

**VISVESVARAYA TECHNOLOGICAL
UNIVERSITY**

"JnanaSangama" Belgaum -590014, Karnataka.



**DATA STRUCTURES AAT REPORT
on
CODING CHALLENGES**

Submitted by:

VIJAYALAKSHMI BHARATESH(1BM25CS456-T)

Under the Guidance of

Prof .Manjula S

Assistant Professor ,BMSCE

in partial fulfilment for the award of the degree of
BACHELOR OF ENGINEERING

**In
COMPUTER SCIENCE AND ENGINEERING**



B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

August to December 2025

**B.M.S COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

Coding Challenge 1 :

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.

Example 1:

Input

```
["MinStack","push","push","push","getMin","pop","top","get"]  
[[],[-2],[0],[-3],[,],[,],[,],[,]]
```

Output :

```
[null,null,null,null,-3,null,0,-2]
```

Explanation :

```
MinStack minStack = new MinStack();  
minStack.push(-2);  
minStack.push(0);  
minStack.push(-3);  
minStack.getMin(); // return -3  
minStack.pop();  
minStack.top();    // return 0  
minStack.getMin(); // return -2
```

Constraints:

- $-2^{31} \leq \text{val} \leq 2^{31} - 1$
- Methods pop, top and getMin operations will always be called on non-empty stacks.
- At most $3 * 10^4$ calls will be made to push, pop, top, and getMin.

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

Coding Challenge 2 :

You are given an array of k linked-lists lists, each linked-list is sorted in ascending order.

Merge all the linked-lists into one sorted linked-list and return it.

Example 1:

Input: lists = [[1,4,5],[1,3,4],[2,6]]

Output: [1,1,2,3,4,4,5,6]

Explanation: The linked-lists are:

```
[
  1->4->5,
  1->3->4,
  2->6
]
```

merging them into one sorted linked list:

```
1->1->2->3->4->4->5->6
```

Example 2:

Input: lists = []

Output: []

Example 3:

Input: lists = [[]]

Output: []

Constraints:

- $k == \text{lists.length}$
- $0 \leq k \leq 10^4$
- $0 \leq \text{lists}[i].\text{length} \leq 500$
- $-10^4 \leq \text{lists}[i][j] \leq 10^4$
- $\text{lists}[i]$ is sorted in **ascending order**.
- The sum of $\text{lists}[i].\text{length}$ will not exceed 10^4 .

Leetcode Link: <https://leetcode.com/problems/merge-k-sorted-lists/description/>

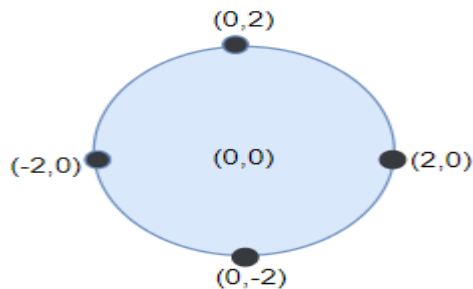
Coding Challenge 3 :

Alice is throwing n darts on a very large wall. You are given an array `darts` where `darts[i] = [xi, yi]` is the position of the i^{th} dart that Alice threw on the wall.

Bob knows the positions of the n darts on the wall. He wants to place a dartboard of radius r on the wall so that the maximum number of darts that Alice throws lie on the dartboard.

Given the integer r , return *the maximum number of darts that can lie on the dartboard*.

Example 1:

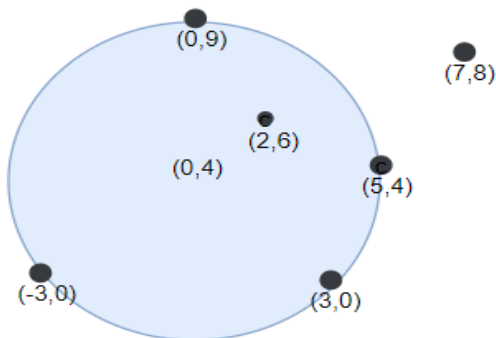


Input: `darts = [[-2,0],[2,0],[0,2],[0,-2]]`, $r = 2$

Output: 4

Explanation: Circle dartboard with center in (0,0) and radius = 2 contain all points.

Example 2:



Input: `darts = [[-3,0],[3,0],[2,6],[5,4],[0,9],[7,8]]`, $r = 5$

Output: 5

Explanation: Circle dartboard with center in (0,4) and radius = 5 contain all points except the point (7,8).

Constraints:

- $1 \leq \text{darts.length} \leq 100$
- `darts[i].length == 2`
- $-10^4 \leq x_i, y_i \leq 10^4$
- All the darts are unique
- $1 \leq r \leq 5000$

Leetcode Link: <https://leetcode.com/problems/maximum-number-of-darts-inside-of-a-circular-dartboard/description/>

Coding Challenge 4:

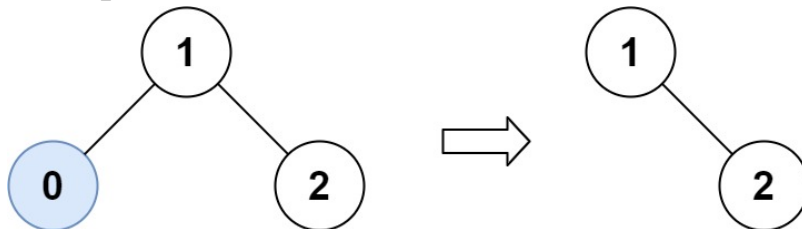
Given the root of a binary search tree and the lowest and highest boundaries as low and high, trim the tree so that all its elements lies in [low, high].

Trimming the tree should **not** change the relative structure of the elements that will remain in the tree (i.e., any node's descendant should remain a descendant).

It can be proven that there is a **unique answer**.

Return *the root of the trimmed binary search tree*. Note that the root may change depending on the given bounds.

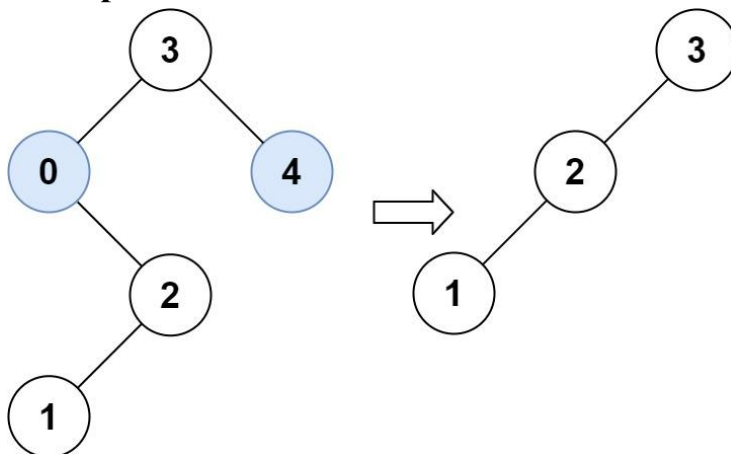
Example 1:



Input: root = [1,0,2], low = 1, high = 2

Output: [1,null,2]

Example 2:



Input: root = [3,0,4,null,2,null,null,1], low = 1, high = 3

Output: [3,2,null,1]

Constraints:

- The number of nodes in the tree is in the range [1, 10⁴].
- 0 ≤ Node.val ≤ 10⁴
- The value of each node in the tree is **unique**.
- 0 ≤ low ≤ high ≤ 10⁴

Leetcode link : <https://leetcode.com/problems/trim-a-binary-search-tree/description/>

Coding Challenge 5 :

An array is **squareful** if the sum of every pair of adjacent elements is a **perfect square**.

Given an integer array `nums`, return *the number of permutations of `nums` that are squareful*.

Two permutations `perm1` and `perm2` are different if there is some index `i` such that `perm1[i] != perm2[i]`.

Example 1:

Input: `nums = [1,17,8]`

Output: 2

Explanation: `[1,8,17]` and `[17,8,1]` are the valid permutations.

Example 2:

Input: `nums = [2,2,2]`

Output: 1

Constraints:

- $1 \leq \text{nums.length} \leq 12$
- $0 \leq \text{nums}[i] \leq 10^9$

Leetcode Link : <https://leetcode.com/problems/number-of-squareful-arrays/description/>

Challenge 1:

Code

Testcase Test Result

Case 1 +

```
["MinStack","push","push","push","getMin","pop","top","getMin"]
```

```
[[],[-2],[0],[-3],[],[],[],[[]]]
```

VijayalakshmiP

Access all features with our Premium subscription!

My Lists

Notebook

Progress

Points

Try New Features

Orders

My Playgrounds

Settings

Appearance

Sign Out

Code

Testcase Test Result

Accepted Runtime: 0 ms

Case 1

Input

```
["MinStack","push","push","push","getMin","pop","top","getMin"]
```

```
[[],[-2],[0],[-3],[],[],[],[[]]]
```

Output

```
[null,null,null,null,-3,null,0,-2]
```

Expected

```
[null,null,null,null,-3,null,0,-2]
```

Contribute a testcase

VijayalakshmiP

Access all features with our Premium subscription!

My Lists

Notebook

Progress

Points

Try New Features

Orders

My Playgrounds

Settings

Appearance

Sign Out

Challenge 2 :

`</> Code`

☒ Testcase | `>_ Test Result`

Accepted Runtime: 0 ms

☒ Case 1 ☒ Case 2 ☒ Case 3

Input


lists =
[]


Output


[]


Expected


[]


 **VijayalakshmiP**
Access all features with our Premium subscription!


 My Lists


 Notebook


 Progress

 Points

 Try New Features

 Orders

 My Playgrounds

 Settings

`</> Code`

☒ Testcase | `>_ Test Result`

Accepted Runtime: 0 ms

☒ Case 1 ☒ Case 2 ☒ Case 3

Input


lists =
[[]]


Output


[]


Expected


[]


 **VijayalakshmiP**
Access all features with our Premium subscription!


 My Lists


 Notebook


 Progress

 Points

 Try New Features

 Orders

 My Playgrounds

 Settings

`</> Code`

☒ Testcase | `>_ Test Result`

Accepted Runtime: 0 ms

☒ Case 1 ☒ Case 2 ☒ Case 3

Input


lists =
[[1,4,5],[1,3,4],[2,6]]


Output


[1,1,2,3,4,4,5,6]


Expected


[1,1,2,3,4,4,5,6]


 **VijayalakshmiP**
Access all features with our Premium subscription!


 My Lists


 Notebook


 Progress

 Points

 Try New Features

 Orders

 My Playgrounds

 Settings

Challenge 3:

 Code

☒ Testcase  Test Result

Accepted Runtime: 0 ms

☒ Case 1 ☒ Case 2

Input

darts =
[[-2,0],[2,0],[0,2],[0,-2]]

r =
2

Output

4

Expected

4

 Code

☒ Testcase  Test Result

Accepted Runtime: 0 ms

☒ Case 1 ☒ Case 2

Input

darts =
[[-3,0],[3,0],[2,6],[5,4],[0,9],[7,8]]

r =
5

Output

5

Expected

5



VijayalakshmiP

Access all features with our
Premium subscription!



My Lists



Notebook



Progress



Points



Try New Features



Orders



My Playgrounds



Settings



Appearance



Sign Out



VijayalakshmiP

Access all features with our
Premium subscription!



My Lists



Notebook



Progress



Points



Try New Features



Orders



My Playgrounds



Settings



Appearance



Sign Out

Challenge 4 :

</> Code

Testcase

Test Result

Accepted Runtime: 0 ms

Case 1

Case 2

Input

root =
[3,0,4,null,2,null,null,1]

low =
1

high =
3

Output

[3,2,null,1]

Expected

[3,2,null,1]

VijayalakshmiP

Access all features with our Premium subscription!

My Lists

Notebook

Progress

Points

Try New Features

Orders

My Playgrounds

Settings

Appearance

Sign Out

</> Code

Testcase

Test Result

Accepted Runtime: 0 ms

Case 1

Case 2

Input

root =
[1,0,2]

low =
1

high =
2

Output

[1,null,2]

Expected

[1,null,2]

VijayalakshmiP

Access all features with our Premium subscription!

My Lists

Notebook

Progress

Points

Try New Features

Orders

My Playgrounds

Settings

Appearance

Sign Out

Challenge 5:

</> Code

☒ Testcase | [Test Result](#)

Accepted Runtime: 0 ms

☒ Case 1 ☒ Case 2

Input

```
nums =  
[2,2,2]
```

Output

1

Expected

1

</> Code

☒ Testcase | [Test Result](#)

Accepted Runtime: 0 ms

☒ Case 1 ☒ Case 2

Input

```
nums =  
[1,17,8]
```

Output

2

Expected

2



VijayalakshmiP

Access all features with our
Premium subscription!



My Lists



Notebook



Progress



Points



Try New Features



Orders



My Playgrounds



Settings



VijayalakshmiP

Access all features with our
Premium subscription!



My Lists



Notebook



Progress



Points



Try New Features



Orders



My Playgrounds



Settings

1 . Min Stack

```
typedef struct {  
    int s[30000];  
    int min[30000];  
    int top;  
} MinStack;
```

```
MinStack* minStackCreate() {  
    MinStack* obj = (MinStack*)malloc(sizeof(MinStack));  
    obj->top = -1;  
    return obj;  
}
```

```
void minStackPush(MinStack* obj, int val) {  
    obj->top++;  
    obj->s[obj->top] = val;  
    if (obj->top == 0)  
        obj->min[obj->top] = val;  
    else  
        obj->min[obj->top] = val < obj->min[obj->top - 1] ? val : obj->min[obj->top - 1];  
}
```

```
void minStackPop(MinStack* obj) {  
    obj->top--;  
}
```

```
int minStackTop(MinStack* obj) {  
    return obj->s[obj->top];  
}
```

```
int minStackGetMin(MinStack* obj) {  
    return obj->min[obj->top];  
}
```

```
void minStackFree(MinStack* obj) {  
    free(obj);  
}
```

2. Merge k Sorted List

```
struct ListNode* mergeTwo(struct ListNode* a, struct ListNode* b) {  
    struct ListNode dummy;  
    struct ListNode* t = &dummy;  
    dummy.next = NULL;  
    while (a && b) {  
        if (a->val < b->val) {  
            t->next = a;  
            a = a->next;  
        } else {  
            t->next = b;  
            b = b->next;  
        }  
        t = t->next;  
    }  
    t->next = a ? a : b;  
    return dummy.next;  
}
```

```
struct ListNode* mergeKLists(struct ListNode** lists, int listsSize) {  
    if (listsSize == 0) return NULL;  
    int interval = 1;  
    while (interval < listsSize) {  
        for (int i = 0; i + interval < listsSize; i += interval * 2) {  
            lists[i] = mergeTwo(lists[i], lists[i + interval]);  
        }  
        interval *= 2;  
    }  
    return lists[0];  
}
```

3. Maximum Number of Darts inside of circular Dartboard

```
#include <math.h>

int numPoints(int** darts, int dartsSize, int* dartsColSize, int r) {
    if (dartsSize == 0) return 0;
    int ans = 1;
    double R = (double)r;
    for (int i = 0; i < dartsSize; i++) {
        for (int j = i + 1; j < dartsSize; j++) {
            double x1 = darts[i][0], y1 = darts[i][1];
            double x2 = darts[j][0], y2 = darts[j][1];
            double dx = x2 - x1, dy = y2 - y1;
            double d = sqrt(dx * dx + dy * dy);
            if (d > 2 * R) continue;
            double mx = (x1 + x2) / 2.0;
            double my = (y1 + y2) / 2.0;
            double h = sqrt(R * R - (d / 2) * (d / 2));
            double ux = -dy / d;
            double uy = dx / d;

            double cx1 = mx + ux * h;
            double cy1 = my + uy * h;
            double cx2 = mx - ux * h;
            double cy2 = my - uy * h;

            int c1 = 0, c2 = 0;
            for (int k = 0; k < dartsSize; k++) {
                double px = darts[k][0], py = darts[k][1];
                if ((px - cx1) * (px - cx1) + (py - cy1) * (py - cy1) <= R * R + 1e-6)
                    c1++;
                if ((px - cx2) * (px - cx2) + (py - cy2) * (py - cy2) <= R * R + 1e-6)
                    c2++;
            }
            if (c1 > ans) ans = c1;
            if (c2 > ans) ans = c2;
        }
    }
    return ans;
}
```

4. Trim A Binary Search Tree

```
struct TreeNode* trimBST(struct TreeNode* root, int low, int high) {  
    if (root == NULL) return NULL;  
    if (root->val < low) return trimBST(root->right, low, high);  
    if (root->val > high) return trimBST(root->left, low, high);  
    root->left = trimBST(root->left, low, high);  
    root->right = trimBST(root->right, low, high);  
    return root;  
}
```

5. Number Of Squareful Arrays

```
#include <math.h>
```

```
int isSquare(int x) {  
    int r = (int)(sqrt(x) + 0.5);  
    return r * r == x;  
}
```

```
void dfs(int* nums, int numsSize, int* used, int prev, int depth, int* count) {  
    if (depth == numsSize) {  
        (*count)++;  
        return;  
    }  
    for (int i = 0; i < numsSize; i++) {  
        if (used[i]) continue;  
        if (i > 0 && nums[i] == nums[i - 1] && !used[i - 1]) continue;  
        if (prev != -1 && !isSquare(prev + nums[i])) continue;  
        used[i] = 1;  
        dfs(nums, numsSize, used, nums[i], depth + 1, count);  
        used[i] = 0;  
    }  
}
```

```
int cmp(const void* a, const void* b) {  
    return (*(int*)a - *(int*)b);  
}
```

```
int numSquarefulPerms(int* nums, int numsSize) {  
    qsort(nums, numsSize, sizeof(int), cmp);  
    int used[12] = {0};  
    int count = 0;  
    dfs(nums, numsSize, used, -1, 0, &count);  
    return count;  
}
```

**B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering**



This is to certify that the Lab work entitled “**DATA STRUCTURES**” carried out by **VIJAYALAKSHMI PADANAD (1BMCS25456-T)**, who is bonafide student of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2025-2026. The Lab report has been approved as it satisfies the academic requirements in respect of Data structures Lab - (**23CS3PCDST**) work prescribed for the said degree.

Mrs. Manjula S
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Kavitha Sooda
Professor and Head
Department of CSE
BMSCE, Bengaluru