LECTURE 2

EFFICIENCY

Po Hao Chen

June 4, 2023

BOSTON UNIVERSITY

Efficiency

Objectives

- Understand when we need efficient algorithms
- Recognizing the strategies to design our algorithms

Time Complexity of the Algorithm

- Optimization is often needed to pass tough test cases. As a programmer, we need to make sure our algorithm doesn't breaks assuming the input is not unreasonable
- How do we know how "good" our algorithm needs to be?
- For a general computer processor (If we want to solve within a reasonable time):
  - O(1) or O(logn) - $n > 10^8$
  - Linear: $O(n)$ - $n \leq 10^8$
  - Logarithmic: $O(nlogn)$ - $n \leq 10^6$
  - Quadratic: $O(n^2)$ - $n \leq 10^4$
  - Cubic: $O(n^3)$ - $n \leq 500$
  - Quartic: $O(n^4)$ - $n \leq 100$
  - Exponential: $O(2^n)$ - $n \leq 25$

Example: Missing Number

- You are given all numbers between 1,2,...,n except one. Your task is to find the missing number.
- Constraint: $2 \leq n \leq 2 * 10^5$
- Example:
  5 $\leftarrow$ n
  2 3 1 5 $\leftarrow$ the numbers we have
- Output: 4

Approach?

- Sort the array and enumerate the array, find the missing element. Complexity?
- Read input and store them in hashtable
- Complexity? Space?

Optimal Solution

- Take the total in O(1), subtract each item in linear time. What remains is the missing number.
- Summation formula: $\sum_{i=1}^{n} = \frac{n(n+1)}{2}$
- O(n) Time and O(1) Space
- This will satisfy the time constraint set by the problem given the input can be up to $2 * 10^5$

Bruteforce

- While some problems have elegant solutions, it is possible to bruteforce the answer given that the input constraint is loose.
- Bitmask Trick:

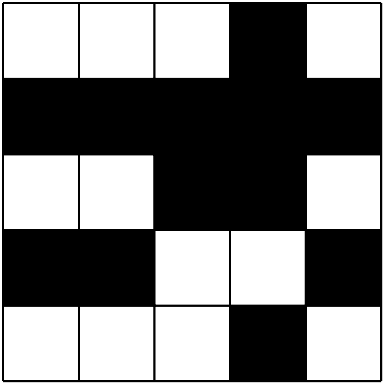
```
int n = 3;
int A[] = {1,2,3};
for(int i = 0 ; i < (1 << n) ; i++){
    for(int j = 0 ; j < n ; j++){
        if ( i & (1 << j) ){
            cout << A[j] << ' ';
        }
    }
    cout << '\n';
}
```

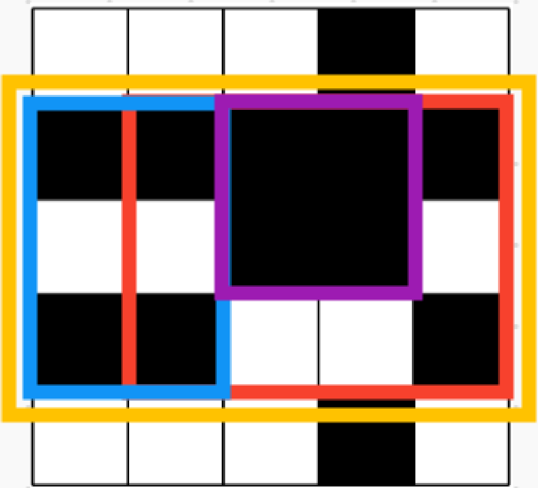
Example: Preparing Olympiad (CF)

- Given n problems, you estimate the i-th one to have difficulty C_i.
- Create a problemset such that each contains **at least** 2 problems.
- Total difficulty D must be $L \leq D \leq R$
- The hardest problem and the easiest problem must differ at least x
- Given n,L,R,x. Find the number of ways to choose the problemset that satsifies the constraints.
- Constraint: $1 \leq n \leq 15$, $1 \leq L \leq R \leq 10^9, 1 \leq x \leq 10^6$
- Example:
  3 5 6 1
  1 2 3

Example: Beautiful Subgrid (CSES)

- Given nxn grid, a subgrid is beautiful if all four of its corner are colored black.
- How many beautiful subgrids? (4 in this picture, can you find them?)

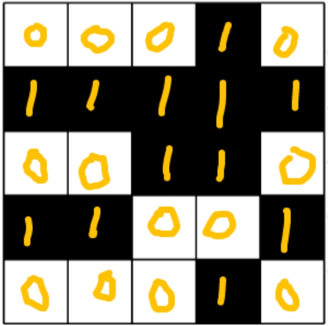




Solution : 4 beautiful subgrids

Using Bits

- We represent each square as 0/1. If it's colored we set it to 1.



- We get a binary matrix:

Strategy

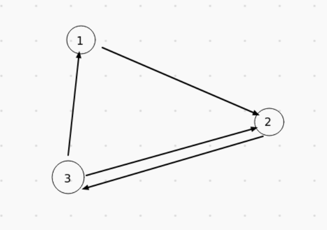
- Our matrix:
  00010
  11111
  00110
  11001
  00010
- we need to check each pair, and compute the number of beautiful subgrids
- Observation:
  - A beautiful subgrid is formed when there are 2 valid sides
  - Each side is only valid, if there are two colored (vertically) opposing each other
  - If find n valid sides between 2 rows, we can find the different combinations using $\binom{n}{2}$

(BONUS) Input Efficiency

- We can also improve the time efficiency of our algorithm by changing how we read the input.
- For example: adjacency list vs adjacency matrix

Example: Graph Path

Consider a directed graph that has n nodes and m edges. Your task is to count the number of paths from node 1 to node n with exactly k edges.



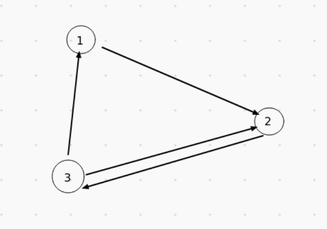
We can iterate each vertex, and keep track of the count of edges we have currently used. Try to reach the destination, if the count equal to k we add it to our solution

Input:

1 2, 2 3, 3 1, 3 2

Strategy 1

Consider a directed graph that has n nodes and m edges. Your task is to count the number of paths from node 1 to node n with exactly k edges.

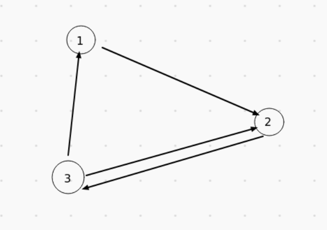


With Adjacency List:

- We can iterate each vertex, and keep track of the count of edges we have currently used. Try to reach the destination, if the count equal to k we add it to our solution
- Complexity: $O(V^k)$

Strategy 2

Consider a directed graph that has n nodes and m edges. Your task is to count the number of paths from node 1 to node n with exactly k edges.

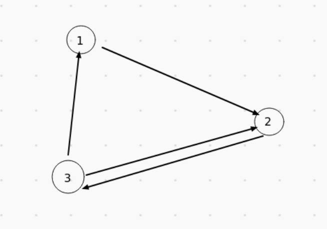


With Adjacency Matrix:

- Raise the matrix M to the power of k
- Our answer is $M[0][n-1]$
- Complexity: $O(V^3 logk)$

Test Cases

Example 1:



$$M = \begin{bmatrix} 010 \\ 001 \\ 110 \end{bmatrix} \qquad M^2 = \begin{bmatrix} 010 \\ 001 \\ 110 \end{bmatrix} * \begin{bmatrix} 010 \\ 001 \\ 110 \end{bmatrix} = \begin{bmatrix} 001 \\ 110 \\ 011 \end{bmatrix}$$

There is one path from Node 1 to Node 3 with 2 edges

$1 \rightarrow 2 \rightarrow 3$

Look at the entry (0,2) of M^2 (0-based index)