Laboratory work #1. Arrays and String.

Solve all of problems for Arrays and for Strings.

Make a report on each tasks you have solved.

Deadline: 15th September 2020.

Arrays

1. <https://leetcode.com/problems/range-sum-query-immutable/>

class NumArray {

public:

vector<int> prefixSum;

NumArray(vector<int>& nums) {

prefixSum.resize(nums.size());

for (int i = 0; i < nums.size(); i++){

prefixSum[i]=nums[i];

if (i > 0){

prefixSum[i]+=prefixSum[i-1];

}

}

}

int sumRange(int i, int j) {

int sum=prefixSum[j];

if (i>0)

sum=sum-prefixSum[i-1];

return sum;

}

};

/\*\*

\* Your NumArray object will be instantiated and called as such:

\* NumArray\* obj = new NumArray(nums);

\* int param\_1 = obj->sumRange(i,j);

\*/

The following problem was solved using Prefix Sum. We declared vector Prefix sum which stores the sum of elements from beginning to the index(i).

For example, if there is such array as nums = [1,2,3]:

prefixSum will store as [1,1+2,1+2+3].

After filling the PrefixSum with all the previous elements, in method sumRange, which has indexes of range: i and j. The prefixSum will return the sum of elements from index 0 to j. But we need the sum only from index I that’s why we just reduce sum of all elements from 0 to i (i not inclusive).

1. <https://leetcode.com/problems/maximum-subarray/>

class Solution {

public:

int maxSubArray(vector<int>& nums) {

int maxSum=nums[0], sum=nums[0];

for (int i=1; i<nums.size();i++){

sum += nums[i];

if (sum < nums[i]){

sum = nums[i];

}

if (maxSum<sum) {

maxSum=sum;

}

}

return maxSum;

}

};

In this problem, the vector nums is given and we have to find its subbaray which’s sum is most among other subarrays.

As you can see, I made the first element of nums as maxSum and sum. Then I looped through array starting from 2nd element to make “sum” as sums of previous nums.

As you can see, if current sum is less than current element (nums[i]), there is no need to store the sums of previous elements, therefore new sum will be equal to the current element.

Also, if maxSum is less than current sum, current sum will be our new MaxSum. Then we just return maxSum.

1. <https://leetcode.com/problems/product-of-array-except-self/>

Here we have to get the product of all elements except t-th (current) element. And to do so we created product from left to right array and vice versa. We assigned it’s first of left and last of right as 1. The solution is if we multiple this l[i] and r[i] together, it will give us the product of array except i. For example,

Given array {1,2,3,4,5}  
The left array will be {1,1,2,6,24}

And right will be {120,60,20,5,1}

Then we multiply l[i] with r[i] : {120,60,40,30,24} and get f-final array.

class Solution {

public:

vector<int> productExceptSelf(vector<int>& nums) {

vector<int> f;

int s = nums.size();

f.resize(s);

int l[s],r[s];

l[0]=1;

r[s-1]=1;

for (int i = 1; i < s; i++) {

l[i] = l[i-1]\*nums[i-1];

}

for (int i = s-2; i >= 0; i--) {

r[i] = r[i+1] \* nums[i+1];

}

for (int i = 0; i<s;i++) {

f[i] = l[i]\*r[i];

}

return f;

}

};

1. https://leetcode.com/problems/missing-number/

class Solution {

public:

int missingNumber(vector<int>& nums) {

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

int expectedNum=0;

for (int i = 0; i < nums.size(); i++) {

if (nums[i] != expectedNum) {

return expectedNum;

}

else {

expectedNum++;

}

}

return expectedNum;

}

};

To solve this problem, I just sorted array using sort method.

Then I declared int expectedNum = 0. And if expectedNum is not equal to the nums[i], we return that number. Else, we add 1 to expectedNum. For example given sorted array:

Nums{0,1,2,3,5} – checks if expectedNum == nums[i], (expectedNum=0,nums[i]=0).  
And so on until i=4 and expectedNum=4. Our nums[i] = 5 therefore we return expectedNum.

Also the method return last element if in the loop there was no missing element found.

1. <https://leetcode.com/problems/maximum-average-subarray-i/>

class Solution {

public:

double findMaxAverage(vector<int>& nums, int k) {

int prefixSum[nums.size()];

for (int i = 0; i < nums.size(); i++){

prefixSum[i]=nums[i];

if (i > 0){

prefixSum[i]+=prefixSum[i-1];

}

}

double maxAvg=prefixSum[k-1]\*1.0/k; //our first avereage

for (int i=k; i< nums.size();i++){

if (maxAvg < (prefixSum[i]-prefixSum[i-k])\*1.0/k){

maxAvg = (prefixSum[i]-prefixSum[i-k])\*1.0/k;

}

}

return maxAvg;

}

};

As in the 1st problem, I used the prefixSum to find all the previous sums of given elements. And then, I declared maxAvg. And it takes the prefixSum[k-1] and divide it to k so for now its our first and maxAvg.

Then we just start the loop from k and compare each time maxAvg with   
currentAvg=( prefixSum[i]-prefixSum[i-k])\*1.0/k). If current avg is more than maxAvg, current avg will be our new maxAvg.

1. <https://leetcode.com/problems/range-sum-query-2d-immutable/>

The solution is not correct at all because of the time limit but simple cases are running correctly. As you see we just add the elements which are in 2d-subarray to the sum.

class NumMatrix {

public:

vector<vector<int>> data;

NumMatrix(vector<vector<int>>& matrix) {

data=matrix;

}

int sumRegion(int row1, int col1, int row2, int col2) {

int sum = 0;

for (int r=row1; r <= row2;r++){

for (int c=col1; c<=col2;c++){

sum+=data[r][c];

}

}

return sum;

}

};

1. <https://leetcode.com/problems/rotate-image/>

class Solution {

public:

void rotate(vector<vector<int>>& matrix) {

vector<vector <int>> temp;

vector<int>temp1;

for (int i = 0; i < matrix.size(); ++i){

temp1 = {};

for (int j = matrix.size()-1; j >= 0; j--) {

temp1.push\_back(matrix[j][i]);

}

temp.push\_back(temp1);

}

matrix = temp;

}

};

--------------

1. <https://leetcode.com/problems/reverse-words-in-a-string/>

Wrong!(Has some mistake somewhere)

In this problem we have to print words in reversed order. Here I just divided the words by spaces and stored in a vector of string. And then I just created f-final string and added the strings backwards.

class Solution {

public:

string reverseWords(string s) {

vector <string> arr;

int cur=0;

for (int i=0;i<s.size();i++){

if (isspace(s[i])){

string el=""+s.substr(cur,i);

arr.push\_back(el);

cur=i+1;

}

}

string f="";

for (int i=arr.size(); i>=0;i++){

f += arr[i];

}

return f;

}

};

1. <https://leetcode.com/problems/license-key-formatting/>

class Solution {

public:

string licenseKeyFormatting(string S, int K) {

string res;

int count=0;

for(int i = S.size() - 1; i >= 0; --i){

if (count == K && S[i] != '-'){

res = '-' + res;

count = 0;

}

if (S[i] != '-') {

res = (char)toupper(S[i]) + res;

count++;

}

}

return res;

}

};

1. <https://leetcode.com/problems/string-to-integer-atoi/>
2. <https://leetcode.com/problems/integer-to-english-words/>
3. <https://leetcode.com/problems/reverse-words-in-a-string-iii/>

class Solution {

public:

string reverseWords(string s) {

s = " " + s;

string cur\_word;

vector <string> rew;

for (int i = s.size(); i >= 0; --i){

if (s[i] == ' ' || i == s.size()){

cur\_word+=" ";

rew.insert(rew.begin(),cur\_word);

cur\_word.clear();

}

else {

cur\_word += s[i];

}

}a

string res;

for (auto w : rew ){

res += w ;

}

res = res.substr(0,res.size()-2);

return res;

}

};