CIT ALGO CELL Contest Feb-2022 Edition

Presented by Aymane Ouahbi, Khalid Samaoui

Contest Link

CONTEST PROBLEMS

01
Maximizing XOR

02
Fibonacci Modified

03
Luck Balance

04
Stock Maximize

CONTEST PROBLEMS

05
Lily's Homework

06
Even Tree

07
Journey To the Moon

08
KnightL on a Chessboard

Topics

- Bitwise Algorithms
- Greedy Algorithms
- Graphs
- Dynamic Programming
- Sorting
- Complexity

“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

— Martin Fowler

01

Maximizing XOR

Naive Solution

Calculating the xor product of all numbers between a and b to get the maximum product, time complexity: O(n^2)

Optimized Solution

Calculate the position of the most significant bit of all products, and return the max product.
Time Complexity: O(n)

02

Fibonacci Modified

Solution Idea

We can just calculate the nth element in a straightforward way, as we do in most sequences.

03

Luck Balance

Solution Idea

This problem can be solved using **greedy algorithms** approach: the amount of luck Lena will lose will be the sum of the minimum lucks in its iteration of the (n-k) iterations (with n being the number of important contests).

04

Stock Maximize

Maximizing the profit

If you can know that a maximum price is coming won't you buy every day stocks so you can sell them all in the maximum day ?

Optimized Solution

We can optimize our solution by iterating backwards, and keeping track of the maximum price so far.

05

Lily’s Homework

Beautiful Array

A beautiful array is simply an array sorted in increasing or decreasing order. The answer is the minimum between the minimum number of swaps needed to sort it increasingly, and the minimum to sort it decreasingly.

Minimum Number of Swaps

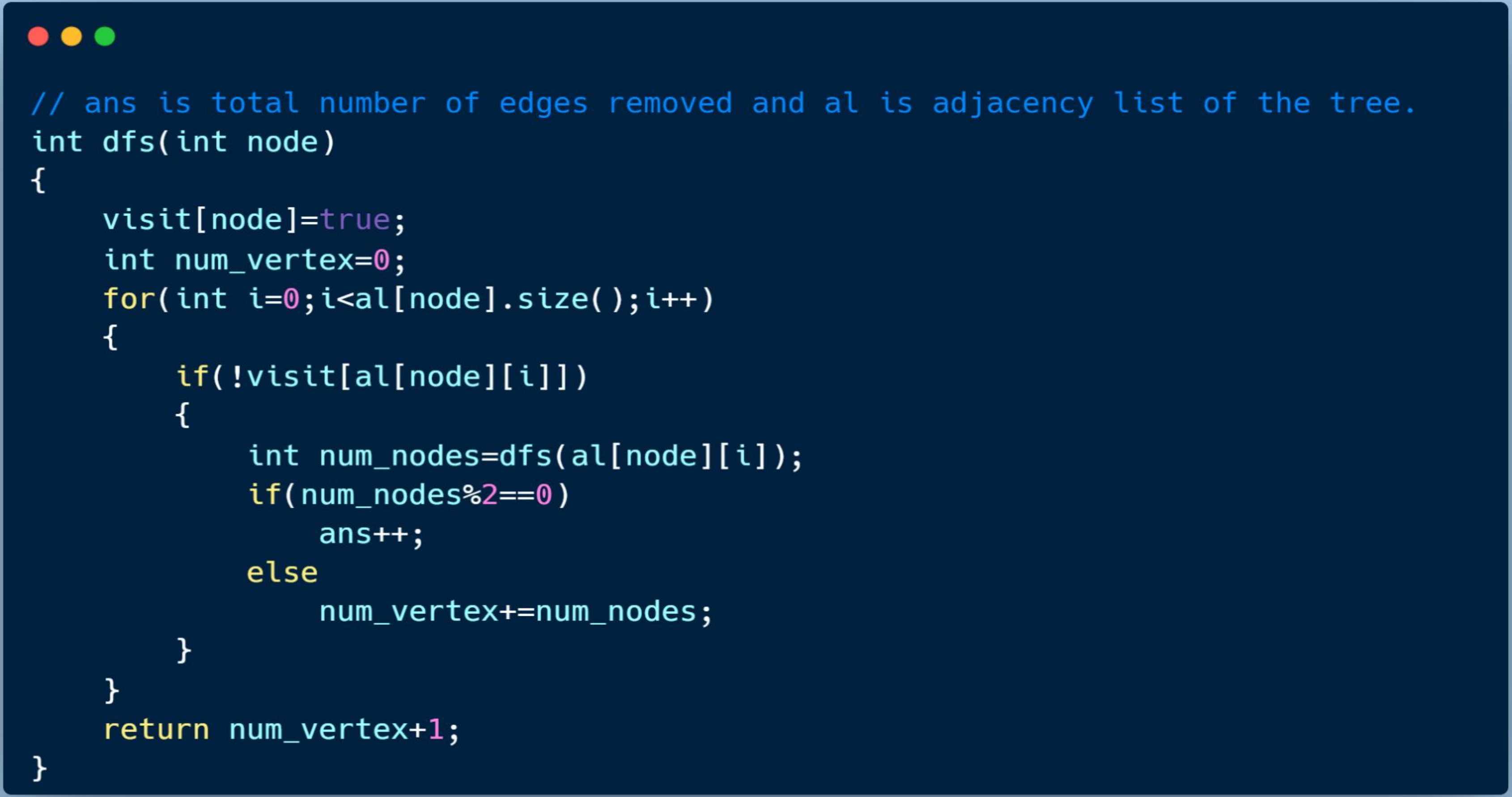
The minimum number of swaps needed to sort a cycle = the size of the cycle – 1

Then the minimum number of swaps needed to sort an array will be the sum of all minimum swaps to sort all the cycles in the array.

Therefore, we will have to present the array elements as nodes holding the value of the element and its index in the original (non sorted) array.

06

Even Tree



```
// ans is total number of edges removed and al is adjacency list of the tree.
int dfs(int node)
{
    visit[node]=true;
    int num_vertex=0;
    for(int i=0;i<al[node].size();i++)
    {
        if(!visit[al[node][i]])
        {
            int num_nodes=dfs(al[node][i]);
            if(num_nodes%2==0)
                ans++;
            else
                num_vertex+=num_nodes;
        }
    }
    return num_vertex+1;
}
```

07

Journey to the Moon

To find astronauts from the same country we are going to use the DFS algorithm (connected components).

We are going to calculate the total number of pairs as we go in our DFS algorithm

08

KnightL on a Chessboard

First we are going to present board cases as graph nodes. Two nodes are linked only if you can go from node u to node v using (i, j) mouvement. Then, we are going to run a BFS from node (0, 0) to node (n-1, n-1) so we can get the shortest path.