AND xor OR

locked

by bokamvinaykumar

Problem | Submissions | Leaderboard | Discussions | Editorial

Given an array $A[]$ of N distinct elements. Let M_1 and M_2 be the smallest and the next smallest element in the interval $[L, R]$ where $1 \leq L < R \leq N$.

$S_i = (((M_1 \wedge M_2) \oplus (M_1 \vee M_2)) \wedge (M_1 \oplus M_2))$.

where $\wedge, \vee, \oplus$, are the bitwise operators AND, OR and XOR respectively.
Your task is to find the maximum possible value of S_i.

Input Format

First line contains integer N.
Second line contains N integers, representing elements of the array $A[]$.

Constraints

$1 < N \leq 10^6$
$1 \leq A_i \leq 10^9$

Output Format

Print the value of maximum possible value of S_i.

Sample Input

```
5
9 6 3 5 2
```

Sample Output

```
15
```

Explanation

Consider the interval $[1, 2]$ the result will be maximum.
$(((9 \wedge 6) \oplus (9 \vee 6)) \wedge (9 \oplus 6)) = 15$

Submissions: 36
Max Score: 30

Rate This Challenge:

More

Current Buffer (saved locally, editable) C

```
1 #include <assert.h>
2 #include <limits.h>
```

```c
#include <math.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* readline();
char** split_string(char*);

/*
 * Complete the andXorOr function below.
 */
int andXorOr(int a_count, int* a) {
    /*
     * Write your code here.
     */

}

int main()
{
    FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");

    char* a_count_endptr;
    char* a_count_str = readline();
    int a_count = strtol(a_count_str, &a_count_endptr, 10);

    if (a_count_endptr == a_count_str || *a_count_endptr != '\0') { exit(EXIT_FAILURE); }

    char** a_temp = split_string(readline());

    int a[a_count];

    for (int a_itr = 0; a_itr < a_count; a_itr++) {
        char* a_item_endptr;
        char* a_item_str = a_temp[a_itr];
        int a_item = strtol(a_item_str, &a_item_endptr, 10);

        if (a_item_endptr == a_item_str || *a_item_endptr != '\0') { exit(EXIT_FAILURE); }

        a[a_itr] = a_item;
    }

    int result = andXorOr(a_count, a);

    fprintf(fptr, "%d\n", result);

    fclose(fptr);

    return 0;
}

char* readline() {
    size_t alloc_length = 1024;
    size_t data_length = 0;
    char* data = malloc(alloc_length);

    while (true) {
        char* cursor = data + data_length;
        char* line = fgets(cursor, alloc_length - data_length, stdin);

        if (!line) { break; }

        data_length += strlen(cursor);

        if (data_length < alloc_length - 1 || data[data_length - 1] == '\n') { break; }

        size_t new_length = alloc_length << 1;
        data = realloc(data, new_length);

        if (!data) { break; }

        alloc_length = new_length;
    }
```

```
77
78      if (data[data_length - 1] == '\n') {
79          data[data_length - 1] = '\0';
80      }
81
82      data = realloc(data, data_length);
83
84      return data;
85  }
86
87  char** split_string(char* str) {
88      char** splits = NULL;
89      char* token = strtok(str, " ");
90
91      int spaces = 0;
92
93      while (token) {
94          splits = realloc(splits, sizeof(char*) * ++spaces);
95          if (!splits) {
96              return splits;
97          }
98
99          splits[spaces - 1] = token;
100
101         token = strtok(NULL, " ");
102     }
103
104     return splits;
105 }
106
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code