Maximum XOR of Two Numbers in an Array - GeeksforGeeks

Source: https://www.geeksforgeeks.org/maximum-xor-of-two-numbers-in-an-array/

Courses Tutorials Practice Jobs DSA Tutorial Interview Questions Quizzes Must Do Advanced DSA System Design Aptitude Puzzles Interview Corner DSA Python Technical Scripter 2026 Explore DSA Fundamentals Logic Building Problems Analysis of Algorithms Data Structures Array Data Structure String in Data Structure Hashing in Data Structure Linked List Data Structure Stack Data Structure Queue Data Structure Tree Data Structure Graph Data Structure Trie Data Structure Algorithms Searching Algorithms Sorting Algorithms Introduction to Recursion Greedy Algorithms Tutorial Graph Algorithms Dynamic Programming or DP Bitwise Algorithms Advanced Segment Tree Binary Indexed Tree or Fenwick Tree Square Root (Sqrt) Decomposition Algorithm Binary Lifting Geometry Interview Preparation Interview Corner GfG160 Practice Problem GeeksforGeeks Practice - Leading Online Coding Platform Problem of The Day - Develop the Habit of Coding DSA Course 90% Refund Maximum XOR of Two Numbers in an Array Last Updated : 12 Jul, 2025 Given an array arr[] of non-negative integers. The task is to find the maximum possible XOR of any two elements of the array. Example : Input: arr[] = [26, 100, 25, 13, 4, 14] Output: 126 Explanation: XOR of 26 ^ 100 = 126, which is the maximum possible XOR. Input: arr[] = [1, 2, 3, 4, 5, 6, 7] Output: 7 Explanation: XOR of 1 ^ 6 = 7, which is the maximum possible XOR. Try it on GfG Practice Table of Content [Naive Approach] Using 2 Nested Loops - O(n^2) Time and O(1) Space [Expected Approach - 1] - Using Bit Masking and HashSet - O(n * log m) Time and O(n * log m) Space [Expected Approach - 2] - Using Trie - O(n * log m) Time and O(n * log m) Space [Naive Approach] Using 2 Nested Loops - O(n^2) Time and O(1) Space The idea is to generate all possible pairs of elements of the array arr[] and find the maximum among them. To do so, use two nested loops to generate all the pairs and compute XOR of them. The maximum of all the pairs is the result. C++ #include <bits/stdc++.h> using namespace std ; // Function to find the maximum XOR int maxXor (vector < int > & arr) { int res = 0 ; int fir = 0 , sec = 0 ; // Generate all possible pairs for (int i = 0 ; i < arr . size (); i ++) { for (int j = i + 1 ; j < arr . size (); j ++) { if ((arr [i] ^ arr [j]) > res){ res = max (res , arr [i] ^ arr [j]); fir = arr [i], sec = arr [j]; } } } return res ; } int main () { vector < int > arr = { 26 , 100 , 25 , 13 , 4 , 14 }; cout << maxXor (arr); return 0 ; } Java class GfG { // Function to find the maximum XOR static int maxXor (int [] arr) { int res = 0 ; // Generate all possible pairs for (int i = 0 ; i < arr . length ; i ++) { for (int j = i + 1 ; j < arr . length ; j ++) { res = Math . max (res , arr [i] ^ arr [j]); } } return res ; } public static void main (String [] args) { int [] arr = { 26 , 100 , 25 , 13 , 4 , 14 }; System . out . println (maxXor (arr)); } } Python # Function to find the maximum XOR def maxXor (arr): res = 0 # Generate all possible pairs for i in range (len (arr)): for j in range (i + 1 , len (arr)): res = max (res , arr [i] ^ arr [j]) return res if __name__ == "__main__" : arr = [26 , 100 , 25 , 13 , 4 , 14] print (maxXor (arr)) C# using System ; class GfG { // Function to find the maximum XOR static int maxXor (int [] arr) { int res = 0 ; // Generate all possible pairs for (int i = 0 ; i < arr . Length ; i ++) { for (int j = i + 1 ; j < arr . Length ; j ++) { res = Math . Max (res , arr [i] ^ arr [j]); } } return res ; } static void Main (string [] args) { int [] arr = { 26 , 100 , 25 , 13 , 4 , 14 }; Console . WriteLine (maxXor (arr)); } } JavaScript // Function to find the maximum XOR function maxXor (arr) { let res = 0 ; // Generate all possible pairs for (let i = 0 ; i < arr . length ; i ++) { for (let j = i + 1 ; j < arr . length ; j ++) { res = Math . max (res , arr [i] ^ arr [j]); } } return res ; } let arr = [26 , 100 , 25 , 13 , 4 , 14]; console . log (maxXor (arr)); Output 126 [Expected Approach - 1] - Using Bit Masking and HashSet - O(n * log m) Time and O(n * log m) Space The idea is to find two numbers in an array arr[] such that their XOR equals a number X , where X is the maximum value we want to achieve at the current bit position (i-th bit). To achieve the largest XOR value, we aim to maximize the number of 1s in the XOR result, starting from the most significant bit (leftmost bit) to the least significant bit (rightmost bit). To evaluate each bit position, we use a mask . A mask helps us focus on specific bits of the numbers in the array by keeping only the relevant bits up to the current position and ignoring the rest. Using the mask, we extract prefixes for all numbers in the array (i.e., the portions of the numbers defined by the mask). These prefixes help us determine if a pair of numbers exists in the array whose XOR can yield a maximum value at the current bit. For each bit position: Apply the mask to extract prefixes from the numbers. Try to update the maximum XOR value by assuming the i-th bit of the result

is 1. Using the set of prefixes, we check if any two prefixes can produce the desired XOR value (with the i-th bit set). This process is repeated for all 32 bits, starting from the leftmost bit, to compute the largest possible XOR value step by step. C++ #include <bits/stdc++.h> using namespace std ; // Function to find the maximum XOR int maxXor (vector < int > & arr) { int res = 0 , mask = 0 ; // to store all unique bits unordered_set < int > s ; for (int i = 30 ; i >= 0 ; i --) { // set the i-th bit in mask mask |= (1 << i); for (auto value : arr) { // keep prefix of all elements // till the i-th bit s . insert (value & mask); } int cur = res | (1 << i); for (int prefix : s) { if (s . count (cur ^ prefix)) { res = cur ; break ; } } s . clear (); } return res ; } int main () { vector < int > arr = { 26 , 100 , 25 , 13 , 4 , 14 }; cout << maxXor (arr); return 0 ; } Java import java.util.HashSet ; class GfG { // Function to find the maximum XOR static int maxXor (int [] arr) { int res = 0 , mask = 0 ; // to store all unique bits HashSet < Integer > s = new HashSet <> (); for (int i = 30 ; i >= 0 ; i --) { // set the i-th bit in mask mask |= (1 << i); for (int value : arr) { // keep prefix of all elements // till the i-th bit s . add (value & mask); } int cur = res | (1 << i); for (int prefix : s) { if (s . contains (cur ^ prefix)) { res = cur ; break ; } } s . clear (); } return res ; } public static void main (String [] args) { int [] arr = { 26 , 100 , 25 , 13 , 4 , 14 }; System . out . println (maxXor (arr)); } } Python # Function to find the maximum XOR def maxXor (arr): res = 0 mask = 0 # to store all unique bits s = set () for i in range (30 , - 1 , - 1): # set the i-th bit in mask mask |= (1 << i) for num in arr : # keep prefix of all elements # till the i-th bit s . add (num & mask) cur = res | (1 << i) for prefix in s : if cur ^ prefix in s : res = cur break s . clear () return res if __name__ == "__main__" : arr = [26 , 100 , 25 , 13 , 4 , 14] print (maxXor (arr)) C# using System ; using System.Collections.Generic ; class GfG { // Function to find the maximum XOR static int maxXor (int [] arr) { int res = 0 , mask = 0 ; // to store all unique bits HashSet < int > s = new HashSet < int > (); for (int i = 30 ; i >= 0 ; i --) { // set the i-th bit in mask mask |= (1 << i); foreach (int value in arr) { // keep prefix of all elements // till the i-th bit s . Add (value & mask); } int cur = res | (1 << i); foreach (int prefix in s) { if (s . Contains (cur ^ prefix)) { res = cur ; break ; } } s . Clear (); } return res ; } static void Main (string [] args) { int [] arr = { 26 , 100 , 25 , 13 , 4 , 14 }; Console . WriteLine (maxXor (arr)); } } JavaScript // Function to find the maximum XOR function maxXor (arr) { let res = 0 , mask = 0 ; // to store all unique bits let s = new Set (); for (let i = 30 ; i >= 0 ; i --) { // set the i-th bit in mask mask |= (1 << i); for (let num of arr) { // keep prefix of all elements // till the i-th bit s . add (num & mask); } let cur = res | (1 << i); for (let prefix of s) { if (s . has (cur ^ prefix)) { res = cur ; break ; } } s . clear (); } return res ; } let arr = [26 , 100 , 25 , 13 , 4 , 14]; console . log (maxXor (arr)); Output 126 Time Complexity: O(n * log m), where n is the size of array and m is the maximum element. Auxiliary Space: O(n * log m) [Expected Approach - 2] - Using Trie - O(n * log m) Time and O(n * log m) Space The idea is to use Trie data structure to effectively store and search bits of each element of array arr[]. For each element arr[i], check if an element with opposite bits is present in the Trie or not i.e. if current bit of arr[i] is set (1), then check if unset bit (0) at current index is present in Trie. Note: Please refer to Trie article to know more about insert and search operation. C++ #include <bits/stdc++.h> using namespace std ; class Node { public : Node * one ; Node * zero ; Node () { one = nullptr ; zero = nullptr ; } }; class Trie { public : Node * root ; Trie () { root = new Node (); } // Function to insert in Trie void insert (int n) { Node * curr = root ; for (int i = 31 ; i >= 0 ; i --) { int bit = (n >> i) & 1 ; // Check if the bit is 0 if (bit == 0) { if (curr -> zero == NULL) { curr -> zero = new Node (); } curr = curr -> zero ; } // Else if bit is 1 else { if (curr -> one == NULL) { curr -> one = new Node (); } curr = curr -> one ; } } } // Function to find element having // the maximum XOR with value n int findXOR (int n) { Node * curr = root ; int res = 0 ; for (int i = 31 ; i >= 0 ; i --) { int bit = (n >> i) & 1 ; // if the bit is 0 if (bit == 0) { // if set bit is present if (curr -> one) { curr = curr -> one ; res += (1 << i); } else { curr = curr -> zero ; } } // Else if bit is 1 else { // if unset bit is present if (curr -> zero) { curr = curr -> zero ; res += (1 << i); } else { curr = curr -> one ; } } } return res ; } }; // Function to find the maximum XOR int maxXor (vector < int > & arr) { int res = 0 ; Trie * t = new Trie (); // insert the first element in trie t -> insert (arr [0]); for (int i = 1 ; i < arr . size (); i ++) { res = max (t -> findXOR (arr [i]), res); t -> insert (arr [i]); } return res ; } int main () { vector < int > arr = { 26 , 100 , 25 , 13 , 4 , 14 }; cout << maxXor (arr); return 0 ; } Java // Java program to find the maximum // XOR n of two elements in an array // using Trie data structure import java.util.* ; class Node { Node one ; Node zero ; Node () { one = null ; zero = null ; } } class Trie { Node root ; Trie () { root = new Node (); } // Function to insert in Trie void insert (int n) { Node curr = root ; for (int i = 31 ; i >= 0 ; i --) { int bit = (n >> i) & 1 ; // Check if the bit is 0 if (bit == 0) { if (curr . zero == null) { curr . zero = new Node (); } curr = curr . zero ; } // Else if bit is 1 else { if (curr . one == null) { curr . one = new Node (); } curr = curr . one ; } } } // Function to find element having // the maximum XOR with value n int findXOR (int n) { Node curr = root ; int res = 0 ; for (int i = 31 ; i >= 0 ; i --) { int bit = (n >> i) & 1 ; // if the bit is 0 if (bit == 0) { // if set bit is present if (curr . one != null) { curr = curr . one ; res += (1 << i); } else { curr = curr . zero ; } } // Else if bit is 1 else { // if unset bit is

present if (curr . zero != null) { curr = curr . zero ; res += (1 << i); } else { curr = curr . one ; } } } return res ; } } class GfG { // Function to find the maximum XOR static int maxXor (int [] arr) { int res = 0 ; Trie t = new Trie (); // insert the first element in trie t . insert (arr [0]); for (int i = 1 ; i < arr . length ; i ++) { res = Math . max (t . findXOR (arr [i]), res); t . insert (arr [i]); } return res ; } public static void main (String [] args) { int [] arr = { 26 , 100 , 25 , 13 , 4 , 14 }; System . out . println (maxXor (arr)); } } Python # Python program to find the maximum # XOR n of two elements in an array # using Trie data structure class Node : def __init__ (self): self . one = None self . zero = None class Trie : def __init__ (self): self . root = Node () # Function to insert in Trie def insert (self , n): curr = self . root for i in range (31 , - 1 , - 1): bit = (n >> i) & 1 # Check if the bit is 0 if bit == 0 : if not curr . zero : curr . zero = Node () curr = curr . zero # Else if bit is 1 else : if not curr . one : curr . one = Node () curr = curr . one # Function to find element having # the maximum XOR with value n def findXOR (self , n): curr = self . root res = 0 for i in range (31 , - 1 , - 1): bit = (n >> i) & 1 # if the bit is 0 if bit == 0 : # if set bit is present if curr . one : curr = curr . one res += (1 << i) else : curr = curr . zero # Else if bit is 1 else : # if unset bit is present if curr . zero : curr = curr . zero res += (1 << i) else : curr = curr . one return res # Function to find the maximum XOR def maxXor (arr): res = 0 t = Trie () # insert the first element in trie t . insert (arr [0]) for i in range (1 , len (arr)): res = max (t . findXOR (arr [i]), res) t . insert (arr [i]) return res if __name__ == "__main__" : arr = [26 , 100 , 25 , 13 , 4 , 14] print (maxXor (arr)) C# // C# program to find the maximum // XOR n of two elements in an array // using Trie data structure using System ; class Node { public Node one ; public Node zero ; public Node () { one = null ; zero = null ; } } class Trie { public Node root ; public Trie () { root = new Node (); } // Function to insert in Trie public void insert (int n) { Node curr = root ; for (int i = 31 ; i >= 0 ; i --) { int bit = (n >> i) & 1 ; // Check if the bit is 0 if (bit == 0) { if (curr . zero == null) { curr . zero = new Node (); } curr = curr . zero ; } // Else if bit is 1 else { if (curr . one == null) { curr . one = new Node (); } curr = curr . one ; } } } // Function to find element having // the maximum XOR with value n public int findXOR (int n) { Node curr = root ; int res = 0 ; for (int i = 31 ; i >= 0 ; i --) { int bit = (n >> i) & 1 ; // if the bit is 0 if (bit == 0) { // if set bit is present if (curr . one != null) { curr = curr . one ; res += (1 << i); } else { curr = curr . zero ; } } // Else if bit is 1 else { // if unset bit is present if (curr . zero != null) { curr = curr . zero ; res += (1 << i); } else { curr = curr . one ; } } } return res ; } } class GfG { // Function to find the maximum XOR static int maxXor (int [] arr) { int res = 0 ; Trie t = new Trie (); // insert the first element in trie t . insert (arr [0]); for (int i = 1 ; i < arr . Length ; i ++) { res = Math . Max (t . findXOR (arr [i]), res); t . insert (arr [i]); } return res ; } static void Main (string [] args) { int [] arr = { 26 , 100 , 25 , 13 , 4 , 14 }; Console . WriteLine (maxXor (arr)); } } JavaScript // JavaScript program to find the maximum // XOR n of two elements in an array // using Trie data structure class Node { constructor () { this . one = null ; this . zero = null ; } } class Trie { constructor () { this . root = new Node (); } // Function to insert in Trie insert (n) { let curr = this . root ; for (let i = 31 ; i >= 0 ; i --) { let bit = (n >> i) & 1 ; // Check if the bit is 0 if (bit === 0) { if (! curr . zero) { curr . zero = new Node (); } curr = curr . zero ; } // Else if bit is 1 else { if (! curr . one) { curr . one = new Node (); } curr = curr . one ; } } } // Function to find element having // the maximum XOR with value n findXOR (n) { let curr = this . root ; let res = 0 ; for (let i = 31 ; i >= 0 ; i --) { let bit = (n >> i) & 1 ; // if the bit is 0 if (bit === 0) { // if set bit is present if (curr . one) { curr = curr . one ; res += (1 << i); } else { curr = curr . zero ; } } // Else if bit is 1 else { // if unset bit is present if (curr . zero) { curr = curr . zero ; res += (1 << i); } else { curr = curr . one ; } } } return res ; } } // Function to find the maximum XOR function maxXOR (arr) { let res = 0 ; let t = new Trie (); // insert the first element in trie t . insert (arr [0]); for (let i = 1 ; i < arr . length ; i ++) { res = Math . max (t . findXOR (arr [i]), res); t . insert (arr [i]); } return res ; } let arr = [26 , 100 , 25 , 13 , 4 , 14]; console . log (maxXOR (arr)); Output 126 Comment Article Tags: Article Tags: DSA Bitwise-XOR