用函数

String类重载了+, <, >, =, ==等运算符，故复制，比较是否相等，附加子串等都可以用运算符直接实现。请参考(http://www.cplusplus.com/reference/string/string/

size_t find (const string& str, size_t pos = 0) const; // Searches the string for the first occurrence of the str, returns index

string substr (size_t pos = 0, size_t len = npos) const; // Returns a newly constructed string object with its value initialized to a copy of a substring starting at pos with length len.
string &erase (size_t pos = 0, size_t len = npos); // erase characters from pos with length len
size_t length(); // Returns the length of the string, in terms of bytes

模式识别

当遇到某些题目需要统计一个元素集中元素出现的次数，应该直觉反应使用Hash Table，即使用std::unordered_map或std::map：key是元素，value是出现的次数。特别地，有一些题目仅仅需要判断元素出现与否(相当于判断value是0还是1)，可以用bitvector，即bitset，利用一个bit来表示当前的下标是否有值

例子1:

Determine if all characters of a string are unique.

一般来说，一旦出现“unique”，就落入使用**hash table**或者**bitset**来判断元素出现与否的范畴。

例子2:

Given two strings, determine if they are permutations of each other.

置换的特性：无论如何变化，每个字符出现的次数一定相同。一旦需要统计一个元素集中元素出现的次数，我们就应该想到**hash table**。
hash table 需要扫描整个string，平均时间复杂度都是O(n)。最后比较两个hash是否相同，每个合法字符都有可能出现，假设字符集大小为m，则需要的时间复杂度是O(m)，故总的时间复杂度O(m+n)。空间上，平均空间是O(m)。

解法2:
对每个string中的字符按照ASCII编码顺序进行排序。如果是一个置换，那么排序完的两个string应该相等。这样做的时间复杂度是O(n log n)，空间复杂度是O(n)。

例子3

Given a newspaper and message as two strings, check if the message can be composed using letters in the newspaper.

解题分析：message中用到的字符必须出现在newspaper中。其次，message中任意字符出现的次数一定少于其在newspaper中出现的次数。统计一个元素集中元素出现的次数，我们就应该想到**hash table**

复杂度分析：假设message长度为m，newspaper长度为n，我们的算法需要hash整条message和整个newspaper，故时间复杂度O(m+n)。假设字符集大小为c，则平均空间是O(c)

Anagram

Write a method anagram(s,t) to decide if two strings are anagrams or not.

Example
Given s="abcd", t="dcab", return true.

Challenge
O(n) time, O(1) extra space

模式识别

当处理当前节点需要依赖于之前的部分结果时，可以考虑使用hash table记录之前的处理结果。其本质类似于Dynamic Programming，利用hash table以O(1)的时间复杂度获得之前的结果。

例子4

Find a pair of two elements in an array, whose sum is a given target number.

最直观的方法是再次扫描数组，判断target – array[i]是否存在在数组中。这样做的时间复杂度是O(n^2)

如何保存之前的处理结果？可以使用**hash table** “target – array[i]是否存在在数组中

复杂度分析：数组中的每个元素进行上述hash处理，从头至尾扫描数组，判断对应的另一个数是否存在在数组中，时间复杂度O(n)

例子5

Get the length of the longest consecutive elements sequence in an array
For example, given [31, 6, 32, 1, 3, 2],the longest consecutive elements sequence is [1, 2, 3]. Return its length: 3.

判断**array[i] – 1，array[i] + 1**是否存在于数组中。如何保存之前的处理结果？可以使用**hash table** 由于序列是一系列连续整数，所以只要序列的最小值以及最大值，就能唯一确定序列。而所谓的“作为后继加入序列”，“作为前驱加入序列”，更新最大最小值。hash table的value可以是一个记录最大／最小值的structure，用以描述当前节点参与构成的最长序列。

时间复杂度O(n)，附加空间O(n)

Longest Common Substring

Given two strings, find the longest common substring.
Return the length of it.

Example
Given A="ABCD", B="CBCE", return 2.
Note
The characters in substring should occur continuously in original string.
This is different with subsequence.

友情提示

String相关问题的一些处理技巧。通常，纯粹的字符串操作难度不大，但是实现起来可能比较麻烦，edge case比较多。例如把字符串变成数字，把数字变成字符串等等。

需要与面试官进行沟通，明确他们期望的细节要求，再开始写代码。

可以利用一些子函数，使得代码结构更加清晰。考虑到时间限制，往往有的时候面试官会让你略去一些过于细节的实现。

Reverse Words in String

Given input -> "I have 36 books, 40 pens2."; output -> "I evah 36 skoob, 40 2snep.”

解题分析：
每个以空格或符号为间隔的单词逆向输出，如果遇到纯数字，则不改变顺序。自然而然地，每次处理分为两个步骤：1)判断是否需要逆向 2)逆向当前单词。这样就可以分为两个子函数：一个负责判断，另一个负责逆向。然后进行分段处理。

Rotate String

Given a string and an offset, rotate string by offset. (rotate from left to right)

Example
Given "abcdefg"

for offset=0, return "abcdefg"

for offset=1, return "gabcdef"

Array

Remove Element

Given an array and a value, remove all occurrences of that value in place and return the new length.

The order of elements can be changed, and the elements after the new length don't matter.

Example
Given an array [0,4,4,0,0,2,4,4], value=4

return 4 and front four elements of the array is [0,0,0,2]

Remove Duplicates from Sorted Array I

Given a sorted array, remove the duplicates in place such that each element appear only once and return the new length.

Do not allocate extra space for another array, you must do this in place with constant memory.

For example,
Given input array nums = [1,1,2],

Your function should return length = 2, with the first two elements of nums being 1 and 2 respectively. It doesn't matter what you leave beyond the new length.

Merge Sorted Array

1. Merge Two Sorted Array into a new Array
2. Merge Two Sorted Array A and B into A, assume A has enough space.

Partition

Given an array nums of integers and an int k, partition the array (i.e move the elements in "nums") such that:

All elements < k are moved to the left
All elements >= k are moved to the right
Return the partitioning index, i.e the first index i nums[i] >= k.

Example
If nums=[3,2,2,1] and k=2, a valid answer is 1.

If all elements in nums are smaller than k, then return nums.length
Challenge
Can you partition the array in-place and in O(n)?

Median

Given a unsorted array with integers, find the median of it.

A median is the middle number of the array after it is sorted.

If there are even numbers in the array, return the N/2-th number after sorted.

Example
Given [4, 5, 1, 2, 3], return 3

Given [7, 9, 4, 5], return 5

Homework

Replace space in the string with “%20”. E.g. given "Big mac", return “Big%20mac”

解题思路：要求in-place的删除或修改，可以用两个int变量分别记录新index与原index，不断地将原index所指的数据写到新index中。

如果改动后string长度增大，则先计算新string的长度，再从后往前对新string进行赋值；反之，则先从前往后顺序赋值，再计算长度。

Homework

Given an array of strings, return all groups of strings that are anagrams.

Example
Given ["lint", "intl", "inlt", "code"], return ["lint", "inlt", "intl"].

Given ["ab", "ba", "cd", "dc", "e"], return ["ab", "ba", "cd", "dc"].
Note
All inputs will be in lower-case

Homework

实现*HashTable*