First to Penalty

Contents

1 Template **1**

2 Data structures **1**
2.1 Simplified DSU (Stolen from GGDem) . . . 1
2.2 Disjoint Set Union . . . 1

3 Graphs **2**

4 Math **2**
4.1 Identities . . . 2
4.2 Sieve Of Eratosthenes . . . 2
4.3 Sieve-based Factorization . . . 2

5 Geometry **3**

6 Strings **3**
6.1 Explode by token . . . 3
6.2 Multiple Hashings DS . . . 3
6.3 Permute chars of string . . . 3
6.4 Longest common subsequence . . . 4
6.5 KMP . . . 4

7 Flow **4**

8 Miscellaneous **4**
8.1 Bit Manipulation . . . 4

9 Testing **5**

1 Template

```cpp
#include "bits/stdc++.h"
//assert(x>0) si falla da RTE
using namespace std;
#define endl '\n'
#define DBG(x) cerr<<#x<< "=" << (x) << endl;
#define RAYA cerr<<"======================"<<endl;
#define RAYAS cerr<<"......................"<<endl;
//#define DBG(x) ;
//#define RAYA ;
//#define RAYAS ;

//--------------------SOLBEGIN---------------------------------
int main() {
  ios_base::sync_with_stdio(false); cout.tie(NULL); cin.tie(NULL);
  int tC;

  cin >> tC;
  while (tC--) {

  }

}
//--------------------EOSOLUTION---------------------------------
```

2 Data structures

2.1 Simplified DSU (Stolen from GGDem)

```cpp
int uf[MAXN];
void uf_init(){memset(uf,-1,sizeof(uf));}
int uf_find(int x){return uf[x]<0?x:uf[x]=uf_find(uf[x]);}
bool uf_join(int x, int y){
  x=uf_find(x);y=uf_find(y);
  if(x==y)return false;
  if(uf[x]>uf[y])swap(x,y);
  uf[x]+=uf[y];uf[y]=x;
  return true;
}
```

2.2 Disjoint Set Union

```
class disjSet {
  int* sz;
  int* par;
public:
  int len;
  disjSet(int tam){
        sz = new int[tam + 4]();
        par = new int[tam + 4]();
        len = 0;
        for(int i = 0; i<=tam; i++){
            par[i] = i;
            sz[i] = 1;
            len++;
        }
    }
  int finds(int el){
        if (el == par[el]) return el;
        return par[el] = finds(par[el]);
    }
  void unions(int a, int b){
        a = finds(a);
      b = finds(b);
        if (a == b) return;
        len--;
        //se hace que el gde sea padre del pequeno
        if (sz[a] > sz[b]) swap(a,b);
        par[a] = b;
        sz[b] += sz[a];
    }
  ~disjSet(){
        delete[] size;
        size = nullptr;
        delete[] parent;
        parent = nullptr;
    }
};
```

3 Graphs

4 Math

4.1 Identities

$$C_n = \frac{2(2n-1)}{n+1} C_{n-1}$$
$$C_n = \frac{1}{n+1}\binom{2n}{n}$$
$$C_n \sim \frac{4^n}{n^{3/2}\sqrt{\pi}}$$
$$\sigma(n) = O(\log(\log(n))) \text{ (number of divisors of } n)$$
$$F_{2n+1} = F_n^2 + F_{n+1}^2$$
$$F_{2n} = F_{n+1}^2 - F_{n-1}^2$$
$$\sum_{i=1}^{n} F_i = F_{n+2} - 1$$
$$F_{n+i}F_{n+j} - F_nF_{n+i+j} = (-1)^n F_i F_j$$
(Möbius Inv. Formula) Let $g(n) = \sum_{d|n} f(d)$, then $f(n) = \sum d \mid ng(d)\mu\left(\frac{n}{d}\right)$.

4.2 Sieve Of Eratosthenes

```
#define MAXN 10e6
class soe{
public:
    bitset<MAXN> isPrime;
    soe(){
        for(int i = 3; i<MAXN; i++) isPrime[i] = (i%2);
        isPrime[2] = 1;
        for(int i = 3; i*i<MAXN; i+=2)
            if(isPrime[i])
                for(int j = i*i; j<MAXN; j+=i)
                    isPrime[j] = 0;
    }
};
```

4.3 Sieve-based Factorization

```
#define MAXN 10e6
class soe{
public:
    int smolf[MAXN];
    soe(){
        for(int i = 2; i<MAXN; i++) smolf[i] = (i%2==0?2:i);

        for(int i = 3; i*i<MAXN; i+=2)
```

```
            if(smolf[i]==i)
                for(int j = i*i; j<MAXN; j+=i)
                    smolf[j] = min(smolf[j],smolf[i]);
    }
};
```

5 Geometry

6 Strings

6.1 Explode by token

```
//#include <sstream>

vector<string> explode(string const& s, char delim) {
  vector<string> result;
  istringstream iss(s);
  for (string token; getline(iss, token, delim); )
  {
    result.push_back(move(token));
  }
  return result;
}
```

6.2 Multiple Hashings DS

```
struct multhash{
    unsigned long long int h1,h2,h3;
    unsigned long long int alf[257];
    bool operator < (multhash b) const { // override less than operator
    if (h1 != b.h1) return h1 < other.h1;
    if (h2 != b.h2) return h2 < other.h2;
    return h3 < b.h3;
  }
  bool operator == (multhash b) const { // override equal operator
    return (h1== b.h1 && h2== b.h2 && h3== b.h3)
  }
public:
    string s;
    multhash(){
        h1 = 0; h2 = 0;h3 = 0; s = "";
        for(char l = 'a'; l<='z'; l++) alf[l] = l-'a'+1;
    }
    void innit(){
        unsigned long long int inf,p,op;

        inf = 666666555557777777;
        p = 47;op = 47;
        for(char l: s){
            h1+=(p*alf[l])%inf;
            p*=op;
            p%=inf;
        }

        inf = 986143414027351997;
        p = 53;op = 53;
        for(char l: s){
            h2+=(p*alf[l])%inf;
            p*=op;
            p%=inf;
        }

        inf = 909090909090909091;
        p = 79;op = 79;
        for(char l: s){
            h3+=(p*alf[l])%inf;
            p*=op;
            p%=inf;
        }
    }
};
//VALORES POSIBLES DE INF, MIENTRAS MAS CERCANOS A 10^17 MEJOR
//666666555557777777
//986143414027351997
//974383618913296759
//973006384792642181
//953947941937929919
//909090909090909091
//VALORES PARA P, USAR PRIMOS MAYORES A |Alfabeto|
//31,47,53,61,79
```

6.3 Permute chars of string

```
void permute(string str){
  // Sort the string in lexicographically
  // ascennding order
```

```
  sort(str.begin(), str.end());

  // Keep printing next permutation while there
  // is next permutation
  do {
    cout<<str<<endl;
  } while (next_permutation(str.begin(), str.end()));
}
```

6.4 Longest common subsequence

```
//O(|te|*|pa|)
//cambiar score para otros problemas, str all match = +2, miss/ins/del = -1
//usar char que no este en el alfabeto para denotar del/ins
string te,pa;
long long int ninf = -10e13;
long long int score(char a, char b){
    if(a=='*' || b=='*') return 0;
    if(a==b) return 1;
    return ninf;
}
long long int lcs(){
    long long int** dp;te = "*"+te; pa = "*"+pa;
    long long int res = 0;

    dp = new long long int*[te.size()];
    for(int i = 0; i<te.size(); i++) dp[i] = new long long int[pa.size()]();

    for(int r = 1; r<te.size(); r++){
        for(int c = 1; c<pa.size(); c++){
            dp[r][c] = dp[r-1][c-1]+score(te[r],pa[c]);
            dp[r][c] = max(dp[r][c-1]+score(te[r],'*'),dp[r][c]);
            dp[r][c] = max(dp[r-1][c]+score('*',pa[c]),dp[r][c]);
        }
    }

    return dp[te.size()-1][pa.size()-1];
}
```

6.5 KMP

```
string T,P;
int bt[MAXN];
//O(|Text|+|Pattern|)
void KMPpre(){
    int i = 0,j = 0; bt[0] = -1;
    while(i<P.size()){
        while(j>=0 && P[i]!=P[(j>=0?j:0)]) j = bt[j];
        i++;j++; bt[i] = j;
    }
}
int kmp(){
    int res =0, i = 0, j = 0;
    while(i<T.size()){
        while(j>=0 && T[i] != P[(j>=0?j:0)])  j = bt[j];
        i++; j++;
        if(j==P.size()){//match, do anything
            res++;j = bt[j];
        }
    }
    return res;
}
```

7 Flow

8 Miscellaneous

8.1 Bit Manipulation

```
#include "bits/stdc++.h"
using namespace std;
#define endl '\n'


int main() {
  ios_base::sync_with_stdio(false); cout.tie(NULL); cin.tie(NULL);
  //Se representan bitmasks de 30 a 62 bits
  //usando signed int y signed long long int
  //para evitar problemas con el complemento de dos
  signed int a, b;
  //para multiplicar un numero por dos solo es necesario aplicar un
  // shifteo de sus bits a la izquierda
  a = 1;
  a= a << 3;
  cout << a << endl;
```

```
  //para dividir un numero entre dos es necesario aplicar un
  //shifteo a la derecha
  a = 32;
  a = a >> 3;
  cout << a << endl;
  //para encender el bit n de a, solo hay que igualar a = a | pow(2,n-1)
  //prende el tercer bit
  a = 1;
  b = 1 << 2;
  a = a | b;
  cout << a << endl;
  //para apagar el bit n de a, solo hay que  a &= ~pow(2,n-1)
  //prende el tercer bit
  a = 5;
  b = 1 << 2;
  a &= ~b;
  cout << a << endl;
  //para revisar si el bit n de a esta encendido
  //revisa si el tercer bit esta encendido
  a = 5;
  b = 1 << 2;
  a = a & b;
  cout << (a?"SI":"NO") << endl;
  //para volter el bit n de a, solo hay que igualar a = a ^ pow(2,n-1)
  //apaga el tercer bit
  a = 5;
  b = 1 << 2;
  a = a ^ b;
  cout << a << endl;
  //para obtener el bit menos significativo que esta encendido a& -a
  a = 12;
  cout << log2(a & ((-1) * a))+1 << endl;
  //para prender todos los bits hasta n
  a = (1<<4)-1;
  cout << a << endl;
}
//--------------------EOSOLUTION---------------------------------
```

```
#include "bits/stdc++.h"
using namespace std;
#define endl '\n'
#pragma GCC optimize("O3")
#pragma GCC target("popcnt")

//no usar con visual c++
//solo con g++ like compilers
int main() {
  ios_base::sync_with_stdio(false); cout.tie(NULL); cin.tie(NULL);
  signed long long int a, b, n;
  //Obtain the remainder (modulo) of a when it is divided by n (n is a
      power of 2)
  a = 15; n = 8-1;
  a &= n;
  cout << "a%n, a = 15, n = 2^3" << endl;
  cout << a << endl;
  //Apaga el bit menos significativo de a
  a = 14;
  b = (a & ((-1) * a));
  a &= ~b;
  cout << a << endl;
  //enciende el ultimo cero de a
  a = 9;
  b = ~a;
  b = (b & ((-1) * b));
  a = a | b;
  cout << a<<endl;
  //contar bits encendidos en a
  cout << __builtin_popcount(a)<<endl;
  //checar la paridad de a
  cout << (__builtin_parity(a) ? "IMPAR" : "PAR") << endl;
  //contar leading zeroes en a
  cout << __builtin_clz(a)<<endl;
  //contar 9,trailling zeroes en a
  cout << __builtin_ctz(a)<<endl;
}
//--------------------EOSOLUTION---------------------------------
```

9 Testing