**04.10.2024**

**Repository**: github.com/langchain-ai/langchain

**11.09.2024**

Integer to Binary Conversion:

Using toBinaryString() inbuilt method of the Integer class of Java

Integer.toBinaryString(num)

The method Integer.toBinaryString(num) returns a String, not an integer. It converts the integer num into its binary representation in string format.

Leetcode POD

My Logic:

class Solution {

public int minBitFlips(int start, int goal) {

String start\_binary = Integer.toBinaryString(start);

String goal\_binary = Integer.toBinaryString(goal);

char[] start\_char = String start\_binary.toCharArray();

char[] goal\_char = String goal\_binary.toCharArray();

int max\_flips = 0;

for(char i : start\_char){

for(char j : goal\_char){

if(i!=j)

max\_flips++;

}

}

return max\_flips;

}

}

Corrected Code:

class Solution {

public int minBitFlips(int start, int goal) {

// Convert to binary strings

String start\_binary = Integer.toBinaryString(start);

String goal\_binary = Integer.toBinaryString(goal);

// Pad the shorter string with leading zeros

int maxLength = Math.max(start\_binary.length(), goal\_binary.length());

// Pad the binary strings so that they are of equal length

start\_binary = String.format("%" + maxLength + "s", start\_binary).replace(' ', '0');

goal\_binary = String.format("%" + maxLength + "s", goal\_binary).replace(' ', '0');

// Convert to char arrays

char[] start\_char = start\_binary.toCharArray();

char[] goal\_char = goal\_binary.toCharArray();

int max\_flips = 0;

// Compare the bits at each position

for(int i = 0; i < start\_char.length; i++){

if(start\_char[i] != goal\_char[i]){

max\_flips++;

}

}

return max\_flips;

}

}

Reasons for correction:

for(char i : start\_char){

for(char j : goal\_char){

if(i != j)

max\_flips++;

}

}

This is incorrect because it compares every character in start\_char with every character in goal\_char. This results in a quadratic comparison, which is not the intended behavior. You should compare corresponding bits, not all combinations of bits.

Handling Binary Strings of Different Lengths: You are comparing the start and goal binary strings directly. However, binary strings for different numbers can have different lengths.

Optimised Code:

class Solution {

public int minBitFlips(int start, int goal) {

// XOR start and goal to find the bits that differ

int xor = start ^ goal;

// Count the number of 1's in the XOR result (those are the bit flips needed)

return Integer.bitCount(xor);

}

}

XOR - same bits - ZERO

XOR - different bits - ONE

If you meant .bitCount(), it's a method in Java, used with integers to count the number of 1 bits in the binary representation of a number:

E.g.; int bitCount = Integer.bitCount(number);

This method returns how many bits are set to 1 in the binary form of the integer.

GFG POTD:

class Solution {

// Function to return the minimum cost of connecting the ropes.

public long minCost(long[] arr) {

PriorityQueue<Long> pq = new PriorityQueue<>();

ArrayList<Long>ls = new ArrayList<>();

for(long a : arr){

pq.add(a);

}

while(pq.size()>1){

long n = pq.poll() + pq.poll();

ls.add(n);

pq.add(n);

}

long ans = 0;

for(int i = 0; i<ls.size(); i++){

ans += ls.get(i);

}

return ans;

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_