Programming Club IITM

CPS-2

progclubiitm.com

Programming Club IITM

TIME COMPLEXITY

progclubiitm.com

You have coded a solution to a problem. How do you decide if your code will finish running in time?

Programming Club IITM

How many operations does a computer perform when this code is run?

```
int sum = 0;
for(int i = 0; i < n; ++i){
    sum += i;
}
```

1 assignment operation

1 assignment operation

```
int sum = 0;
for(int i = 0; i < n; ++i){
    sum += i;
}
```

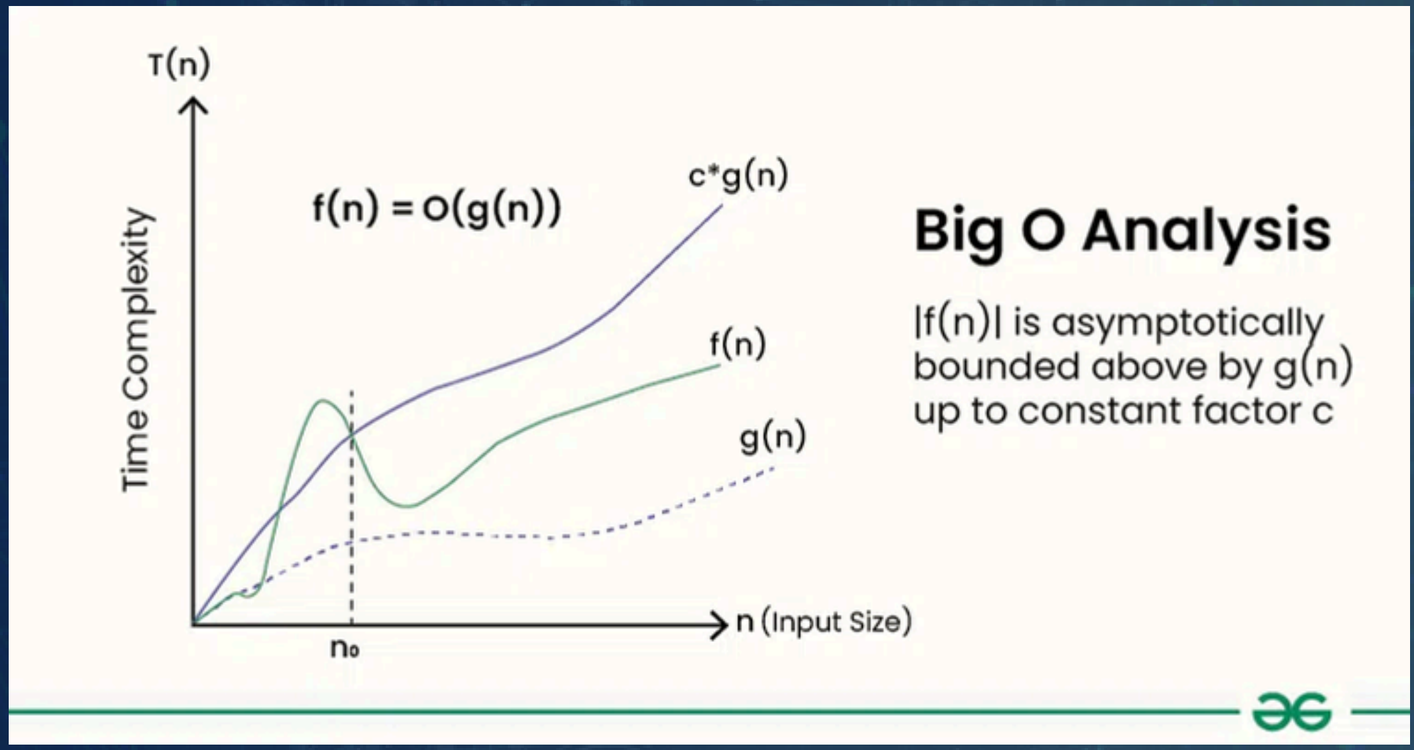
n additions +
n assignments

n+1 comparison
operations

n increment operations

Total : 4n + 3 operations

Big O notation



Given two functions f(n) and g(n), we say that f(n) is O(g(n)) if there exist constants c > 0 and n_0 >= 0 such that f(n) <= c*g(n) for all n >= n_0.

Programming Club IITM

Big O notation

Big O is used as a quick estimate to the amount of time an algorithm takes.

Here are a few examples:

- 4n + 3 operations = O(n)
- 5 operations = O(1)
- $0.5n^2 + 2n$ operations = $O(n^2)$
- $2^n + n! + n^4$ operations = O(n!)

Its basically the largest term as n scales.

Relation between time complexity of expected solution and problem constraints: (CF Judge can do around 1e8 operations/sec)

Let n be the main variable in the problem.

- If $n \leq 12$, the time complexity can be $O(n!)$.
- If $n \leq 25$, the time complexity can be $O(2^n)$.
- If $n \leq 100$, the time complexity can be $O(n^4)$.
- If $n \leq 500$, the time complexity can be $O(n^3)$.
- If $n \leq 10^4$, the time complexity can be $O(n^2)$.
- If $n \leq 10^6$, the time complexity can be $O(n \log n)$.
- If $n \leq 10^8$, the time complexity can be $O(n)$.
- If $n > 10^8$, the time complexity can be $O(\log n)$ or $O(1)$.

Programming Club IITM

Chocolate Question!

What's the time complexity?

```
for (int i = n; i > 0; i /= 2){
    for (int j = 0; j < i; j++){
        // Some O(1) operations
    }
}
```

Solution

```
for (int i = n; i > 0; i /= 2){
    for (int j = 0; j < i; j++){
        // Some O(1) operations
    }
}
```

i = n : j = 1, 2, 3, ... n → n ops
i = n/2 : j = 1, 2, 3, ... n/2 → [n/2] ops
i = n/4 : j = 1, 2, 3, ... n/4 → [n/4] ops
and so on. Summing them up,
total = n (1 + ½ + ¼ + ⅛ ...) ops
< 2n ops

Answer : O (n)

Programming Club IITM

Double Chocolate Question!

What's the time complexity?

```
for (int i = 1; i <= n; ++i) {
    for (int j = i; j <= n; j+=i) {
        // Some O(1) operations
    }
}
```

Solution

```
for (int i = 1; i <= n; ++i) {
    for (int j = i; j <= n; j+=i) {
        // Some O(1) operations
    }
}
```

i = 1 : j = 1, 2, 3, ... n → n ops
i = 2 : j = 2, 4, 6, ... n → [n/2] ops
i = 3 : j = 3, 6, 9, ... n → [n/3] ops
and so on. Summing them up,
total = n (1 + ½ + ⅓ + ... 1/n) ops

$$1 + \frac{1}{2} + \frac{1}{3} + \ldots \frac{1}{n} < \ln(n) + \text{constant}$$

Answer : O (n log n)

Programming Club IITM

PREFIX SUMS

progclubiitm.com

Try this out!

Given an array of ‘n’ integers, find the sum of all the elements from (one-based) index ‘l’ to ‘r’ (both inclusive)

Solution

Just iterate through the required indices to get the answer

```
int ans = 0;
for (int i = l; i <= r; ++i) {
    ans += a[i];
}
cout << ans << '\n';
```

Time complexity?

Now try this!

Given an array of ‘n’ integers and ‘q’ queries of the form “ l r ”, find the sum of all the elements from index ‘l’ to ‘r’ (both inclusive) for each query

Solution

Just iterate through the required indices to get the answer for each query

```
for (int qi = 0; qi < q; ++i) {
    int l, r;
    cin >> l >> r;
    int ans = 0;
    for (int i = l; i <= r; ++i) {
        ans += a[i];
    }
    cout << ans << '\n';
}
```

Time complexity?

How can we speed it up?
Can we reuse previous work?

Programming Club IITM

Prefix Sum Arrays

Definition

We are given an array ‘a’ of ‘n’ elements

The ‘i’th element of the prefix sum array of ‘a’ is given by

pref[i] = a[1] + a[2] + ... + a[i] for 1 <= i <= n

How does it help?

We need to find

a[l] + a[l + 1] + ... + a[r - 1] + a[r]

This can smartly be rewritten as

(a[1] + a[2] + ... a[r-1] + a[r]) - (a[1] + a[2] + ... a[l-1])

= pref[r] - pref[l-1]

Time complexity?

How does it help?

So the answer to a query “ l r ”= pref[r] - pref[l-1]

which is O(1) per query, much better than the initial O(n)

For ‘q’ queries, the the complexity is O(q)

Programming Club IITM

Building the Prefix Sum Array

Using the definition,

pref[i] = a[1] + a[2] + ... + a[i - 2] + a[i-1] +a[i]

pref[i-1] = a[1] + a[2] + ... + a[i-2] + a[i-1]

Subtracting both equations, we get

pref[i] = pref[i-1] + a[i]

```
int pref[n+1];
pref[0] = 0;
for (int i = 1; i <= n; ++i) {
    pref[i] = pref[i-1] + a[i];
}
```

Time complexity?

progclubiitm.com

Back to the original probem

Now that we know the answer to a query “ l r ”= pref[r+1] - pref[l]

and we’ve built the prefix array, here is the final code

```
int n, q;
cin >> n >> q;

vector<int> a(n+1);
for (int i = 1; i <= n; ++i) {
    cin >> a[i];
}

vector<int> pref(n + 1);
pref[0] = 0;
for (int i = 0; i < n; ++i) pref[i] = pref[i-1] + a[i];

for (int i = 0; i < q; ++i) {
    int l, r;
    cin >> l >> r;
    cout << pref[r] - pref[l-1] << '\n';
}
```

Common Errors

- Allocate size ‘n+1’ and not ‘n’ to the prefix array
- Don’t forget to inititalize pref[0] = 0
- Pay attention to the query ranges, for e.g. [l, r] [l, r) (l, r] (l, r)
- Check whether l and r are zero-indexed or one-indexed

Programming Club IITM

Let's solve one more problem

https://codeforces.com/problemset/problem/18/C

C. Stripe

time limit per test: 2 seconds
memory limit per test: 64 megabytes

Once Bob took a paper stripe of n squares (the height of the stripe is 1 square). In each square he wrote an integer number, possibly negative. He became interested in how many ways exist to cut this stripe into two pieces so that the sum of numbers from one piece is equal to the sum of numbers from the other piece, and each piece contains positive integer amount of squares. Would you help Bob solve this problem?

Input

The first input line contains integer n ($1 \leq n \leq 10^5$) — amount of squares in the stripe. The second line contains n space-separated numbers — they are the numbers written in the squares of the stripe. These numbers are integer and do not exceed 10000 in absolute value.

Output

Output the amount of ways to cut the stripe into two non-empty pieces so that the sum of numbers from one piece is equal to the sum of numbers from the other piece. Don't forget that it's allowed to cut the stripe along the squares' borders only.

Approach

Say the sum of all the elements is ‘S’.

Now, the sum of the elements in the left piece is equal to a prefix sum ‘p’.

So the sum of the elements in the right piece = S - p

According to the question, p = S - p

So p = S/2

Programming Club IITM

Approach

Now iterate through the prefix array and count the number of occurences of S/2

Let’s code it!

Code

```cpp
int main() {
    int n;
    cin >> n;

    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    vector<int> pref(n + 1);
    pref[0] = 0;
    for (int i = 1; i <= n; i++) {
        pref[i] = pref[i - 1] + a[i];
    }

    int s = pref[n];
    if (((s % 2) + 2) % 2 == 1) {
        cout << 0 << '\n';
    } else {
        int count = 0;
        for (int i = 1; i < n; i++) {
            if (pref[i] == s / 2) {
                count++;
            }
        }
        cout << count << '\n';
    }

    return 0;
}
```

Extensions of Prefix sum

Like addition, the idea of prefix array can be extended to other operations as well. For example,

- Multiplication : pref[r+1]/pref[l] (beware of overflow)
- Bitwise XOR: pref[r+1] ^ pref[l]

Programming Club IITM

BINARY SEARCH

progclubiitm.com

A SMALL PROBLLEM

Given a sorted list of n elements, find the position of a target or where it should be inserted to keep the list sorted.

HOW TO DO IT?

In the question, assume there are 10 elements in the array and we need to find an element = 170

HOW TO DO IT?

Step 1:
Take a sorted array and find middle element.

0	1	2	3	4	5	6	7	8	9
154	156	157	163	165	168	171	175	180	188

HOW TO DO IT?

Step 2:
Compare middle element with target value (Key)

0	1	2	3	4	5	6	7	8	9
154	156	157	163	165	168	171	175	180	188

HOW TO DO IT?

Step 3:
Update search range and repeat

0	1	2	3	4	5	6	7	8	9
154	156	157	163	165	168	171	175	180	188

HOW TO DO IT?

Step 3:
Update search range and repeat

0	1	2	3	4	5	6	7	8	9
154	156	157	163	165	168	171	175	180	188

HOW TO DO IT?

Step 3:
Update search range and repeat

0	1	2	3	4	5	6	7	8	9
154	156	157	163	165	168	171	175	180	188

HOW TO DO IT?

Step 3:
Update search range and repeat

0	1	2	3	4	5	6	7	8	9
154	156	157	163	165	168	171	175	180	188

HOW TO DO IT?

Step 3:
Update search range and repeat

0	1	2	3	4	5	6	7	8	9
154	156	157	163	165	168	171	175	180	188

HOW TO DO IT?

Step 3:
Update search range and repeat

0	1	2	3	4	5	6	7	8	9
154	156	157	163	165	168	171	175	180	188

HOW TO DO IT?

Step 3:
Update search range and repeat

0	1	2	3	4	5	6	7	8	9
154	156	157	163	165	168	171	175	180	188

WHAT IS IT?

Binary Search is a searching algorithm that operates on a sorted or monotonic search space, repeatedly dividing it into halves to find a target value or optimal answer in logarithmic time O(log N).

CODE FOR BINARY SEARCH

```cpp
#include <bits/stdc++.h>

using namespace std;

int binary_search(vector<int> &arr, int key){
    int high=arr.size(), low=0;
    while(low<=high){
        int mid = low + (high-low)/2;
        if(arr[mid]==key){
            return mid;
        }else if(arr[mid]<key){
            low=mid+1;
        }else if(arr[mid]>key){
            high=mid-1;
        }
    }
    return -1;
}
```

TRY THESE

- https://codeforces.com/problemset/problem/1915/C
- https://codeforces.com/problemset/problem/1873/E

Programming Club IITM

STL

progclubiitm.com

WHAT IS IT?

STL is a collection of pre-built classes and functions that make it easy to manage data using common data structures like vectors, stacks, and maps. It saves time and effort by providing ready-to-use, efficient algorithms and containers.

CONTAINERS

Containers are the data structures used to store objects and data according to the requirement. Each container is implemented as a template class that also contains the methods to perform basic operations on it. Every STL container is defined inside its own header file.

WHAT ALL DOES IT INCLUDE?

- Vectors
- strings
- stacks
- Queues
- Priority Queues
- Maps
- Sets
- Multisets
- Multimaps
- Unordered sets

Many more...

ITERATORS

Iterators are the pointer like objects that are used to point to the memory addresses of STL containers. They are one of the most important components that contributes the most in connecting the STL algorithms with the containers. Iterators are defined inside the <iterator> header file.

MAPS

C++ std::map is a container, allowing you to store keys associated with values, for easy and efficient retrieval.

KEY OPERATIONS

Insertion (Put/Insert): Adding a new key-value pair to the map.

```cpp
#include <iostream>
#include <map>
using namespace std;

int main() {
    map<int, string> mp;

    // Insert elements
    mp[1] = "One";
    mp[2] = "Two";
    mp[3] = "Three";

    // Another way of inserting
    mp.insert({4, "Four"});
```

KEY OPERATIONS

- Retrieval (Get/Search): Accessing the value associated with a given key.

```
// Accessing elements
cout << "\nValue at key 3: " << mp[3] << "\n";
```

KEY OPERATIONS

- Deletion (Remove/Delete): Removing a key-value pair from the map.

```
// Erasing elements
mp.erase(3); // remove key 3
cout << "\nAfter erasing key 3:\n";
for (auto &p : mp) {
    cout << p.first << " -> " << p.second << "\n";
}
```

KEY OPERATIONS

- Update (Set/Modify): Changing the value associated with an existing key

```
// Updating value
mp[2] = "Five";
```

KEY OPERATIONS

- Checking for Existence (Contains Key/Has Key): Determining if a specific key is present in the map.

```
// Searching for a key
int key = 2;
if (mp.find(key) != mp.end()) {
    cout << "\nKey " << key << " found with value: " << mp[key] << "\n";
} else {
    cout << "\nKey " << key << " not found\n";
}
```

HOW DOES IT WORK?

Visualisation: https://visualgo.net/en/bst

https://www.geeksforgeeks.org/dsa/binary-search-tree-data-structure/

https://www.geeksforgeeks.org/dsa/introduction-to-red-black-tree/

TRY THIS

- https://codeforces.com/problemset/problem/2133/A

```cpp
#include <iostream>
#include <map>
using namespace std;

int main() {
    map<int, string> mp;

    // Insert elements
    mp[1] = "One";
    mp[2] = "Two";
    mp[3] = "Three";

    // Another way of inserting
    mp.insert({4, "Four"});

    // Accessing elements
    cout << "\nValue at key 3: " << mp[3] << "\n";

    // Searching for a key
    int key = 2;
    if (mp.find(key) != mp.end()) {
        cout << "\nKey " << key << " found with value: " << mp[key] << "\n";
    } else {
        cout << "\nKey " << key << " not found\n";
    }

    // Erasing elements
    mp.erase(3); // remove key 3
    cout << "\nAfter erasing key 3:\n";
    for (auto &p : mp) {
        cout << p.first << " -> " << p.second << "\n";
    }

    // Checking size
    cout << "\nSize of map: " << mp.size() << "\n";

    return 0;
}
```

SETS

Sets are containers which stores unique elements in some sorted order. By default, it is sorted ascending order of the keys, but this can be changed as per requirement.

KEY OPERATIONS

Insertion (Put/Insert): Adding a new element pair to the set.

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    set<int> s; // ordered set

    // Insert elements
    s.insert(10);
    s.insert(5);
    s.insert(20);
    s.insert(15);

    cout << "Elements after insertion: ";
    for (int x : s) cout << x << " ";
    cout << "\n";
```

KEY OPERATIONS

- Retrieval (Get/Search): Accessing the an element by value

```
// Find element by value
int val = 15;
auto it = s.find(val);
```

KEY OPERATIONS

- Deletion (Remove/Delete): Removing a removing a particular element from set.

```
// Delete element
s.erase(10);
```

KEY OPERATIONS

- Checking for Existence: Determining if a specific element is present in set.

```
// Find element by value
int val = 15;
auto it = s.find(val);
```

IMPLEMENTATION

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    set<int> s; // ordered set

    // Insert elements
    s.insert(10);
    s.insert(5);
    s.insert(20);
    s.insert(15);

    cout << "Elements after insertion: ";
    for (int x : s) cout << x << " ";
    cout << "\n";

    // Delete element
    s.erase(10);

    // Find largest element (last element)
    if (!s.empty()) {
        cout << "Largest element: " << *s.rbegin() << "\n";
    }

    // Find smallest element (first element)
    if (!s.empty()) {
        cout << "Smallest element: " << *s.begin() << "\n";
    }

    // Find element by value
    int val = 15;
    auto it = s.find(val);

    return 0;
}
```

TRY THIS

- https://codeforces.com/problemset/problem/1703/B

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    set<int> s; // ordered set

    // Insert elements
    s.insert(10);
    s.insert(5);
    s.insert(20);
    s.insert(15);

    cout << "Elements after insertion: ";
    for (int x : s) cout << x << " ";
    cout << "\n";

    // Delete element
    s.erase(10);

    // Find largest element (last element)
    if (!s.empty()) {
        cout << "Largest element: " << *s.rbegin() << "\n";
    }

    // Find smallest element (first element)
    if (!s.empty()) {
        cout << "Smallest element: " << *s.begin() << "\n";
    }

    // Find element by value
    int val = 15;
    auto it = s.find(val);

    return 0;
}
```

MULTIISET

Multiset is an associative container similar to the set, but it can store multiple elements with same value. It is sorted in increasing order by default, but it can be changed to any desired order.

Read more at:
https://www.geeksforgeeks.org/cpp/multiset-in-cpp-stl/

TRY THIS

https://codeforces.com/problemset/problem/1542/A

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    // Declaration
    multiset<int> ms;

    // Insertion (duplicates allowed)
    ms.insert(5);
    ms.insert(3);
    ms.insert(3);
    ms.insert(1);

    for (auto x : ms) cout << x << " ";
    cout << "\n";

    // Accessing smallest and largest
    cout << "Smallest element: " << *ms.begin() << "\n";
    cout << "Largest element: " << *ms.rbegin() << "\n";

    // Finding an element
    int val = 3;
    auto it = ms.find(val);

    // Traversing all occurrences of a value using equal_range
    auto range = ms.equal_range(3);
    cout << "All occurrences of 3: ";
    for (auto it = range.first; it != range.second; ++it)
        cout << *it << " ";
    cout << "\n";

    // Deletion
    // Erase one occurrence of 3
    auto it2 = ms.find(3);
    if (it2 != ms.end()) ms.erase(it2);

    // Erase all occurrences of 3
    ms.erase(3);

    return 0;
}
```

POLICY BASED DS

Policy Based Data Structures (PBDS) are advanced data structures provided in the GNU C++ library that extend the standard template library (STL) containers with extra functionality like order-statistics (finding the k-th smallest element) and order-of-key queries (finding the rank of a key).

Headers : <ext/pb_ds/assoc_container.hpp>, <ext/pb_ds/tree_policy.hpp>

WHY DO WE NEED THEM?

But they lack two very useful operations often required in competitive programming and algorithm design:

1. Find by order → "What is the k-th smallest element in the set?"
2. Order of key → "How many elements are strictly smaller than a given key?"

PBDS fills this gap by providing these order-statistics operations.

Read more at:
https://www.geeksforgeeks.org/cpp/policy-based-data-structures-g/

Programming Club IITM

THANK YOU!

progclubiitm.com