1.**先写binary search**

**follow up1** 有duplicates的情况，返回任意index怎么做，返回最左边的index怎么做

if(input[mid] == target && (mid == 0 || input[mid-1] != target)) return mid;

**follow up2** 是[9, 8, 7, 1, 2, 3, 4] binary search. search in increasing and then decreasing array

pivot = Min in rotated sorted array

然后找到pivot后，我说然后再分别在左半边和右半边查找。

int BS(vector<int> &input){

int left = 0;

int right = input.size() - 1;

while(left <= right){

int mid = left + (right - left)/2;

if(mid + 1 < input.size() && input[mid] < input[mid+1] && mid > 0 && input[mid] < input[mid-1])

return mid;

else if(mid + 1 < input.size() && input[mid] > input[mid+1]) left = mid + 1;

else right = mid - 1;

}

return left;

}

**1.Missing Number 变种**

**从[x,y]中找到missing number, 输入是一个sorted array (如果未排序，使用xor)**

**if it is already sorted, use binary search O(logn) time**

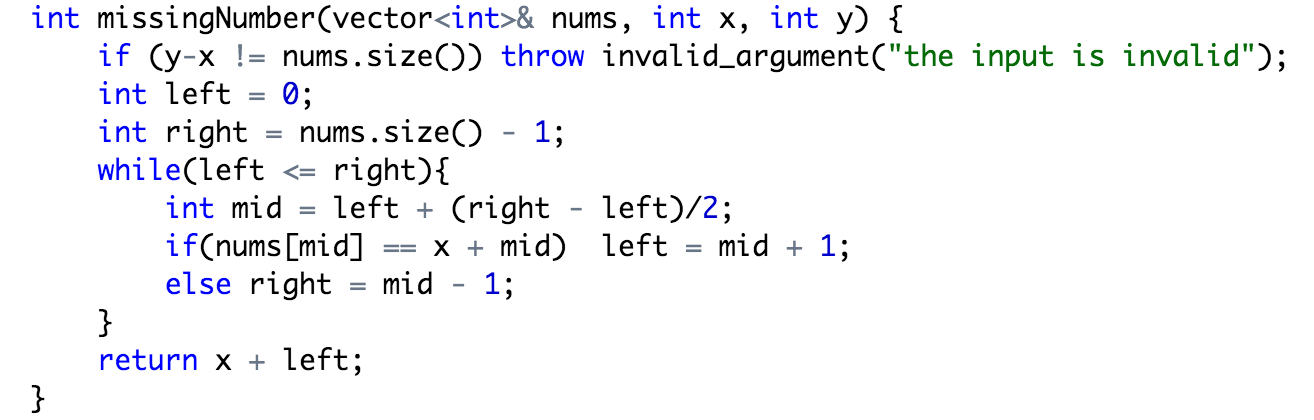
**最后 left points to the first number which does does not match the position.**

**Corner Case**

**1. x = y []**

**2. [0] x = 0, y = 1**

**3. [1] x = 0, y = 1**



**2.Single Number 变种**

**给的数组里面pair是一起的，比如这样［1，1，4，3，3，7，7］**

**1.hashtable, xor O(n) time（如果数字随即放）**

**2.Binary Search**

int bs(vector<int> &input){

int left = 0;

int right = input.size() - 1;

while(left <= right){

int mid = left + (right - left)/2;

if(mid > 0 && input[mid] == input[mid-1])

mid&1 ? left = mid + 1 : right = mid - 2;

else if(mid + 1 < input.size() && input[mid] == input[mid+1])

mid&1 ? right = mid - 1 : left = mid + 2;

else return input[mid];

}

throw invalid\_argument("invalid input");

}

**4.给一个数组[a][b][c][d]找到第一个broken code**

其实就是给一个sorted array 0 0 0 1 1 1 1 然后找出第一个为1 的数

// std::vector<int>::iterator low = lower\_bound (v.begin(), v.end(), 1);

Corner Case：

1. array为空 || 比如数组全0的时候, 返回值为n

int bs(vector<int> &input){

int left = 0;

int right = input.size() - 1;

while(left <= right){

int mid = left + (right - left)/2;

if(input[mid] == 0) left = mid + 1;

else right = mid - 1;

}

return left;

}