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: $[-2^{31}, 2^{31}-1] \cong [-2.1 \times 10^{9}, 2.1 \times 10^{9}]$
- long long int, int64_t: $[-2^{63}, 2^{63}-1] \cong [-9.2 \times 10^{18}, 9.2 \times 10^{18}]$
- unsigned long long int, uint64_t: $[0, 2^{64}-1] \cong [0, 1.8 \times 10^{19}]$

1.4 MAXIMUM POSSIBLE TEST CASES BY ALGORITHM COMPLEXITY

N	Time Complexity
<= 10 | 11	O(n!) | $O(n^6)$
<= 15 – 18	$O(2^n \times n^2)$
<= 18 – 22	$O(2^n \times n)$
<= 100 | 10^2	$O(n^4)$
<= 400 | 4 x 10^2	$O(n^3)$

N	Time Complexity
<= 2000 | 2 x 10^3	$O(n^2 \log_2 n)$
<= 10000 | 1 x 10^4	$O(n^2)$
<= 1000000 | 1 x 10^6	$O(n \log_2 n)$
<= 100000000 | 1 x 10^8	O(n)

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

$$C(n,k) = \frac{n!}{k!\,(n-k)!} \qquad C(n,k)\ mod\ x \equiv \left[n! * (k!)^{-1} mod\ x * \left((n-k)!\right)^{-1} mod\ x\right] mod\ x$$

```
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 <= 10^8
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);
}
```