**ICPC NOTES – CRUSADERS 2024**

**1.1 SET INPUT AND OUTPUT**

void setIO() {

freopen("input.in", "r", stdin);

freopen("output.out", "w", stdout);

}

**1.2 IMPORTANT BEFORE START**

int main(){

ios::sync\_with\_stdio(false);

cin.tie(nullptr); // don´t use this on interactive problems

cout << flush; // force display in console

setIO();

}

**1.3 DATA TYPES**

* int, int32\_t: [-231, 231-1] ≅ [-2.1 x 109, 2.1 x 109]
* long long int, int64\_t: [-263, 263-1] ≅ [-9.2 x 1018, 9.2 x 1018]
* unsigned long long int, uint64\_t: [0, 264-1] ≅ [0, 1.8 x 1019]

**1.4 MAXIMUM POSSIBLE TEST CASES BY ALGORITHM COMPLEXITY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Time Complexity** |  | **N** | **Time Complexity** |
| <= 10 | 11 | O(n!) | O(n6) |  | <= 2000 | 2 x 103 | O(n2 log2n) |
| <= 15 – 18 | O(2n x n2) |  | <= 10000 | 1 x 104 | O(n2) |
| <= 18 – 22 | O(2n x n) |  | <= 1000000 | 1 x 106 | O(n log2n) |
| <= 100 | 102 | O(n4) |  | <= 100000000 | 1 x 108 | O(n) |
| <= 400 | 4 x 102 | O(n3) |  |  |  |

**2.1 RECURSIVE BINARY EXPONENTIATION**

int power(int a, int n){

if(n == 0){

return 1;

}else if(n%2 == 0){

long long tmp = power(a, n/2);

return tmp \* tmp % mod;

}else{

long long tmp = power(a, (n-1)/2);

return a \* tmp % mod \* tmp % mod;

}

}

**2.2 ITERATIVE BINARY EXPONENTIATION**

long long power(int a, int b){

long long output = 1;

while(b > 0){

if(b & 1){

output = output \* a % mod;

}

a = (long long)a \* a % mod;

b = b >> 1;

}

return output;

}

**2.3 MODULAR INVERSE**

Note that gcd(a,mod) must be 1 for this algorithm to work.

int inverse(int a) {

return power(a, mod - 2);

}

**2.4 COMBINATIONS USING MODULAR INVERSE**

int comb (int n, int k) {

if (n < 0 || k < 0 || n < k) return 0;

return fac[n] \* inverse(fac[k]) % mod \* inverse(fac[n - k]) % mod;

}

**3.1 GREATEST COMMON DIVISOR (EUCLIDEAN ALGORITHM)**

int gcd(int a, int b){

if(b == 0){

return abs(a);

}else{

return gcd(b, a%b);

}

}

**3.2 LEAST COMMON MULTIPLE**

int lcm(int a, int b){

return a / gcd(a,b) \* b;

}

**4.1 SIEVE OF ERASTÓTENES**

// O(n log(log n)) -> n <= 108

vector<bool> sieve(int n){

vector<bool> isPrime(n+1, true);

isPrime [0] = isPrime [1] = false;

for(int i = 4; i <= n; i += 2){

isPrime [i] = false;

}

for(int i = 3; i\*i <= n; i += 2){

if(isPrime [i]){

for(int j = i\*i; j <= n; j += 2\*i)

isPrime [j] = false;

}

}

return isPrime; // algorithm to obtain the first n primes optimally

}

**5.1 CUSTOM COMPARATOR**

bool customComparison(int a, int b) {

return a < b; // it sorts in ascending order

}

bool customComparisonPair(const pair<int, int>& a, const pair<int, int>& b) {

if (a.first != b.first)

return a.first < b.first;

return a.second > b.second;

}

int main(){

vector<int> vec = { 7, 5, 2, 1, 4, 3 };

vector<pair<int, int>> vec = {{1, 5}, {2, 2}, {1, 2}, {4, 3}, {2, 3}, {1, 6}};

sort(vec.begin(), vec.end(), customComparison);

sort(vec.begin(), vec.end(), customComparisonPair);

}