Lecture 14 Searching, Sorting and Prime no.

1. <https://leetcode.com/problems/merge-intervals/>

class Solution {

public:

vector<vector<int>> merge(vector<vector<int>>& intervals) {

int n = intervals.size();

vector<vector<int>> result;

sort(intervals.begin(), intervals.end());

int s1 = intervals[0][0];

int e1 = intervals[0][1];

for(int i=1; i<n; i++)

{

int s2 = intervals[i][0];

int e2 = intervals[i][1];

if(s2 <=e1)

{

e1 = max(e1, e2);

}

else

{

result.push\_back({s1, e1});

s1 = s2;

e1 = e2;

}

}

result.push\_back({s1, e1});

return result;

}

};

1. <https://leetcode.com/problems/search-in-rotated-sorted-array/>

class Solution {

public:

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

int n = nums.size();

int s = 0;

int e = n-1;

while(s<=e)

{

int mid = (s+e)/2;

if(nums[mid]==target)

{

return mid;

}

else if(nums[s] <= nums[mid])

{

if(nums[s] <= target && target <= nums[mid])

{

e = mid - 1;

}

else

{

s = mid + 1;

}

}

else

{

if(nums[mid] <= target && target <= nums[e])

{

s = mid+1;

}

else

{

e = mid-1;

}

}

}

return -1;

}

};

1. <https://leetcode.com/problems/count-primes/>

class Solution {

void prime\_sieve(vector<bool> &p, int n)

{

for(long long int i=3; i<=n; i+=2)

{

p[i] = true;

}

for(long long int i=3; i<=n; i+=2)

{

if(p[i]==true)

{

for(long long int j=i\*i; j<=n; j=j+i)

{

p[j] = false;

}

}

}

p[2] = true;

p[1] = p[0] = false;

}

public:

int countPrimes(int n) {

if(n==0 || n==1)

{

return 0;

}

vector<bool> p(n+1, false);

prime\_sieve(p, n);

int count = 0;

for(long long int i=0; i<n; i++)

{

if(p[i]==true)

{

count++;

}

}

return count;

}

};