Algorithms In A White Box

First, solve the problem. Then, write the code.

John Johnson

By

Sergio Gabriel Sanchez Valencia

gabrielsanv97@gmail.com

searleser97

Contents

Number Theory 3

Greatest Common Divisor 4

BITs Manipulation 5

Least Significant Set Bit 6

Code 6

Graph Theory 7

Articulation Points And Bridges 8

Definition 8

Naive Approach 8

Tarjan’s Approach 8

Idea 9

Number Theory

Greatest Common Divisor

To compute the *GCD* we use one of the most important Euclidean Theorems.

BITs Manipulation

Least Significant Set Bit

First thing we need to notice is that when we add 1 to a number N, what we are doing is just converting the first (right to left) 0-bit into a 1-bit and the 1-bits before get converted to 0-bits because $1 + 1 = 0$ with carry of 1 in binary, therefore we will be having a carry of 1-bit until we find a 0-bit.

Example:

$$00100111 + 1 = 00101000$$

Second thing we need to notice is very simple, lets start by denoting $\overline{N}$ as N with all it's bits inverted (1-bits change to 0-bit and viceversa), if we perform an AND operation between N and $\overline{N}$ we will get all bits in 0 as result.

Example:

$$N = 00100111$$
$$\overline{N} = 11011000$$

So, to achieve our main objective which is to extract the least significant bit (rightmost bit) we can just invert N and add 1 to it that will convert the first 0-bit to 1-bit so if we make an AND operation with N and $\overline{N}$ we get everything before the lsb as 0-bit and after the lsb we also get everything as 0-bit.

And we can write this as the 2's complement since what we did was just to invert bits and add one, which is just the exact definition of 2's complement.

Code

```
int lsb(int n) {
  return n & -n;
}
```

Graph Theory

Articulation Points And Bridges

Definition

We say that a vertex V in a graph G with C connected components is an *articulation point* if its removal increases the number of connected components of G. In other words, let C' be the number of connected components after removing vertex V, if $C' > C$ then V is an *articulation point*.

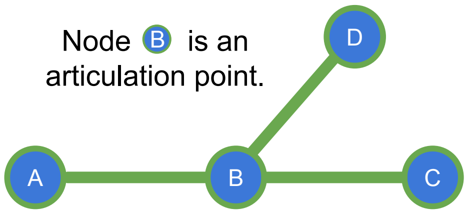

Figure 1

Naive Approach

```
for every vertex V in the graph G do
    Remove V from G
    if the number of connected components increases then
        V is an articulation point
    Add V back to G
```

The complexity of counting the number of *connected components* is $O(V + E)$ therefore, the total complexity of this naive approach is $O(V * (V + E))$.

Tarjan's Approach

First, we need to know that an *ancestor* of some node V is a node A that was discoverd before V in a DFS traversal. i.e. In the graph of figure 1 shown above, if we start our DFS from A and follow the path to C through B ($A \to B \to C$), then A is an ancestor of B and C in this spanning tree generated from the DFS traversal.

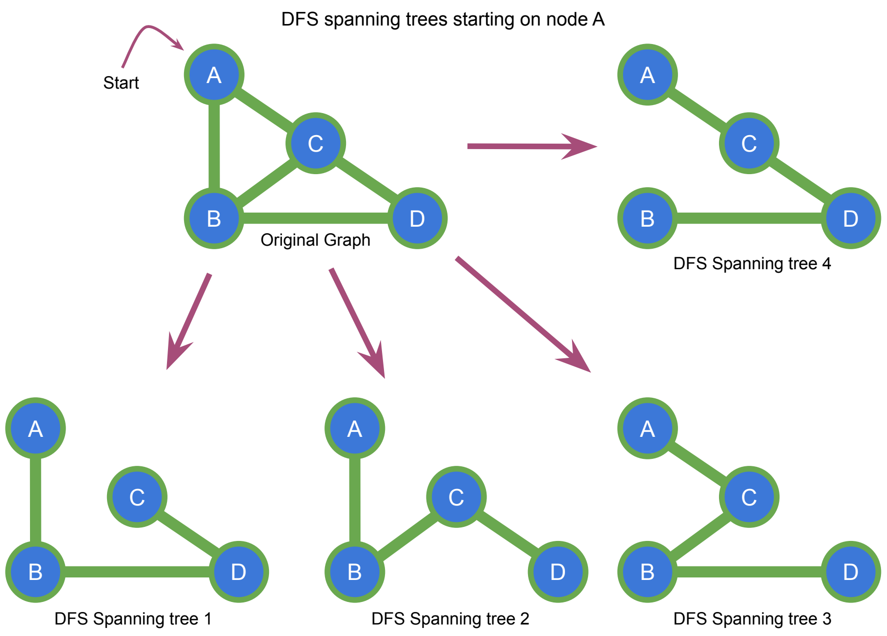

Figure 2: Example of DFS spanning trees of a graph

Now that we know the definition of *ancestor* let's dive into the main idea.

Idea

Let's say there is a node V in some graph G that can be reached by a node U through some intermediate nodes (maybe non intermediate nodes) following some DFS traversal, if V can also be reached by A = "ancestor of U" without passing through U then, U is NOT an articulation point because it means that if we remove U from G we can still reach V from A, hence, the number of *connected components* will remain the same. Hence, we can conclude that the only 2 conditions for U to be an *articulation point* are:

1. If all paths from A to V require U to be in the graph.

2. If U is the root of the DFS traversal with at least 2 children subgraphs disconnected from each other.

Then we can break condition #1 into 2 subconditions:

- U is an *articulation point* if all paths from every ancestor of U to V require U to be in the graph.

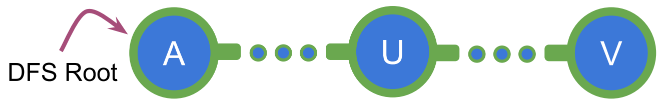

Figure 3

- U is an *articulation point* if it is the root of some cycle in the DFS traversal.

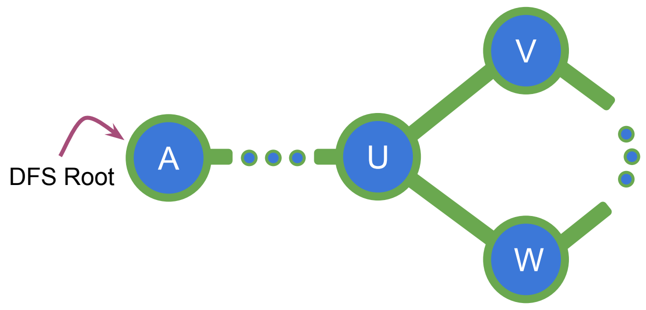

Figure 4