Lecture 12 Binary Search

1. <https://leetcode.com/problems/binary-search/>

class Solution {

public:

int search(vector<int>& nums, int target) {

int s = 0;

int e = nums.size()-1;

while(s<=e)

{

int mid = (s+e)/2;

if(nums[mid]==target)

{

return mid;

}

else if(nums[mid] < target)

{

s = mid+1;

}

else

{

e = mid-1;

}

}

return -1;

}

};

1. <https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/>

class Solution {

public:

vector<int> searchRange(vector<int>& nums, int target) {

int n = nums.size();

int s = 0;

int e = n-1;

int first = -1;

while(s<=e)

{

int mid = (s+e)/2;

if(nums[mid]==target)

{

first = mid;

e = mid - 1;

}

else if(nums[mid] > target)

{

e = mid-1;

}

else

{

s = mid+1;

}

}

s = 0;

e = n-1;

int last = -1;

while(s<=e)

{

int mid = (s+e)/2;

if(nums[mid]==target)

{

last = mid;

s = mid+1;

}

else if(nums[mid] > target)

{

e = mid-1;

}

else

{

s = mid+1;

}

}

return {first, last};

}

};

1. <https://leetcode.com/problems/sqrtx/>

class Solution {

public:

int mySqrt(int x) {

int s = 0;

int e = x;

int res = 0;

while(s<=e)

{

long long int mid = (s+e)/2;

if(mid\*mid==x)

{

return mid;

}

else if(mid\*mid > x)

{

e = mid-1;

}

else

{

res = mid;

s = mid+1;

}

}

return res;

}

};

1. <https://leetcode.com/problems/search-a-2d-matrix/>

class Solution {

public:

bool searchMatrix(vector<vector<int>>& matrix, int target) {

int rows = matrix.size();

int cols = matrix[0].size();

int s = 0;

int e = cols-1;

while(s < rows && e >= 0)

{

if(matrix[s][e]==target)

{

return true;

}

else if(matrix[s][e]>target)

{

e--;

}

else

{

s++;

}

}

return false;

}

};