In The Name Of God

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*While(cin >> n) With Scanf\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

while (scanf("%lld", &n) != EOF)

EOF 🡪 End Of File

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Convert String To Digit\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <sstream>

#include <string>

long long stoi(string s)

{

long long r;

stringstream ss;

ss << s;

ss >> r;

return r;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Conevrt Digit To String\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <sstream>

#include <string>

string itos(long long r)

{

stringstream ss;

string s;

ss << r;

ss >> s;

return s;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Convert Decimal To Negative Base\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <sstream>

string negative (long long n, long long base)

{

stringstream ss;

if (n == 0)

return "0";

string s, converted = "";

while (n != 0)

{

int r = n % base;

n /= base;

if (r < 0)

{

r += (-base);

n += 1;

}

ss << r;

ss >> s;

converted = s + converted;

}

return converted;

}

cout << negative (n, base) << endl;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Convert Decimal To Positive Base\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <cstring>

char reVal(long long m)

{

if (m >= 0 && m <= 9)

return (char)(m + '0');

else

return (char)(m - 10 + 'A');

}

void strev(char \*s)

{

char temp;

long long i, len = strlen(s);

for (i = 0; i < len / 2; i++)

{

temp = s[i];

s[i] = s[len - i - 1];

s[len - i - 1] = temp;

}

}

char\* fromDeci(char r[], long long base, long long n)

{

long long i = 0;

while (n > 0)

{

r[i++] = reVal(n % base);

n /= base;

}

r[i] = '\0';

strev(r);

return r;

}

printf("Equivalent of %lld in base %lld is : %s\n", n, base, fromDeci(r, base, n));

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Convert Roman Number To Decimal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <string>

int value(char r)

{

if (r == 'I')

return 1;

if (r == 'V')

return 5;

if (r == 'X')

return 10;

if (r == 'L')

return 50;

if (r == 'C')

return 100;

if (r == 'D')

return 500;

if (r == 'M')

return 1000;

return -1;

}

long long romanToDecimal(string &s)

{

long long b, c, r = 0;

for (int i = 0; i<s.length(); i++)

{

b = value(s[i]);

if (i + 1 < s.length())

{

c = value(s[i + 1]);

if (b >= c)

r = r + b;

else

{

r += c - b;

i++;

}

}

else

{

r += b;

i++;

}

}

return r;

}

cout << "Integer form of Roman Numeral is : " << romanToDecimal(s) << endl;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Convert Any Base To Decimal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <cstring>

int val(char c)

{

if (c >= '0' && c <= '9')

return (int)c - '0';

else

return (int)c - 'A' + 10;

}

long long toDeci(char \*s, int base)

{

long long i, num = 0, power = 1, len = strlen(s);

for (i = len - 1; i >= 0; i--)

{

if (val(s[i]) >= base)

{

printf("Invalid Number");

return -1;

}

num += val(s[i]) \* power;

power \*= base;

}

return num;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Long Long Factorial\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define MAX 500

int multiply(int x, int res[], int res\_size);

void factorial(int n)

{

int res[MAX];

res[0] = 1;

int res\_size = 1;

for (int x = 2; x <= n; x++)

res\_size = multiply(x, res, res\_size);

for (int i = res\_size - 1; i >= 0; i--)

cout << res[i];

}

int multiply(int x, int res[], int res\_size)

{

int carry = 0;

for (int i = 0; i<res\_size; i++)

{

int prod = res[i] \* x + carry;

res[i] = prod % 10;

carry = prod / 10;

}

while (carry)

{

res[res\_size] = carry % 10;

carry = carry / 10;

res\_size++;

}

return res\_size;

}

factorial(n);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Long Long power\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <vector>

#include <algorithm>

#include <string>

#include <cmath>

vector < pair <long long, pair < vector <long long>, long long> > > v;

vector <long long> vi;

#define MAX 100000

long long multiply(long long x, long long res[], long long res\_size)

{

long long prod, carry = 0;

for (int i = 0; i < res\_size; i++)

{

prod = res[i] \* x + carry;

res[i] = prod % 10;

carry = prod / 10;

}

while (carry) {

res[res\_size] = carry % 10;

carry = carry / 10;

res\_size++;

}

return res\_size;

}

void power(int x, int n)

{

vi.clear();

if (n == 0)

{

vi.push\_back(1);

v.push\_back({ n,{vi, 1} });

return;

}

long long i, res[MAX], res\_size = 0, temp = x;

while (temp != 0)

{

res[res\_size++] = temp % 10;

temp = temp / 10;

}

for (i = 2; i <= n; i++)

res\_size = multiply(x, res, res\_size);

for (i = res\_size - 1; i >= 0; i--)

vi.push\_back(res[i]);

v.push\_back({ n,{vi,res\_size} });

}

long long i, base, exponent;

for (i = 0; i <= exponent; i++)

power(base, i);

for (i = 0; i <= exponent; i++)

{

cout << base <<" ^ " << v[i].first << " = ";

for (j = 0; j < v[i].second.second; j++)

cout << v[i].second.first[j];

cout << " , size = " << v[i].second.second << endl;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Power\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long powering(long long a, long long b)

{

if (a == 0)

return 0;

if (b == 0)

return 1;

long long ans;

if (b == 1)

return a;

else if (b % 2 == 0)

{

long long t = powering(a, b / 2);

return t \* t;

}

else

return a \* powering(a, b - 1);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Long Long Mul And Odd\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <vector>

#define Mod 1000000000

vector<long long> sum(vector<long long> a, vector<long long> b)

{

vector<long long> v;

long long i, c, a\_s, b\_s;

a\_s = a.size();

b\_s = b.size();

for (i = 0; i < a\_s && i < b\_s; i++)

v.push\_back(a[i] + b[i]);

while (v[v.size() - 1] == 0)

v.pop\_back();

if (a\_s == i && b\_s != i)

for (i; i < b\_s; i++)

v.push\_back(b[i]);

else if (a\_s != i && b\_s == i)

for (i; i < a\_s; i++)

v.push\_back(a[i]);

c = v.size();

for (i = 0; i < c - 1; i++)

{

v[i + 1] += v[i] / Mod;

v[i] %= Mod;

}

if (v[i] > Mod)

{

c = v[i] / Mod;

v[i] %= Mod;

v.push\_back(c);

}

return v;

}

vector<long long> product(vector<long long> a, vector<long long> b)

{

vector<long long> v;

long long i, j, c, a\_s, b\_s;

a\_s = a.size();

b\_s = b.size();

for (i = 0; i < b\_s; i++)

v.push\_back(0);

for (i = 0; i < a\_s; i++)

{

v.push\_back(0);

for (j = 0; j < b\_s; j++)

{

c = (a[i] \* b[j]) / Mod;

v[j + i] += (a[i] \* b[j]) % Mod;

v[j + i + 1] += c;

}

}

if (a[a.size() - 1] == 0 || b[b.size() - 1] == 0)

{

v.clear();

v.push\_back(0);

}

else

while (v.back() == 0)

v.pop\_back();

return v;

}

int main()

{

long long i, v\_s, c, j, k, n;

vector <long long> r, re[100007], v, a, b;

cin >> i >> j >> n;

re[0].push\_back(0);

re[1].push\_back(i);

re[2].push\_back(j);

for (i = 3; i <= n; i++)

{

a = b = re[i - 1];

a = product(a, b);

b = re[i - 2];

re[i] = sum(a, b);

}

cout << re[n][re[n].size() - 1];

for (i = re[n].size() - 2; i >= 0; i--)

{

k = re[n][i];

while (k / 100000000 < 1)

{

cout << '0';

k = k \* 10;

}

cout << re[n][i];

}

cout << endl;

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Jaygasht\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <algorithm>

int main()

{

long long n, i, a[10000 + 7];

scanf\_s("%lld", &n);

for (i = 0; i < n; i++)

scanf\_s("%lld", a + i);

sort(a, a + n);

std::cout << "The n! possible permutations with n elements:\n";

do

{

for (i = 0; i < n; i++)

printf("%lld ", a[i]);

printf("\n");

} while (next\_permutation(a, a + n));

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Be Dast Avardan Tedad Mosalas Az Roi Mohit\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long recursive(long long n, long long s)

{

if (n <= 1)

return s;

recursive(n - 3, s + (n - 1) / 2);

}

int main()

{

long long n, s = 0, k, m;

scanf("%lld", &n);

if (n == 3)

printf("1\n");

else

{

if (n % 2 == 1)

m = (n + 3)\*(n + 3) / 48;

else

m = recursive(n / 2, s);

printf("%lld\n", m);

}

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*No. Of Prime Number Till N\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long N = 100007;

bool prime[100000 + 10];

int np[100000 + 10];

void notprime()

{

for (long long i = 2; i \* i <= N; i++)

if (!prime[i])

for (long long j = i \* i; j <= p; j += i)

prime[j] = 1;

for (long long i = 2; i <= N; i++)

np[i] += np[i - 1] + !prime[i];

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Prime Number\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <bitset>

#define Primei1 10000000

#define Primei2 (int)(Primei1 - 3) / 2

bitset<Primei2 + 1> bset;

vector <int> primes;

bool isPrime(long long a)

{

if (a <= primes[primes.size() - 1])

return binary\_search(primes.begin(), primes.end(), a);

else

for (int i = 0; (long long)primes[i] \* primes[i] <= a; i++)

if (a % primes[i] == 0) return false;

return true;

}

void setPrimes()

{

int i, j;

for (i = 0; (i \* i) <= sqrt(Primei1); i++) //we only have to get primes up to sqrt(Primei1)

if (!bset.test(i))

for (j = i + 1; (2 \* j + 1)\*(2 \* i + 3) <= Primei1; j++)

bset.set(((2 \* j + 1)\*(2 \* i + 3) - 3) / 2); //setting all non-primes

primes.push\_back(2); //store the first prime (that is 2)

for (i = 1, j = 0; j < Primei2 + 1; j++)

if (!bset.test(j))

primes.push\_back(2 \* j + 3);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LIS Dynamic Programing\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <algorithm> // Lonegst Incresing sequence

long long lis(long long arr[], long long n)

{

long long lis[10000 + 7]; //length n

lis[0] = 1;

for (long long i = 1; i < n; i++)

{

lis[i] = 1;

for (long long j = 0; j < i; j++)

if (arr[i] > arr[j] && lis[i] < lis[j] + 1)

lis[i] = lis[j] + 1;

}

return \*max\_element(lis, lis + n); // Return maximum value in lis[]

}

int main()

{

long long a[100000 + 7], n;

cin >> n;

for (size\_t i = 0; i < n; i++)

cin >> a[i];

printf("Length of lis is %lld\n", lis(a, n));

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LCS Dynamic Programing\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <algorithm> // Lonegst common sequence

#include <string>

int L[100 + 123][100 + 123];

int main()

{

string x, y, LCS;

LCS.resize(100);

cin >> x >> y;

//radif va sotoone avval ro 0 mikone garche to in barname hame az avval 0an

for (int i = 0; i <= x.length(); i++)//ta khode x.length

L[i][0] = 0;

for (int i = 0; i <= y.length(); i++)// ta khode y.length

L[0][i] = 0;

//por kardane matrix

for (int i = 1; i <= x.length(); i++)// havaset bashe indota az 1 shoro mishan va ta khode x.length mire

for (int j = 1; j <= y.length(); j++)// havaset bashe indota az 1 shoro mishan va ta khode y.length mire

{

if (x[i - 1] == y[j - 1])

L[i][j] = L[i - 1][j - 1] + 1;

else

L[i][j] = max(L[i - 1][j], L[i][j - 1]);

}

//peyda kardane khode LCS in ye tikke jozve cheatsheete

int r = x.length();

int c = y.length();

int i = L[r][c];

while (r > 0 && c > 0)

{

if (x[r - 1] == y[c - 1])

{

LCS[i - 1] = x[r - 1];

i--; r--; c--;

}

else if (L[r - 1][c] > L[r][c - 1])

r--;

else

c--;

}

cout << L[x.length()][y.length()] << "\t" << LCS << endl;

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Largest Sum Continuous Subarray\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long maxSubArraySum(long long a[], long size)

{

long long max\_so\_far = -100000000 - 7, max\_ending\_here = 0;

for (int i = 0; i < size; i++)

{

max\_ending\_here = max\_ending\_here + a[i];

if (max\_so\_far < max\_ending\_here)

max\_so\_far = max\_ending\_here;

if (max\_ending\_here < 0)

max\_ending\_here = 0;

}

return max\_so\_far;

}

long long a[1000 + 7], n, max\_sum;

max\_sum = maxSubArraySum(a, n);

cout << "Maximum contiguous sum is " << max\_sum;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Maximum Sum Incresing Subsequence\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long maxSumIS(long long arr[], long long n)

{

long long i, j, max = 0, m[10000 + 7];

for (i = 0; i < n; i++)

m[i] = arr[i];

for (i = 1; i < n; i++)

for (j = 0; j < i; j++)

if (arr[i] > arr[j] &&

m[i] < m[j] + arr[i])

m[i] = m[j] + arr[i];

for (i = 0; i < n; i++)

if (max < m[i])

max = m[i];

return max;

}

long long a[10000 + 7], n;

printf("Sum of maximum sum increasing subsequence is %d\n", maxSumIS(a, n));

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Maximum Prouduct Of Subarray Size k\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <algorithm>

long long maxProductSubarrayOfSizeK(long long A[], long long n, long long k)

{

sort(A, A + n);

long long product = 1;

if (A[n - 1] == 0 && (k & 1))

return 0;

if (A[n - 1] <= 0 && (k & 1)) {

for (long long i = n - 1; i >= n - k; i--)

product \*= A[i];

return product;

}

long long i = 0;

long long j = n - 1;

if (k & 1) {

product \*= A[j];

j--;

k--;

}

k >>= 1;

for (long long itr = 0; itr < k; itr++)

{

long long left\_product = A[i] \* A[i + 1];

long long right\_product = A[j] \* A[j - 1];

if (left\_product > right\_product) {

product \*= left\_product;

i += 2;

}

else {

product \*= right\_product;

j -= 2;

}

}

return product;

}

long long a[1000 + 7], n, k;

cout << maxProductSubarrayOfSizeK(a, n, k);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Maximum Prouduct Continuous Subarray\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <cstdio>

long long min(int x, int y) { return x < y ? x : y; }

long long max(int x, int y) { return x > y ? x : y; }

long long maxSubarrayProduct(long long arr[], long long n)

{

long long max\_ending\_here = 1, min\_ending\_here = 1, max\_so\_far = 1, i;

for (i = 0; i < n; i++)

{

if (arr[i] > 0)

{

max\_ending\_here = max\_ending\_here \* arr[i];

min\_ending\_here = min(min\_ending\_here \* arr[i], 1);

}

else if (arr[i] == 0)

{

max\_ending\_here = 1;

min\_ending\_here = 1;

}

else

{

int temp = max\_ending\_here;

max\_ending\_here = max(min\_ending\_here \* arr[i], 1);

min\_ending\_here = (temp \* arr[i]);

}

if (max\_so\_far < max\_ending\_here)

max\_so\_far = max\_ending\_here;

}

return max\_so\_far;

}

long long n, a[20 + 7];

printf("Maximum Sub array product is %lld", maxSubarrayProduct(a, n));

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Fibonacci\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define long long long

const long M = 1000000000; // modulo

map<long, long> F;

long f(long n)

{

if (F.count(n)) return F[n];

long k = n / 2;

if (n % 2 == 0) // n=2\*k

return F[n] = (f(k)\*f(k) + f(k - 1)\*f(k - 1)) % M;

else // n=2\*k+1

return F[n] = (f(k)\*f(k + 1) + f(k - 1)\*f(k)) % M;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Binary Search Tree\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long a[10000 + 7], c[10000 + 7];

long long binarySearch(long long low, long long high, long long key)

{

if (key > high)

return high;

long long mid;

while (low <= high)

{

mid = (low + high) / 2;

if (a[mid] == key)

{

low = mid + 1;

mid++;

}

else if (a[mid] > key)

high = mid - 1;

else

low = mid + 1;

}

return mid;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BFS (Graph)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <queue>

#include <vector>

vector <int> list[100000 + 12];

bool mark[100000 + 12];

void bfs(int u)

{

queue<int> q;

q.push(u);

while (!q.empty())

{

int current = q.pop;

mark[current] = true;

for (int i = 0; i<list[current].size(); i++) //(every node in adjacency list of current)

if (list[current][i] == false) //(node is not visited)

q.push(list[current][i]);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DFS (Graph)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <queue>

#include <vector>

vector <int> list[100000 + 12];

bool mark[100000 + 12];

void dfs(int v)

{

mark[v] = true;

for (int i = 0; i<list[v].size(); i++)

{

if (!mark[list[v][i]])

dfs(list[v][i]);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Trie (Graph)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <map>

#include <string>

string my\_itoa(int value, int base)

{

enum { kMaxDigits = 35 };

string buf;

buf.reserve(kMaxDigits);

if (base < 2 || base > 16) return buf;

int quotient = value;

do

{

buf += "0123456789abcdef"[abs(quotient % base)];

quotient /= base;

} while (quotient);

if (value < 0 && base == 10) buf += '-';

reverse(buf.begin(), buf.end());

return buf;

}

struct trie

{

bool end;

bool vojoodechar[150];

map < int, int> md;

vector <trie\* > d;

int index;

trie()

{

fill(vojoodechar, vojoodechar + 150, false);

end = false;

index = 0;

}

};

trie \*root;

void insertstring(string s)

{

trie \*q = root;

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

{

if (q->vojoodechar[s[i]] == true)

{

q = q->d[q->md[s[i]]];

continue;

}

q->vojoodechar[s[i]] = true;

trie \*e = new trie;

q->md[s[i]] = q->index;

q->d.push\_back(e);

q->index++;

q = e;

}

q->end = true;

}

bool findstring(string s)

{

trie \*q = root;

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

{

if (q->vojoodechar[s[i]] == false)

return false;

q = q->d[q->md[s[i]]];

}

if (q->end == false)

return false;

return true;

}

root = new trie; // dakhele main hamishe bashe

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Max Heap (Graph)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <vector>

vector <pair<long long, long long > > maxheap;

void darj\_maxheap(long long a, long long idx)

{

maxheap.push\_back(make\_pair(a, idx));

if (maxheap.size() == 2)

return;

int i = maxheap.size() - 1;

while (maxheap[i].first > maxheap[i / 2].first)

{

pair<long long, long long> temp = maxheap[i / 2];

maxheap[i / 2] = maxheap[i];

maxheap[i] = temp;

i = i / 2;

if (i <= 1)

return;

}

}

void hazf\_maxheap()

{

if (maxheap.size() == 2)

{

maxheap.pop\_back();

return;

}

if (maxheap.size() == 1)

return;

maxheap[1] = maxheap[maxheap.size() - 1];

maxheap.pop\_back();

int i = 1;

lab:

if (maxheap.size() - 1 >= (i \* 2) + 1)

{

if (maxheap[i].first < maxheap[i \* 2].first || maxheap[i].first < maxheap[(i \* 2) + 1].first)

{

if (maxheap[i \* 2].first > maxheap[(i \* 2) + 1].first)

{

pair<long long, long long> temp = maxheap[i \* 2];

maxheap[i \* 2] = maxheap[i];

maxheap[i] = temp;

i = i \* 2;

goto lab;

}

else

{

pair<long long, long long> temp = maxheap[i \* 2 + 1];

maxheap[i \* 2 + 1] = maxheap[i];

maxheap[i] = temp;

i = i \* 2 + 1;

goto lab;

}

}

return;

}

if (maxheap.size() - 1 == i \* 2)

{

if (maxheap[i].first < maxheap[i \* 2].first)

{

pair<long long, long long> temp = maxheap[i \* 2];

maxheap[i \* 2] = maxheap[i];

maxheap[i] = temp;

i = i \* 2;

goto lab;

}

return;

}

if (maxheap.size() - 1 < i \* 2)

return;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Segment (Graph)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long segment[4000000 + 12],a[4000000 + 12];//az yek shoroo mishe pas node hamishe yeke

void build(long long node, long long start, long long end) //build(1,0,n-1) node 1 bayad bashe, 0 , n-1 ke bazeye arrayast

{

if (start == end)

segment[node] = a[start];

else

{

long long mid = (start + end) / 2;

build(2 \* node, start, mid);

build(2 \* node + 1, mid + 1, end);

segment[node] = segment[2 \* node] + segment[2 \* node + 1];//in khat moteghayyere

}

}

void update(long long node, long long start, long long end, long long idx, long long val) //node kollan hamishe 1e, starto end avvalo akhare arraye, index az 0e val ham ke update bayad beshe

{

if (start == end)

{

a[idx] = val;

segment[node] = val;

}

else

{

long long mid = (start + end) / 2;

if (start <= idx && idx <= mid)

update(2 \* node, start, mid, idx, val);

else

update(2 \* node + 1, mid + 1, end, idx, val);

segment[node] = segment[2 \* node] + segment[2 \* node + 1];//in khat moteghayyere

}

}

long long query(long long node, long long start, long long end, long long l, long long r)//node baz yeke starto end bazeye arraye, l va r ham ke query

{

if (r < start || end < l)

return 0; //in khat baraye sum intori shode, baraye har no' derakhti bayad ozve khonsa bargardoone

if (l <= start && end <= r)

return segment[node];

long long mid = (start + end) / 2;

long long p1 = query(2 \* node, start, mid, l, r);

long long p2 = query(2 \* node + 1, mid + 1, end, l, r);

return p1 + p2; //va in khat moteghayyere

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Binary Tree (Graph)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class nd

{

public:

int n;

nd \*ped;

nd \*left;

nd \*right;

nd()

{

ped = left = right = NULL;

}

};

nd \*total; //sarasari

void set(nd \*n, int m)//n rishe darakht-m adadi ke bayad darj shavad

{

if ((\*n).n<m)

{

//right

if ((\*n).right != NULL)

set((\*n).right, m);

else

{

total = n;

nd \*p = new nd;

(\*p).n = m;

(\*total).right = p;

(\*p).ped = total;

}

}

if ((\*n).n>m)

{

//left

if ((\*n).left != NULL)

set((\*n).left, m);

else

{

total = n;

nd \*p = new nd;

(\*p).n = m;

(\*total).left = p;

(\*p).ped = total;

}

}

}

void print\_pre(nd \*r)//rishe-chap-rast(r rishe ye koli ast ke dar main ersal mishavad)

{

cout << (\*r).n << endl;

if ((\*r).left != NULL)

print\_pre((\*r).left);

if ((\*r).right != NULL)

print\_pre((\*r).right);

}

void print\_in(nd \*r)//chap-rishe-rast(r rishe ye koli ast ke dar main ersal mishavad)

{

if ((\*r).left != NULL)

print\_in((\*r).left);

cout << (\*r).n << endl;

if ((\*r).right != NULL)

print\_in((\*r).right);

}

void print\_post(nd \*r)//chap-rast-rishe(r rishe ye koli ast ke dar main ersal mishavad)

{

if ((\*r).left != NULL)

print\_post((\*r).left);

if ((\*r).right != NULL)

print\_post((\*r).right);

cout << (\*r).n << endl;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Disjoint (Graph)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int n\_d = 0;//andazeye araye

int link[1000 + 12];

int size[1000 + 10];

bool sargoroh[10000 + 12];

void set\_n\_d(int n)//hatmaaaaaaaan seda zade shavad

{

n\_d = n;

}

void build\_d()//hatmaaaaaaaan seda zade shavad

{

for (int i = 1; i <= n\_d; i++)

link[i] = i;

for (int i = 1; i <= n\_d; i++)

size[i] = 1;

for (int i = 1; i <= n\_d; i++)

sargoroh[i] = 1;

}

int find\_d(int x)//return sargoroh

{

while (x != link[x]) x = link[x];

return x;

}

bool same\_d(int a, int b) //bebine sargoroh ha yekie ya na :|

{

return find\_d(a) == find\_d(b);

}

void unite\_d(int a, int b)

{

a = find\_d(a);

b = find\_d(b);

if (size[a] < size[b]) swap(a, b);

size[a] += size[b];

link[b] = a;

sargoroh[b] = 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Kruskal (Graph)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

const int MAX = 1e4 + 5;

int id[MAX], nodes, edges;

pair <long long, pair<int, int> > p[MAX];

void initialize()

{

for (int i = 0; i < MAX; ++i)

id[i] = i;

}

int root(int x)

{

while (id[x] != x)

{

id[x] = id[id[x]];

x = id[x];

}

return x;

}

void union1(int x, int y)

{

int p = root(x);

int q = root(y);

id[p] = id[q];

}

long long kruskal(pair<long long, pair<int, int> > p[])

{

int x, y;

long long cost, minimumCost = 0;

for (int i = 0; i < edges; ++i)

{

// Selecting edges one by one in increasing order from the beginning

x = p[i].second.first;

y = p[i].second.second;

cost = p[i].first;

// Check if the selected edge is creating a cycle or not

if (root(x) != root(y))

{

minimumCost += cost;

union1(x, y);

}

}

return minimumCost;

}

int main()

{

int x, y;

long long weight, cost, minimumCost;

initialize();

cin >> nodes >> edges;

for (int i = 0; i < edges; ++i)

{

cin >> x >> y >> weight;

p[i] = make\_pair(weight, make\_pair(x, y));

}

// Sort the edges in the ascending order

sort(p, p + edges);

minimumCost = kruskal(p);

cout << minimumCost << endl;

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Prime (Graph)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <queue>

#include <vector>

#include <functional>

const int MAX = 1e4 + 5;

typedef pair<long long, int> PII;

bool marked[MAX];

vector <PII> adj[MAX];

long long prim(int x)

{

priority\_queue<PII, vector<PII>, greater<PII> > Q;

int y;

long long minimumCost = 0;

PII p;

Q.push(make\_pair(0, x));

while (!Q.empty())

{

// Select the edge with minimum weight

p = Q.top();

Q.pop();

x = p.second;

// Checking for cycle

if (marked[x] == true)

continue;

minimumCost += p.first;

marked[x] = true;

for (int i = 0; i < adj[x].size(); ++i)

{

y = adj[x][i].second;

if (marked[y] == false)

Q.push(adj[x][i]);

}

}

return minimumCost;

}

int main()

{

int nodes, edges, x, y;

long long weight, minimumCost;

cin >> nodes >> edges;

for (int i = 0; i < edges; ++i)

{

cin >> x >> y >> weight;

adj[x].push\_back(make\_pair(weight, y));

adj[y].push\_back(make\_pair(weight, x));

}

// Selecting 1 as the starting node

minimumCost = prim(1);

cout << minimumCost << endl;

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Floyd Warshall (Graph)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for (int k = 1; k <= n; k++)

for (int i = 1; i <= n; i++)

for (int j = 1; j <= n; j++)

dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Dijkstra (Graph)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <cstring>

#include <vector>

#include <set>

#define SIZE 100000 + 1

///Assume the source vertex = 1

vector < pair < int, int > > v[SIZE]; // each vertex has all the connected vertices with the edges weights

int dist[SIZE];

bool vis[SIZE];

void dijkstra()

{

int e, w, x, wei;

// set the vertices distances as infinity

memset(vis, false, sizeof vis); // set all vertex as unvisited

dist[1] = 0;

multiset < pair < int, int > > s; // multiset do the job as a min-priority queue

s.insert({ 0 , 1 }); // insert the source node with distance = 0

while (!s.empty())

{

pair <int, int> p = \*s.begin(); // pop the vertex with the minimum distance

s.erase(s.begin());

x = p.second, wei = p.first;

if (vis[x]) continue; // check if the popped vertex is visited before

vis[x] = true;

for (int i = 0; i < v[x].size(); i++)

{

e = v[x][i].first, w = v[x][i].second;

if (dist[x] + w < dist[e])

{ // check if the next vertex distance could be minimized

dist[e] = dist[x] + w;

s.insert({ dist[e], e }); // insert the next vertex with the updated distance

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Hamilton (Graph)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define MAXN 10000000

#define NOT\_IN\_STACK 0

#define IN\_STACK 1

bool dfs(int v, bool adj[][MAXN], int label[MAXN], int instack\_count, int n)

{

if (instack\_count == n)

return true;

for (int i = 0; i<n; i++)

if (adj[v][i] && label[i] == NOT\_IN\_STACK)

{

label[i] = IN\_STACK;

if (dfs(i, adj, label, instack\_count + 1, n))

return true;

label[i] = NOT\_IN\_STACK;

}

return false;

}

bool check\_using\_dfs(bool adj[][MAXN], int n)

{

int label[MAXN];

for (int i = 0; i<n; i++)

label[i] = NOT\_IN\_STACK;

for (int i = 0; i<n; i++)

{

label[i] = IN\_STACK;

if (dfs(i, adj, label, 1, n))

return true;

label[i] = NOT\_IN\_STACK;

}

return false;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Bit Tree (Graph)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long n;//////size array

long long BIT[10000005];

void update(long long index, long long value)

{

while (index <= n)

{

BIT[index] += value;

index += (index&(-index));

}

}

long long query(long long index)

{

long long ans = 0;

while (index>0)

{

ans += BIT[index];

index -= (index&(-index));

}

return ans;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Bit Manipulation\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We can use integer az bit and we can use **ASCII** representation for convert **characters** to **bit**.

**BitWise Operation:**

1. **NOT ( ~ ):** if the ith bit is 0, it will change it to 1 and vice versa.
2. **AND ( & )**
3. **OR ( | )**
4. **XOR ( ^ ):** If both are 0 or 1, the bit in the resulting is 0, otherwise 1.
5. **Left Shift ( << ):**  1 << n = 2n
6. **Right Shift ( >> ):** 1 >> n = 1 / 2n

\*Point: **We Can Understand Array Has Only One Odd Repeated Number Or Not By XOR(^) All Elements.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Is power Of 2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool isPowerOfTwo(long long x)

{

return (x && !(x & (x - 1)));

}

isPowerOfTwo(n) ? printf("Yes\n") : printf("No\n");

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Count Number Of Ones\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long count\_one(long long n)

{

long long count = 0;

while (n)

{

n = n & (n - 1);

count++;

}

return count;

}

printf("%d\n", count\_one(t));

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*If Number In I’th Possition Is Set\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool check(long long N, int i)

{

if (N & (1 << i))

return true;

else

return false;

}

printf("%d\n", check(t, i));

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Compute XOR From 1 To n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long computeXOR(long long n)

{

if (n % 4 == 0)

return n;

if (n % 4 == 1)

return 1;

if (n % 4 == 2)

return n + 1;

else

return 0;

}

cout << computeXOR(n) << endl;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Swap 2 Number\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void swap(long long n, long long m)

{

m ^= n;

n ^= m;

m ^= n;

cout << n << "\t" << m << endl;

}

swap(n, m);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Generate All The Possible Subsets Of A Set\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

As we all know there are 2N possible subsets of any given set with N element. We have 2 conditions for each elements :

1. There is Sth (Element is 1)
2. there isn’t Sth (Element is 0)

set :

A = { a, b }

All The Possible Subsets :

0 = (00)2 = {}  
1 = (01)2 = {a}  
2 = (10)2 = {b}  
3 = (11)2 = {a, b}

void possibleSubsets(char A[], long long N)

{

for (int i = 0; i < (1 << N); ++i)

{

for (int j = 0; j < N; ++j)

if (i & (1 << j))

cout << A[j] << " ";

cout << endl;

}

}

possibleSubsets(A , N);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Largest Power Of 2 Less Than OR Equal To N (MSB)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long largest\_power(long N)

{

//changing all right side bits to 1 :

N = N | (N >> 1);

N = N | (N >> 2);

N = N | (N >> 4);

N = N | (N >> 8);

//as now the number is 2 \* x-1, where x is required answer, so adding 1 and dividing it by:

return (N + 1) >> 1;

}

largest\_power(N);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Next Power 2 Of A Diget\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned long long nextPowerOf2(unsigned long long n)

{

unsigned long long p = 1;

if (n && !(n & (n - 1)))

return n;

while (p < n)

p <<= 1;

return p;

}

cout << nextPowerOf2(n) << endl;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Tricks With Bit\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **x ^ ( x & (x - 1)) OR x & (-x) :** Returns the rightmost 1 in binary representation of x. //Getting lowest set bit of a number

**Eg :** 5 🡪 101 🡪 01 (1) 10 🡪 1010 🡪 10 (2) 96 🡪 110000 🡪 10000 (32)

1. **x | (1 << n) :**  Returns the number x with the nth bit set.

**Eg :** if n = 2 then

10 🡪 1010 🡪 1110 (14) 16 🡪 10000 🡪 10100 (20) 6 🡪 110 🡪 110 (6)

1. **~x : 1’s complement**
2. **~x + 1 OR -x : 2’s complement**
3. **x & (x - 1):** Stripping off the lowest set bit. //Khonsa kardan Kamtarin Mizan Bit

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*If Number Has Alternate Pattern\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool allBitsAreSet(long long n)

{

if (((n + 1) & n) == 0)

return true;

return false;

}

bool bitsAreInAltOrder(long long n)

{

long long num = n ^ (n >> 1);

return allBitsAreSet(num);

}

long long n;

if (bitsAreInAltOrder(n))

cout << "Yes";

else

cout << "No";

Eg : 5 🡪 101 🡪 Yes 7 🡪 111 🡪 No 1 🡪 01 🡪 Yes

2🡪 10 🡪 Yes 21 🡪 10101 🡪 Yes 15 🡪 1111 🡪 No

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Clear All Bit From LSB To I’th Bit\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long Clear\_Lsb\_To\_ith(long long x, int i)

{

long long mask;

mask = ~((1 << i + 1) - 1);

x &= mask;

return x;

}

cout << Clear\_Lsb\_To\_ith(n, i);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Clear All Bit From MSB To I’th Bit\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long Clear\_Msb\_To\_ith(long long x, int i)

{

long long mask;

mask = (1 << i) - 1;

x &= mask;

return x;

}

cout << Clear\_Msb\_To\_ith(n, i);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Convert Upper Case To Lower Case And Vice Versa\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The bit representation of upper case and lower case English alphabets are :

A -> 01000001 a -> 01100001

B -> 01000010 b -> 01100010

C -> 01000011 c -> 01100011

. .

. .

Z -> 01011010 z -> 01111010

1. **Convert Upper Case English Alphabet To Lower Case:**

char Upper\_Case\_To\_Lower\_Case(char ch)

{

char mask = ' ';

ch |= mask;

return ch;

}

cout << Upper\_Case\_To\_Lower\_Case(ch);

**Eg:**

ch = 'A'(01000001)

mask = ' '(00100000)

ch | mask = 'a'(01100001)

1. **Convert Lower Case English Alphabet To Upper Case:**

char Upper\_Case\_To\_Lower\_Case(char ch)

{

char mask = '\_';

ch &= mask;

return ch;

}

cout << Upper\_Case\_To\_Lower\_Case(ch);

**Eg:**

ch = 'a'(01100001)

mask = '\_'(11011111)

ch & mask = 'A'(01000001)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Toggle a Bit At n Position\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void toggle(int &num, int pos) //If Bit Is 1 Change To 0 And Vice Versa

{

num ^= (1 << pos);

}

toggle(num, pos)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Check IF Bit At Position n Is Set Or Unset\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool at\_position(int num, int pos)

{

return num & (1 << pos);

}

cout << at\_position(num, pos);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Check IF Number Can Be Expressed As 2^x + 2^y\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool isPowerOfTwo(long long n)

{

return (n && !(n & (n - 1)));

}

long long previousPowerOfTwo(long long n)

{

while (n & n - 1) {

n = n & n - 1;

}

return n;

}

bool checkSum(long long n)//check if n can be expressed as 2^x + 2^y or not

{

if (n == 0 || n == 1)//if value of n is 0 or 1, it can not be expressed

return false;

else if (isPowerOfTwo(n))// if a number is power of 2, it can not be expressed

{

cout << n / 2 << " " << n / 2;

return true;

}

else //if the remainder after subtracting previous power of 2,

{ //is also a power of 2 then, it can be expressed

long long x = previousPowerOfTwo(n);

long long y = n - x;

if (isPowerOfTwo(y))

{

cout << x << " " << y;

return true;

}

}

return false;

}

if (checkSum(n) == false)

cout << "No";

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Number Equal Sum Of Powers 2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long block(long long x)

{

vector<long long> v;// Converting the decimal number // into its binary equivalent.

cout << "Blocks for " << x << " : ";

while (x > 0)

{

v.push\_back(x % 2);

x = x / 2;

}

for (long long i = 0; i < v.size(); i++) // Displaying the output when // the bit is '1' in binary // equivalent of number.

if (v[i] == 1)

{

cout << i;

if (i != v.size() - 1)

cout << ", ";

}

cout << endl;

}

block(x);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Count Of Set Bit In Number\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

long long popcount(long long x) // We Can Use “\_\_builtin\_popcountll(x)” In Gcc Compiler

{ // As Ready Function

long long c; // Library :: #include <cstdlib> && # include <cstdio>

for (c = 0; x != 0; x >>= 1)

if (x & 1)

c++;

return c;

}

cout << Popcount(n) << endl;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ghavanin Mod Giri\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. (a + b) % c = ((a % c) + (b % c)) % c

2. (a \* b) % c = ((a % c) \* (b % c)) % c

3. (a – b) % c = ((a % c) –(b % c)) % c

4. (a / b) % c NOT EQUAL TO((a % c) / (b % c)) % c

5. (a / b) % c = (a \* b^ -1) % c;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Backtrack(Find All Possible subset)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <iostream>

#include <vector>

using namespace std;

void subsetsUtil(vector<long long>& A, vector<vector<long long> >& res, vector<long long>& subset, long long index)

{

for (long long i = index; i < A.size(); i++)

{

subset.push\_back(A[i]);// include the A[i] in subset.

res.push\_back(subset);

subsetsUtil(A, res, subset, i + 1);// move onto the next element.

subset.pop\_back();// exclude the A[i] from subset and triggers //backtracking.

}

}

vector<vector<long long> > subsets(vector<long long>& A) // below function returns the subsets of vector A.

{

vector<long long> subset;

vector<vector<long long> > res;

res.push\_back(subset); // include the null element in the set.

long long index = 0; // keeps track of current element in vector A;

subsetsUtil(A, res, subset, index);

return res;

}

int main()

{

long n, i, j;

vector<long long> array;

cin >> n;

for (i = 0; i < n; i++)

{

cin >> j;

array.push\_back(j);

}

vector<vector<long long> > res = subsets(array);

for (i = 0; i < res.size(); i++)

{

for (j = 0; j < res[i].size(); j++)

cout << res[i][j] << " ";

cout << endl;

}

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Backtrack(Power Set WithOut Repeat)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <algorithm>

#include <vector>

#include <sstream>

#include <string>

vector<string> split(const string &s, char d)

{

vector<string> e;

stringstream ss(s);

string s1;

while (getline(ss, s1, d))

e.push\_back(s1);

return e;

}

void printPowerSet(long long a[], long long n)

{

string s1, subset;

long long i, j, k = pow(2, n);

vector<string> list;

for (i = 0; i < k; i++)

{

subset = "";

for (j = 0; j < n; j++)

if ((i & (1 << j)) != 0)

{

stringstream ss;

ss << a[j];

ss >> s1;

subset += s1 + "|";

}

if (find(list.begin(), list.end(), subset) == list.end())

list.push\_back(subset);

}

i = 0;

for (string subset : list)

{

vector<string> arr = split(subset, '|');

for (string str : arr)

cout << str << " ";

if (i++ != 0)

cout << endl;

}

}

long long a[1000 + 7], n;

printPowerSet(a, n);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Time Complexity\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

