1) Delete mid of stack

06 November 2024 00:02

Given a stack, delete the **middle element** of the stack without using any additional data structure.

Middle element:- floor((size_of_stack+1)/2) (1-based indexing) from the bottom of the stack.

From <https://www.geeksforgeeks.org/problems/delete-middle-element-of-a-stack/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=practice_card>

```
void helper(stack<int>& s, int pos){
    // Base case: If position is 1, pop the middle element
    if(pos == 1){
        s.pop();
        return;
    }

    // Store top element and pop it to reach the middle
    int temp = s.top();
    s.pop();

    // Recursive call with decremented position
    helper(s, pos - 1);

    // Push the stored element back after recursive call
    s.push(temp);
}

void deleteMid(stack<int>& s, int size) {
    if(s.empty()) return;

    // Calculate the middle position correctly for both odd and even sizes
    int pos= floor((size/2)+1);  // Use 1-based indexing for middle
    helper(s, pos);
}
```

2) Insert an element at the bottom of stack

06 November 2024 00:32

You are given a stack **st** of **n** integers and an element **x**. You have to insert **x** at the bottom of the given stack.

From <https://www.geeksforgeeks.org/problems/insert-an-element-at-the-bottom-of-a-stack/1>

```
void helper(stack<int> &st,int x){
    if(st.empty()){
        st.push(x);
        return;
    }

    //1 case solved
    int temp = st.top();
    st.pop();

    //recursion
    helper(st, x);

    //backtracking
    st.push(temp);
}
stack<int> insertAtBottom(stack<int> st,int x){
    int n = st.size();
    if(n == 0) return st;

    helper(st, x);
    return st;
}
```

3) Reverse a stack using recursion

06 November 2024 00:56

You are given a stack **St**. You have to reverse the stack using recursion.

From <https://www.geeksforgeeks.org/problems/reverse-a-stack/1>

```
void insertAtBottom(stack<int> &st, int n){
    if(st.empty()){
        st.push(n);
        return;
    }

    int temp = st.top();
    st.pop();

    insertAtBottom(st, n);

    st.push(temp);
}
void Reverse(stack<int> &st) {
    if(st.empty()) return;
    int temp = st.top(); st.pop();

    Reverse(st);

    insertAtBottom(st, temp);
}
```

4) Insert an element in an already sorted stack

06 November 2024 01:08

```
void insertInSortedStack(stack<int> &st, int x){
    if(st.empty() || x > st.top()){
        st.push(x);
        return;
    }
    int temp = st.top();
    st.pop();
    //recursion
    insertInSortedStack(st, x);
    //backtrack
    st.push(temp);
}
```

5) Sort a stack using recursion (based on prev problem)

06 November 2024 01:09

You're given a stack consisting of 'N' integers. Your task is to sort this stack in descending order using recursion.
We can only use the following functions on this stack S.

From <https://www.naukri.com/code360/problems/sort-a-stack_985275?leftPanelTabValue=PROBLEM>

```
void insertInSortedStack(stack<int> &st, int x){
    if(st.empty() || x > st.top()){
        st.push(x);
        return;
    }
    int temp = st.top();
    st.pop();
    //recursion
    insertInSortedStack(st, x);
    //backtrack
    st.push(temp);
}
void sortStack(stack<int> &st)
{
    if(st.empty()) return ;
    int temp = st.top();
    st.pop();
    //recursion
    sortStack(st);
    //backtrack
    insertInSortedStack(st, temp);
}
```

6) Stack implementation using array

06 November 2024 01:15

```
class Stack {
  public:
    int* arr;
    int size;
    int top;

    Stack(int size) {
      arr = new int[size];
      this->size = size;
      this->top = -1;
    }

    void push(int data) {
      if(top == size-1) {
        cout << "Stack overflow" << endl;
        return;
      }
      else {
        top++;
        arr[top] = data;
      }
    }

    void pop() {
      if(top == -1) {
        cout << "Stack underflow" << endl;
        return;
      }
      else {
        top--;
      }
    }

    bool isEmpty() {
      if(top == -1) {
        return true;
      }
      else {
        return false;
      }
    }

    int getTop() {
      if(top == -1) {
        cout << "Stack is empty" << endl;
        return -1;
      }
      else {
        return arr[top];
      }
```

```
    }

    int getSize() {
      return top+1;
    }
};
```

7) Implement two stacks using single array

06 November 2024 01:22

```
class Stack{
 public:
  int* arr;
  int size;
  int top1;
  int top2;

 Stack(int size){
  arr = new int[size];
  this->size = size;
  top1 = -1;
  top2 = size;
 }

 void push1(int data){
  if(top2 - top1 == 1){
   cout << "Stack overflow" << endl;
  }
  else{
   top1 ++;
   arr[top1] = data;
  }
 }

 void push2(int data){
  if(top2 - top1 == 1){
   cout << "Stack overflow" << endl;
  }
  else{
   top2 --;
   arr[top2] = data;
  }
 }

 void pop1(){
  if(top1 == -1){
   cout << "Stack underflow" << endl;
  }
  else{
   arr[top1] = 0;
   top1--;
  }
 }

 void pop2(){
  if(top2 == size){
   cout << "Stack underflow" << endl;
  }
  else{
   arr[top2] = 0;
```

```
    top2++;
   }
  }
};
```

8)Valid parenthesis

06 November 2024 01:36

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

From <https://leetcode.com/problems/valid-parentheses/description/>

```
void helper(stack<char>& st, string& s){
    for(int i = 0; i < s.length(); i++){
        if(!st.empty() && st.top() == '(' && s[i] == ')'){
            st.pop(); continue;
        }
        if(!st.empty() && st.top() == '[' && s[i] == ']'){
            st.pop(); continue;
        }
        if(!st.empty() && st.top() == '{' && s[i] == '}'){
            st.pop(); continue;
        }
        st.push(s[i]);
    }
}
bool isValid(string s) {
    stack<char> st;
    if(s.length() == 0) return true;
    if(s.length() == 1) return false;
    helper(st, s);
    if(st.empty()) return true;
    return false;
}
```

From <https://leetcode.com/problems/valid-parentheses/submissions/1444182403/>

9) Redundant brackets

06 November 2024 01:47

Given valid mathematical expressions in the form of a string. You are supposed to return true if the given expression contains a pair of redundant brackets, else return false. The given string only contains ‘(‘, ’)’, ‘+’, ‘-’, ‘*’, ‘/’ and lowercase English letters.

From <https://www.naukri.com/code360/problems/redundant-brackets_975473?leftPanelTabValue=PROBLEM>

```
bool findRedundantBrackets(string &s){
  bool flag = false;
  stack<char>st;
  for (int i = 0; i < s.size(); i++){
    if(s[i] == '+' || s[i] == '*' ||s[i] == '-' ||s[i] == '/' || s[i] == '('){
      st.push(s[i]);
    }
    if(s[i] == ')'){
      if(!st.empty() && st.top() == '(') flag = true;
      while(st.top() == '+' || st.top()  == '*' ||st.top()  == '-' ||st.top()  == '/') st.pop();
      st.pop();
    }
  }
  return flag;
}
```

11) Infix to postfix

06 November 2024 23:37

Given an infix expression in the form of string **str**. Convert this infix expression to postfix expression.

From <https://www.geeksforgeeks.org/problems/infix-to-postfix-1587115620/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=practice_card>

```
int priority(char ch){
    if(ch == '^') return 3;
    if(ch == '+' || ch == '-') return 1;
    if(ch == '*' || ch == '/') return 2;
    return -1;
}
string infixToPostfix(string& s) {
    int n = s.length();
    int i = 0;
    stack<char>st;
    string ans = "";

    while(i < n){
        if((s[i] >= 'A' && s[i] <= 'Z') || (s[i] >= 'a' && s[i] <= 'z') ||(s[i] >= '0' && s[i] <= '9')){
            ans += s[i];
        }
        else if(s[i] == '(') st.push(s[i]);
        else if(s[i] == ')'){
            while(!st.empty() && st.top() != '('){
                ans += st.top();
                st.pop();
            }
            st.pop();
        }

        else{
            while(!st.empty() && priority(s[i]) <= priority(st.top())){
                ans += st.top();
                st.pop();
            }
            st.push(s[i]);
        }
        i++;
    }
    while(!st.empty()){
        ans += st.top();
        st.pop();
    }
    return ans;
}
```

12) Infix to prefix

06 November 2024 23:38

```
int priority(char ch){
   if(ch == '^') return 3;
   if(ch == '*' || ch == '/') return 2;
   if(ch == '+' || ch == '-') return 1;
   return -1;
}

std::string infixToPrefix(std::string& s) {
   int n = s.length();
   std::string reversed = "", ans = "";
   std::stack<char> st;

   for(int i = n - 1; i >= 0; i--) {
      if(s[i] == '(') reversed += ')';
      else if(s[i] == ')') reversed += '(';
      else reversed += s[i];
   }

   for(int i = 0; i < n; i++) {
      char ch = reversed[i];
      if(isalnum(ch)) {
         ans += ch;
      }
      else if(ch == '(') {
         st.push(ch);
      }
      else if(ch == ')') {
         while(!st.empty() && st.top() != '(') {
            ans += st.top();
            st.pop();
         }
         if(!st.empty()) st.pop();
      }
      else {
         while(!st.empty() && priority(ch) <= priority(st.top())) {
            if (ch == '^' && st.top() == '^') {
               break;
            }
            ans += st.top();
            st.pop();
         }
         st.push(ch);
      }
   }

   while(!st.empty()) {
      ans += st.top();
      st.pop();
   }

   std::reverse(ans.begin(), ans.end());
   return ans;
}
```

13) Postfix to prefix

06 November 2024 23:39

You are given a string that represents the postfix form of a valid mathematical expression. Convert it to its prefix form.

From <https://www.geeksforgeeks.org/problems/postfix-to-prefix-conversion/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=practice_card>

```
string postToPre(string s) {
    stack<string> st;
    for (int i = 0; i < s.length(); i++) {
        if (isalpha(s[i]) || isdigit(s[i])) {
            // Convert char to string and push to stack
            st.push(string(1, s[i]));
        } else {
            // Pop two operands from the stack for the operator
            string op2 = st.top();
            st.pop();
            string op1 = st.top();
            st.pop();
            // Form the prefix expression and push back to stack
            string con = s[i] + op1 + op2;
            st.push(con);
        }
    }
    return st.top();
}
```

14) Prefix to postfix

06 November 2024 23:40

15) Postfix to Infix

06 November 2024 23:41

You are given a string that represents the postfix form of a valid mathematical expression. Convert it to its infix form.

From <https://www.geeksforgeeks.org/problems/postfix-to-infix-conversion/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=practice_card>

```
string postToInfix(string s) {
    stack<string> st;
    int n = s.length();
    for(int i = 0; i < n; i++) {
        if(isalnum(s[i])) {
            st.push(string(1, s[i]));
        }
        else {
            string op1 = st.top(); st.pop();
            string op2 = st.top(); st.pop();
            string temp = "(" + op2 + s[i] + op1 + ")";
            st.push(temp);
        }
    }
    return st.top();
}
```

16) Prefix to infix

06 November 2024 23:41

You are given a string **S** of size **N** that represents the prefix form of a valid mathematical expression. The string **S** contains only lowercase and uppercase alphabets as operands and the operators are +, -, *, /, %, and ^.Convert it to its infix form.

From <https://www.geeksforgeeks.org/problems/prefix-to-infix-conversion/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=practice_card>

```
string preToInfix(string s) {
    stack<string> st;
    int n = s.length();
    for(int i = n-1; i >= 0; i--) {
        if(isalnum(s[i])) {
            st.push(string(1, s[i]));
        }
        else {
            string op1 = st.top(); st.pop();
            string op2 = st.top(); st.pop();
            string temp = "(" + op1 + s[i] + op2 + ")";
            st.push(temp);
        }
    }
    return st.top();
}
```

17) Implement min stack

06 November 2024 23:52

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.
Implement the MinStack class:

- MinStack() initializes the stack object.
- void push(int val) pushes the element val onto the stack.
- void pop() removes the element on the top of the stack.
- int top() gets the top element of the stack.
- int getMin() retrieves the minimum element in the stack.

You must implement a solution with O(1) time complexity for each function.

From <https://leetcode.com/problems/min-stack/description/>

```
vector<pair<int, int>>st;
MinStack() {

}

void push(int val) {
    if(st.empty()){
        pair<int, int> p;
        p.first = val; p.second = val;
        st.push_back(p);
    }
    else{
        pair<int, int> p;
        p.first = val;
        p.second = min(st.back().second, val);
        st.push_back(p);
    }

}

void pop() {
    st.pop_back();
}

int top() {
    return st.back().first;
}

int getMin() {
    return st.back().second;
}
```

From <https://leetcode.com/problems/min-stack/submissions/1184776607/>

18) Problem Based on (next smaller element)

07 November 2024 00:21

You are given an integer array prices where prices[i] is the price of the i^{th} item in a shop.
There is a special discount for items in the shop. If you buy the i^{th} item, then you will receive a discount equivalent to prices[j] where j is the minimum index such that j > i and prices[j] <= prices[i]. Otherwise, you will not receive any discount at all.
Return an integer array answer where answer[i] is the final price you will pay for the i^{th} item of the shop, considering the special discount.

From <https://leetcode.com/problems/final-prices-with-a-special-discount-in-a-shop/description/>

```
vector<int> finalPrices(vector<int>& prices) {
    stack<int> st;
    int n = prices.size();
    vector<int> result(n);
    for (int i = n - 1; i >= 0; i--) {
        while (!st.empty() && st.top() > prices[i]) st.pop();

        if (!st.empty()) result[i] = prices[i] - st.top();
        else result[i] = prices[i];

        st.push(prices[i]);
    }
    return result;
}
```

19) Smaller on left (same as next smaller element)

07 November 2024 00:35

Given an array **a** of integers of length **n**, find the nearest smaller number for every element such that the smaller element is on left side.If no small element present on the left print -1.

From <https://www.geeksforgeeks.org/problems/smallest-number-on-left3403/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=practice_card>

```
vector<int> leftSmaller(int n, int a[]){
    stack<int> st;
    vector<int> ans(n);

    for (int i = 0; i < n; i++) {
        while (!st.empty() && st.top() >= a[i]) st.pop();

        if (!st.empty()) ans[i] = st.top();
        else ans[i] = -1;

        st.push(a[i]);
    }
    return ans;
}
```

20) Largest rectangle in histogram

07 November 2024 01:00

Given an array of integers heights representing the histogram's bar height where the width of each bar is 1, return *the area of the largest rectangle in the histogram*.

From <https://leetcode.com/problems/largest-rectangle-in-histogram/description/>

```
vector<int> nextSmaller(vector<int>& heights) {
    int n = heights.size();
    vector<int> ans(n);
    stack<int> st;
    for (int i = n - 1; i >= 0; i--) {
        while (!st.empty() && heights[st.top()] >= heights[i])
            st.pop();
        ans[i] = st.empty() ? n : st.top();
        st.push(i);
    }
    return ans;
}
vector<int> prevSmaller(vector<int>& heights) {
    int n = heights.size();
    vector<int> ans(n);
    stack<int> st;
    for (int i = 0; i < n; i++) {
        while (!st.empty() && heights[st.top()] >= heights[i])
            st.pop();
        ans[i] = st.empty() ? -1 : st.top();
        st.push(i);
    }
    return ans;
}
int largestRectangleArea(vector<int>& heights) {
    int n = heights.size();
    vector<int> next = nextSmaller(heights);
    vector<int> prev = prevSmaller(heights);
    int maxi = 0;
    for (int i = 0; i < n; i++) {
        maxi = max(maxi, heights[i] * (next[i] - prev[i] - 1)) ;
    }
    return maxi;
}
```

22) Next greater Element I (based on prev problem)

07 November 2024 01:56

The **next greater element** of some element x in an array is the **first greater** element that is **to the right** of x in the same array.
You are given two **distinct 0-indexed** integer arrays nums1 and nums2, where nums1 is a subset of nums2.
For each 0 <= i < nums1.length, find the index j such that nums1[i] == nums2[j] and determine the **next greater element** of nums2[j] in nums2. If there is no next greater element, then the answer for this query is -1.

From <https://leetcode.com/problems/next-greater-element-i/description/>

```
vector<int> nextGreaterElement(vector<int>& arr) {
    int n = arr.size();
    vector<int> ans(n);
    stack<int> st;
    for(int i = n - 1; i >= 0; i--) {
        while(!st.empty() && st.top() <= arr[i]) {
            st.pop();
        }

        ans[i] = st.empty() ? -1 : st.top();

        st.push(arr[i]);
    }
    return ans;
}
vector<int> nextGreaterElement(vector<int>& nums1, vector<int>& nums2) {
    int n1 = nums1.size();
    int n2 = nums2.size();
    vector<int> temp = nextGreaterElement(nums2);
    vector<int> ans(n1);
    for(int i = 0; i < nums1.size(); i++){
        for(int j = 0; j < nums2.size(); j++){
            if(nums1[i] == nums2[j]){
                ans[i] = temp[j];
            }
        }
    }
    return ans;
}
```

23) Next greater element II

07 November 2024 02:10

Given a circular integer array nums (i.e., the next element of nums[nums.length - 1] is nums[0]), return *the* ***next greater number*** *for every element in* nums.
The **next greater number** of a number x is the first greater number to its traversing-order next in the array, which means you could search circularly to find its next greater number. If it doesn't exist, return -1 for this number.

From <https://leetcode.com/problems/next-greater-element-ii/description/>

```
    vector<int> nextGreaterElements(vector<int>& nums) {
        int n = nums.size();
        stack<int> st;
        vector<int> ans(n);
        for(int i = 2*n-1; i >= 0; i--){
            while(!st.empty() && st.top() <= nums[i%n]){
                st.pop();
            }
            if(i < n){
                ans[i] = st.empty()? -1:st.top();
            }
            st.push(nums[i%n]);
        }
        return ans;
    }
```

Or its very blunt
||

```
    vector<int> nextGreaterElements(vector<int>& nums) {
        vector<int> ans;
        int n = nums.size();
        for(int i = 0; i < nums.size(); i++){
            ans.push_back(nums[i]);
        }
        for(int i = 0; i < nums.size(); i++){
            ans.push_back(nums[i]);
        }
        stack<int> st;
        vector<int> v(2*n);
        for(int i = ans.size()-1; i>= 0; i--){
            while(!st.empty() && st.top() <= ans[i]){
                st.pop();
            }
            v[i] = st.empty()? -1 : st.top();
            st.push(ans[i]);
        }
        vector<int>v2(n);
        for(int i = 0; i < n; i++){
            v2[i] = v[i];
        }
        return v2;
    }
```

24) Asteroid Collision

07 November 2024 02:49

We are given an array asteroids of integers representing asteroids in a row.
For each asteroid, the absolute value represents its size, and the sign represents its direction (positive meaning right, negative meaning left). Each asteroid moves at the same speed.
Find out the state of the asteroids after all collisions. If two asteroids meet, the smaller one will explode. If both are the same size, both will explode. Two asteroids moving in the same direction will never meet.

From <https://leetcode.com/problems/asteroid-collision/description/>

```
vector<int> asteroidCollision(vector<int>& asteroids) {
    stack<int> st;
    for(auto ast: asteroids){
        bool destroy = false;//initially nothing is destroyed
        if(ast > 0){
            st.push(ast);
        }
        else{
            if(st.empty() || st.top() <0){
                st.push(ast);
            }
            else{
                //collision happens only when st.top() > 0 && ast < 0
                while(!st.empty() && st.top() > 0){
                    if(abs(ast) == st.top()){
                        destroy = true;
                        st.pop();
                        break;
                    }
                    else if(abs(ast) > st.top()){
                        st.pop();
                    }
                    else{
                        destroy = true;
                        break;
                    }
                }
                if(!destroy){
                    st.push(ast);
                }
            }
        }
    }
    vector<int>ans(st.size());
    for (int i = st.size() - 1; i >= 0; i--){
        ans[i] = st.top();
        st.pop();
    }
    return ans;
}
```

From <https://leetcode.com/problems/asteroid-collision/submissions/1445177033/>

25) Remove K digits

07 November 2024 02:58

Given string num representing a non-negative integer num, and an integer k, return *the smallest possible integer after removing* k *digits from* num.

From <https://leetcode.com/problems/remove-k-digits/description/>

```
string removeKdigits(string num, int k) {
    string ans;
    stack <char> st;
    for (auto digit : num){
        if(k>0){
            while(!st.empty() && st.top() > digit){
                st.pop();
                k--;
                if(k == 0) break;
            }
        }
        st.push(digit);
    }
    if(k > 0){
        while(!st.empty() && k){
            st.pop();
            k--;
        }
    }
    while(!st.empty()){
        ans.push_back(st.top());
        st.pop();
    }
    //removing leading zeroes
    while(ans.size() > 0 && ans.back() == '0'){
        ans.pop_back();
    }
    //get real ans
    reverse (ans.begin(), ans.end());
    return ans == "" ? "0" : ans;
}
```

From <https://leetcode.com/problems/remove-k-digits/submissions/1229553971/>

26) Sum of subarray minimum

09 November 2024 02:44

Given an array of integers arr, find the sum of min(b), where b ranges over every (contiguous) subarray of arr. Since the answer may be large, return the answer modulo $10^9 + 7$.

From <https://leetcode.com/problems/sum-of-subarray-minimums/description/>

```
vector<int> nextSmallerElement(vector<int> &arr, int n) {
    vector<int> ans(n);
    stack<int> st;
    for (int i = n - 1; i >= 0; i--) {
        if(st.empty()){
            ans[i] = n;
        }
        while (!st.empty() && arr[st.top()] >= arr[i]) {
            st.pop();
        }
        ans[i] = st.empty() ? n : st.top();
        st.push(i);
    }
    return ans;
}
vector<int> leftSmaller(vector<int> &arr, int n) {
    vector<int> ans(n);
    stack<int> st;
    for (int i = 0; i < n; i++) {
        if(st.empty()){
            ans[i] = -1;
        }
        while (!st.empty() && arr[st.top()] > arr[i]) {
            st.pop();
        }
        ans[i] = st.empty() ? -1 : st.top();
        st.push(i);
    }
    return ans;
}
int sumSubarrayMins(vector<int>& arr) {
    int n = arr.size();
    vector<int> nse = nextSmallerElement(arr, n);
    vector<int> pse = leftSmaller(arr, n);
    long long sum = 0;
    int mod = 1e9 + 7;
    for (int i = 0; i < n; i++) {
        long long left = i - pse[i];
        long long right = nse[i] - i;
        long long totalWays = left*right;
        long long totalSum = arr[i]*totalWays;
        sum = (sum + totalSum) % mod;
    }
    return sum;
}
```

27) Maximal Rectangle

09 November 2024 02:56

Given a rows x cols binary matrix filled with 0's and 1's, find the largest rectangle containing only 1's and return its area.

From <https://leetcode.com/problems/maximal-rectangle/description/>

```
vector<int> nextSmaller(vector<int>& heights) {
    int n = heights.size();
    vector<int> ans(n);
    stack<int> st;
    for (int i = n - 1; i >= 0; i--) {
        while (!st.empty() && heights[st.top()] >= heights[i])
            st.pop();
        ans[i] = st.empty() ? n : st.top();
        st.push(i);
    }
    return ans;
}
vector<int> prevSmaller(vector<int>& heights) {
    int n = heights.size();
    vector<int> ans(n);
    stack<int> st;
    for (int i = 0; i < n; i++) {
        while (!st.empty() && heights[st.top()] >= heights[i])
            st.pop();
        ans[i] = st.empty() ? -1 : st.top();
        st.push(i);
    }
    return ans;
}
int largestRectangleArea(vector<int>& heights) {
    int n = heights.size();
    vector<int> next = nextSmaller(heights);
    vector<int> prev = prevSmaller(heights);
    int maxi = 0;
    for (int i = 0; i < n; i++) {
        maxi = max(maxi, heights[i] * (next[i] - prev[i] - 1)) ;
    }
    return maxi;
}
int maximalRectangle(vector<vector<char>>& matrix) {
    int n = matrix.size();
    int m = matrix[0].size();
    vector<vector<int>>v(n, vector<int>(m));
    for(int j = 0; j < m; j++){
        int sum = 0;
        for(int i = 0; i < n; i++){
            sum += matrix[i][j] - '0';
            if(matrix[i][j] == '0') sum = 0;
            v[i][j] = sum;
        }
    }
    int maxi = 0;
    for(int i = 0; i <n ;i++){
        maxi = max(maxi, largestRectangleArea(v[i]));
    }
    return maxi;
}
```

From <https://leetcode.com/problems/maximal-rectangle/submissions/1220924748/>

28) Online stock span

09 November 2024 14:47

Design an algorithm that collects daily price quotes for some stock and returns **the span** of that stock's price for the current day.
The **span** of the stock's price in one day is the maximum number of consecutive days (starting from that day and going backward) for which the stock price was less than or equal to the price of that day.

From <https://leetcode.com/problems/online-stock-span/description/>

```
stack<pair<int, int>>st;
StockSpanner() {

}

int next(int price) {
    int span = 1;
    while(!st.empty() && st.top().first <= price){
        span += st.top().second;
        st.pop();
    }
    st.push({price, span});
    return span;
}
```

From <https://leetcode.com/problems/online-stock-span/submissions/1187950704/>

29) Sum of subarray ranges

09 November 2024 15:28

You are given an integer array nums. The **range** of a subarray of nums is the difference between the largest and smallest element in the subarray.
Return *the* ***sum of all*** *subarray ranges of* nums.
A subarray is a contiguous **non-empty** sequence of elements within an array.

From <https://leetcode.com/problems/sum-of-subarray-ranges/description/>

```
vector<int> nextSmallerElement(vector<int> &arr, int n) {
    vector<int> ans(n);
    stack<int> st;
    for (int i = n - 1; i >= 0; i--) {
        if(st.empty()){
            ans[i] = n;
        }
        while (!st.empty() && arr[st.top()] >= arr[i]) {
            st.pop();
        }
        ans[i] = st.empty() ? n : st.top();
        st.push(i);
    }
    return ans;
}
vector<int> leftSmaller(vector<int> &arr, int n) {
    vector<int> ans(n);
    stack<int> st;
    for (int i = 0; i < n; i++) {
        if(st.empty()){
            ans[i] = -1;
        }
        while (!st.empty() && arr[st.top()] > arr[i]) {
            st.pop();
        }
        ans[i] = st.empty() ? -1 : st.top();
        st.push(i);
    }
    return ans;
}
long long sumSubarrayMins(vector<int>& arr) {
    int n = arr.size();
    vector<int> nse = nextSmallerElement(arr, n);
    vector<int> pse = leftSmaller(arr, n);
    long long sum = 0;
    for (int i = 0; i < n; i++) {
        long long left = i - pse[i];
        long long right = nse[i] - i;
        long long totalWays = left*right;
        long long totalSum = arr[i]*totalWays;
        sum = (sum + totalSum);
    }
    return sum;
}
vector<int> nextLargerElement(vector<int> &arr, int n) {
    vector<int> ans(n);
    stack<int> st;
    for (int i = n - 1; i >= 0; i--) {
        if(st.empty()){
            ans[i] = n;
        }
        while (!st.empty() && arr[st.top()] <= arr[i]) {
            st.pop();
        }
        ans[i] = st.empty() ? n : st.top();
```

```
            st.push(i);
        }
        return ans;
    }
    vector<int> leftLarger(vector<int> &arr, int n) {
        vector<int> ans(n);
        stack<int> st;
        for (int i = 0; i < n; i++) {
            if(st.empty()){
                ans[i] = -1;
            }
            while (!st.empty() && arr[st.top()] < arr[i]) {
                st.pop();
            }
            ans[i] = st.empty() ? -1 : st.top();
            st.push(i);
        }
        return ans;
    }
    long long sumSubarrayMaxs(vector<int>& arr) {
        int n = arr.size();
        vector<int> nse = nextLargerElement(arr, n);
        vector<int> pse = leftLarger(arr, n);
        long long sum = 0;
        for (int i = 0; i < n; i++) {
            long long left = i - pse[i];
            long long right = nse[i] - i;
            long long totalWays = left*right;
            long long totalSum = arr[i]*totalWays;
            sum = (sum + totalSum);
        }
        return sum;
    }
    long long subArrayRanges(vector<int>& nums) {
        return sumSubarrayMaxs(nums) - sumSubarrayMins(nums);
    }
```

From <https://leetcode.com/problems/sum-of-subarray-ranges/submissions/1447480574/>

30) Celebrity Problem

09 November 2024 15:56

A celebrity is a person who is known to all but **does not know** anyone at a party. A party is being organized by some people. A square matrix **mat** (n*n) is used to represent people at the party such that if an element of row i and column j is set to 1 it means ith person knows jth person. You need to return the index of the celebrity in the party, if the celebrity does not exist, return **-1**.

From <https://www.geeksforgeeks.org/problems/the-celebrity-problem/1>

```
int celebrity(vector<vector<int> >& M, int n)
  {
     stack <int> st;

     //step 1: push all persons into stack
     for (int i = 0; i < n; i++){
        st.push(i);
     }

     //step 2: run discard method to get a might be celebrity
     while(st.size() != 1){
        int a = st.top();
        st.pop();
        int b = st.top();
        st.pop();

        //if a knows b?
        if(M[a][b]){
           //a isn't celebrity, b might be
           st.push(b);
        }
        else{
           st.push(a);
        }
     }

     //check that single person is actually a celebrity
     int mightBeCelebrity = st.top();
     st.pop();

     //celebrity should not know anyone
     for (int i = 0; i < n; i++){
        if(M[mightBeCelebrity][i] != 0)
        return -1;
     }
     //everyone should know celebrity
     for (int i = 0; i < n; i++){
        if(M[i][mightBeCelebrity] == 0 && i != mightBeCelebrity)
        return -1;
     }
     //mightBeCelebrity is the cell
     return mightBeCelebrity;
  }
```

From <https://www.geeksforgeeks.org/problems/the-celebrity-problem/1>

31) 132 pattern

09 November 2024 16:39

Given an array of n integers nums, a **132 pattern** is a subsequence of three integers nums[i], nums[j] and nums[k] such that i < j < k and nums[i] < nums[k] < nums[j]. Return true *if there is a **132 pattern** in* nums*, otherwise, return* false.

From <https://leetcode.com/problems/132-pattern/description/>

```
bool find132pattern(vector<int>& nums) {
    int n = nums.size();
    stack<int> st;
    int nums3 = INT_MIN;
    for(int i = n-1; i >= 0; i--){
        if(nums3 > nums[i]) return true;
        while(!st.empty() && st.top() < nums[i]){
            nums3 = st.top();st.pop();
        }
        st.push(nums[i]);
    }
    return false;
}
```

32) Basic calculator

09 November 2024 20:37

Given a string s representing a valid expression, implement a basic calculator to evaluate it, and return *the result of the evaluation*.

From <https://leetcode.com/problems/basic-calculator/description/>

```
int calculate(string s) {
    int n = s.length();
    int number = 0;
    int result = 0;
    int sign = 1;
    stack<int> st;
    for(int i = 0; i < n; i++){
        if(isdigit(s[i])){
            number = number*10 + (s[i] - '0');
        }
        else if(s[i] == '+'){
            result += number*sign;
            number = 0;
            sign = 1;
        }

        else if(s[i] == '-'){
            result += number*sign;
            number = 0;
            sign = -1;
        }
        else if(s[i] == '('){
            st.push(result);
            st.push(sign);
            number = 0;
            result = 0;
            sign = 1;
        }

        else if(s[i] == ')'){
            result += number*sign;
            number = 0;
            int stack_sign = st.top() ;st.pop();
            int last_result = st.top() ;st.pop();
            result *= stack_sign;
            result += last_result;
        }
    }
    result += number*sign;
    return result;
}
```

33) Basic calculator II

09 November 2024 20:37

Given a string s which represents an expression, *evaluate this expression and return its value*.
The integer division should truncate toward zero.

From <https://leetcode.com/problems/basic-calculator-ii/description/>

```
int calculate(string s) {
    stack<int> st;
    int num = 0;
    char prevOperator = '+';
    for (int i = 0; i <= s.length(); i++) {
        char ch = (i < s.length()) ? s[i] : '\0';
        if (isdigit(ch)) {
            num = num * 10 + (ch - '0');
        }
        if ((!isdigit(ch) && ch != ' ') || i == s.length()) {
            if (prevOperator == '+') st.push(num);
            if (prevOperator == '-') st.push(-num);
            if (prevOperator == '*') {
                int temp = st.top() * num;
                st.pop();
                st.push(temp);
            }
            if (prevOperator == '/') {
                int temp = st.top() / num;
                st.pop();
                st.push(temp);
            }
            prevOperator = ch;
            num = 0;
        }
    }
    int result = 0;
    while (!st.empty()) {
        result += st.top();
        st.pop();
    }
    return result;
}
```

34) Help classmate (based on problem 18)

19 March 2025 19:24

Professor X wants his students to help each other in the chemistry lab. He suggests that every student should help out a classmate who scored less marks than him in chemistry and whose roll number appears after him. But the students are lazy and they don't want to search too far. They each pick the first roll number after them that fits the criteria. Find the marks of the classmate that each student picks.
Note: one student may be selected by multiple classmates.

From <https://www.geeksforgeeks.org/problems/help-classmates--141631/0>

```
vector<int> help_classmate(vector<int> prices, int n) {
    stack<int> st;
    vector<int> result(n);

    for (int i = n - 1; i >= 0; i--) {
        while (!st.empty() && st.top() >= prices[i]) {
            st.pop();
        }

        result[i] = (!st.empty()) ? st.top() : -1; // Store the next smaller element or -1 if none exists

        st.push(prices[i]);
    }
    return result;
}
```