GeeksforGeeks
A computer science portal for geeks

GeeksQuiz

Login

Home Algorithms DS GATE Interview Corner Q&A C C++ Java Books Contribute Ask a Q About

Array Bit Magic C/C++ Articles GFacts Linked List MCQ Misc Output String Tree Graph

Measure one litre using two vessels and infinite water supply

There are two vessels of capacities 'a' and 'b' respectively. We have infinite water supply. Give an efficient algorithm to make exactly 1 litre of water in one of the vessels. You can throw all the water from any vessel any point of time. Assume that 'a' and 'b' are Coprimes.

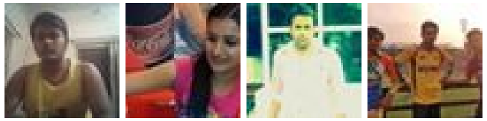
Following are the steps:
Let V1 be the vessel of capacity 'a' and V2 be the vessel of capacity 'b' and 'a' is smaller than 'b'.
1) Do following while the amount of water in V1 is not 1.
....**a)** If V1 is empty, then completely fill V1
....**b)** Transfer water from V1 to V2. If V2 becomes full, then keep the remaining water in V1 and empty V2
2) V1 will have 1 litre after termination of loop in step 1. Return.

Following is C++ implementation of the above algorithm.

```
/* Sample run of the Algo for V1 with capacity 3 and V2 with capacity
1. Fill V1:                               V1 = 3, V2 = 0
2. Transfer from V1 to V2, and fill V1:   V1 = 3, V2 = 3
2. Transfer from V1 to V2, and fill V1:   V1 = 3, V2 = 6
3. Transfer from V1 to V2, and empty V2:  V1 = 2, V2 = 0
4. Transfer from V1 to V2, and fill V1:   V1 = 3, V2 = 2
5. Transfer from V1 to V2, and fill V1:   V1 = 3, V2 = 5
6. Transfer from V1 to V2, and empty V2:  V1 = 1, V2 = 0
7. Stop as V1 now contains 1 litre.

Note that V2 was made empty in steps 3 and 6 because it became full */

#include <iostream>
```



Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```
using namespace std;

// A utility function to get GCD of two numbers
int gcd(int a, int b) { return b? gcd(b, a % b) : a; }

// Class to represent a Vessel
class Vessel
{
    // A vessel has capacity, and current amount of water in it
    int capacity, current;
public:
    // Constructor: initializes capacity as given, and current as 0
    Vessel(int capacity) { this->capacity = capacity; current = 0; }

    // The main function to fill one litre in this vessel. Capacity of
    // must be greater than this vessel and two capacities must be co-
    void makeOneLitre(Vessel &V2);

    // Fills vessel with given amount and returns the amount of water
    // transferred to it. If the vessel becomes full, then the vessel
    // is made empty.
    int transfer(int amount);
};
```

```
// The main function to fill one litre in this vessel. Capacity
// of V2 must be greater than this vessel and two capacities
// must be coprime
void Vessel:: makeOneLitre(Vessel &V2)
{
    // solution exists iff a and b are co-prime
    if (gcd(capacity, V2.capacity) != 1)
        return;

    while (current != 1)
    {
        // fill A (smaller vessel)
        if (current == 0)
            current = capacity;

        cout << "Vessel 1: " << current << "   Vessel 2: "
             << V2.current << endl;

        // Transfer water from V1 to V2 and reduce current of V1 by
        //  the amount equal to transferred water
        current = current - V2.transfer(current);
    }
```

Geometric Algorithms

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
    // Finally, there will be 1 litre in vessel 1
    cout << "Vessel 1: " << current << "   Vessel 2: "
         << V2.current << endl;
}
```

```
// Fills vessel with given amount and returns the amount of water
// transferred to it. If the vessel becomes full, then the vessel
// is made empty
int Vessel::transfer(int amount)
{
    // If the vessel can accommodate the given amount
    if (current + amount < capacity)
    {
        current += amount;
        return amount;
    }

    // If the vessel cannot accommodate the given amount, then
    // store the amount of water transferred
    int transferred = capacity - current;

    // Since the vessel becomes full, make the vessel
    // empty so that it can be filled again
    current = 0;

    return transferred;
}

// Driver program to test above function
int main()
{
    int a = 3, b = 7;  // a must be smaller than b

    // Create two vessels of capacities a and b
    Vessel V1(a), V2(b);

    // Get 1 litre in first vessel
    V1.makeOneLitre(V2);

    return 0;
}
```

Output:

```
Vessel 1: 3   Vessel 2: 0
Vessel 1: 3   Vessel 2: 3
Vessel 1: 3   Vessel 2: 6
Vessel 1: 2   Vessel 2: 0
Vessel 1: 3   Vessel 2: 2
Vessel 1: 3   Vessel 2: 5
Vessel 1: 1   Vessel 2: 0
```

How does this work?

To prove that the algorithm works, we need to proof that after certain number of iterations in the while loop, we will get 1 litre in V1.
Let ‘a’ be the capacity of vessel V1 and ‘b’ be the capacity of V2. Since we repeatedly transfer water from V1 to V2 until V2 becomes full, we will have ‘a – b (mod a)’ water in V1 when V2 becomes full first time . Once V2 becomes full, it is emptied. We will have ‘a – 2b (mod a)’ water in V1 when V2 is full second time. We repeat the above steps, and get ‘a – nb (mod a)’ water in V1 after the vessel V2 is filled and emptied ‘n’ times. We need to prove that the value of ‘a – nb (mod a)’ will be 1 for a finite integer ‘n’. To prove this, let us consider the following property of coprime numbers.
For any two coprime integers ‘a’ and ‘b’, the integer ‘b’ has a multiplicative inverse modulo ‘a’. In other words, there exists an integer ‘y’ such that $by \equiv 1 (mod\, a)$ (See 3rd point here). After ‘(a – 1)*y’ iterations, we will have ‘a – [(a-1)*y*b (mod a)]‘ water in V1, the value of this expression is ‘a – [(a - 1) * 1] mod a’ which is 1. So the algorithm converges and we get 1 litre in V1.

This article is compiled by Aashish Barnwal. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 12 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 52 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 56 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

Related Tpoics:

- Backtracking | Set 8 (Solving Cryptarithmetic Puzzles)
- Tail Recursion
- Find if two rectangles overlap
- Analysis of Algorithm | Set 4 (Solving Recurrences)
- Print all possible paths from top left to bottom right of a mXn matrix
- Generate all unique partitions of an integer
- Russian Peasant Multiplication
- Closest Pair of Points | O(nlogn) Implementation

21 Tweet 0 0

Writing code in comment? Please use **ideone.com** and share the link here.

27 Comments GeeksforGeeks

Sort by Newest ▾

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

Join the discussion…

Harmeet Singh • 6 months ago

Instead of stopping at 1.
we can iterate n iterations.
where n is z/b;
z = (a(k+1)-1) is divisible by b
for any k.

Shubhankar • 9 months ago

This method will only work in case of |an-b| == 1, where n is a natural number.
Eg: it wont work for say a = 2 and b = 4 or a = 3 and b = 6

Correct me if I am wrong.

1

Aparna Yadav Shubhankar • 9 months ago

yups it only works in |an-bm| = 1 , in other cases its not possible to me

rupayandutta • 9 months ago

Is the proposed algorithm optimized ?
Please consider the below scenerio :

Sample run of the Algo for V1 with capacity 2 and V2 with capacity 3 and we n
1. Fill V1: V1 = 2, V2 = 0
2. Transfer from V1 to V2: V1 = 0, V3 = 2
3. Fill V1: V1 = 2, V2 = 2
4. Transfer from V1 to V2: V1 = 1, V2 = 3

--> Stop as V1 now contains 1 litre. (total 4 operations)

Whereas, if I start with V2 the steps are:
1. Fill V2: V1 = 0, V2 = 3
2. Transfer from V2 to V1: V1 = 2, V2 = 1

--> Stop as V2 now contains 1 litre. (total 2 operations)

So, I feel that we need to take decision whether to start with V1 or V2 dependir
measured(when more than 1unit has to be measured).

srinath • a year ago

There is fundamental logical flaw in the approach...

you are saying "Do this until there is 1 litre in the container a"

If I "know" when there is one litre in the container whats the point of all this....I v
is 1 litre in the container...

To fix this flaw do the following...

find C such that C*b/a gives remainder 1...then repeat the process C times ins
in container a"

2

geekguy ➦ srinath • 11 months ago

Am also thinking the same. -_-

Geeksforgeeks : please clear out doubt ! :)

albert3010 • a year ago

```
main()
{ if(a>b)
{ a=b;b=a;}
while(a!=1)
{k=b/a;
a=a*(k+1)-b;
}
```

Himanshu Gupta · a year ago

I think, by following the above algorithm, one can generate all the multiples of H
this case of co-prime numbers, it should be possible to generalize it from 1 up

Himanshu Gupta · a year ago

I think, by following the above algorithm, one can generate all the multiples of H
this case of co-prime numbers, it should be possible to generalize it from 1 up

Neelesh Nirmal · a year ago

can it be extended for general n in place of 1....

Neelesh Nirmal · a year ago

can it be extended for general n in place of 1....

GeeksforGeeks · a year ago

Saurabh, you can run the given program for your inputs and see how it works.

GeeksforGeeks · a year ago

Saurabh, you can run the given program for your inputs and see how it works.

Saurabh Gupta • a year ago

Can you prove it for a = 41 and b =37.

Saurabh Gupta • a year ago

Can you prove it for a = 41 and b =37.

Varun Tewari • a year ago

Nice one!

Varun Tewari • a year ago

Nice one!

Aditya • a year ago

Considering 2 vessels of capacity 7 and 18.The first time when vessel V2 will water will be in vessel V1. But according to the formula given above(b*n mod a

GeeksforGeeks Mod → Aditya • a year ago

@Aditya: We have updated the proof. Thanks for your inputs. Keep it u

GeeksforGeeks Mod → Aditya • a year ago

@aditya: Thanks for pointing this out. The amount of water in vessel V

We will soon be updating the post.

Aditya • a year ago

"we always have “b*n mod a” water in vessel V1, here n is the number of time then made empty." How???

```
/* Paste your code here (You may delete these lines if not writing co
```

Nikhil Lohia • a year ago

an interesting observation and a generalized solution to the classic water-jug p

Nikhil Lohia • a year ago

an interesting observation and a generalized solution to the classic water-jug p

Venki • a year ago

The algorithm is interesting, making use of congruence properties and Euclid l

There is something interesting to note with regards to C++. See the following s

```
cout << "Vessel 1: " << current << "   Vessel 2: " << V2.current << e
```

V2 is another object, and we are accessing it's private content inside V1 mem avoided, except in few mandatory cases. One can observe similar semantics

How is it possible? I am just curious to see reader's comments. We will cover

sreeram ➜ Venki • a year ago

its because the access modifiers work at class level and not object lev objects separate variables will be created and functions are created or distinguish the variables ..Please correct me if i am wrong ..

Shreyas Basarge • a year ago

An application of Euclid&#039s algorithm for GCD.

Shreyas Basarge • a year ago

An application of Euclid&#039s algorithm for GCD.

Subscribe   Add Disqus to your site