Brute Force Algorithms

ACM-ICPC Week 2

The most powerful technique

Mathematics: Dang after 1000 years of study we don’t have a way to figure out if a number N is prime

Programmer: Easy, just check if any number divides it

Overview

Checking every subset

Meet-In-The-Middle

Checking every permutation

Checking every subset

Suppose you are given an array of n numbers $\{a_1, a_2, \ldots, a_n\}$, and a number X.

You want to find if it is possible to make a set $S = \{i_1, i_2, \ldots i_k\}$

Such that $a_{i_1} \oplus a_{i_2} \oplus \ldots \oplus a_{i_{k-1}} \oplus a_{i_k} = X$

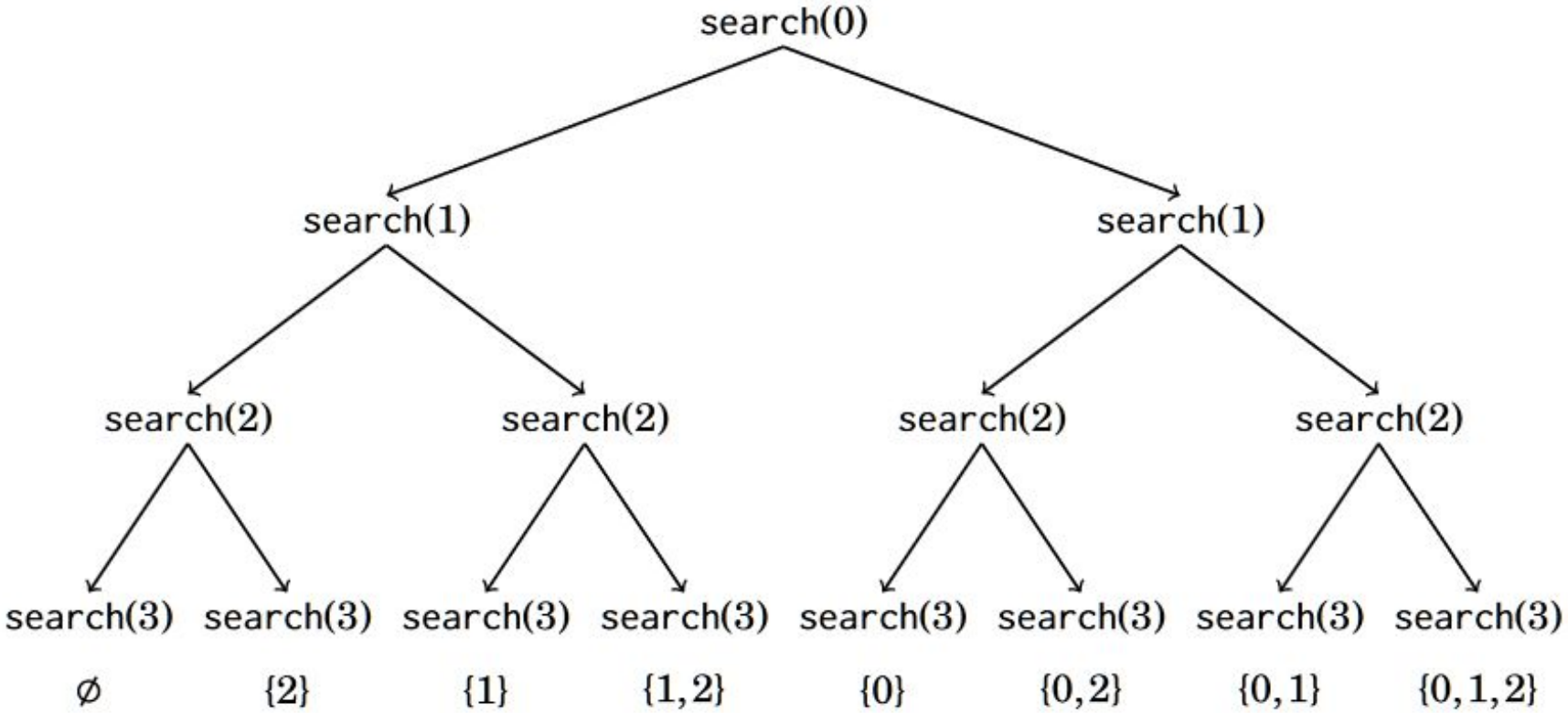
Checking every subset - Recursion

How can we generate and check every set S?

For example, if n=3, we want to generate the 8 sets:

$$\{\{\}, \{0\}, \{1\}, \{2\}, \{3\},$$
$$\{0,1\}, \{1,2\}, \{0,2\}, \{0,1,2\}\}$$

Do n recursive calls. For each call, split into two paths: 1 with a given element, 1 without.


search(0)
search(1)
search(1)
search(2)
search(2)
search(2)
search(2)
search(3)
search(3)
search(3)
search(3)
search(3)
search(3)
search(3)
search(3)
∅
{2}
{1}
{1,2}
{0}
{0,2}
{0,1}
{0,1,2}

Checking every subset - iterative

Utilize the binary representation of integers

- Runs much faster

Meet-In-The-Middle

- Divide the array into 2 equal-sized parts, get every possible combination for each half, and combine the results.
- Compute all values two halves of the array can make, creating 2 arrays of size 2^(n/2). Call them A and B.
- For each value a in A, check if B contains a⊕X. If it does, than it is possible, since a⊕(a⊕X)=X
    - Can be done in linear time using a hashset
    - Can also be done with sorting and two-pointers technique

Checking every Permutation (Recursion)

- How do you generate all permutations of length n?
- For each element, generate all permutations without that element. Then add that element at the end for the final permutation.

Weekly Challenge #2

https://github.com/OSUACM/Weekly_Events