

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
7	Flow	4
8	Miscellaneous	4
8.1	Bit Manipulation	4
9	Testing	5

1 Template

```
1 #include "bits/stdc++.h"
2 //assert(x>0) si falla da RTE
3 using namespace std;
4 #define endl '\n'
5 #define DBG(x) cerr<<#x<< " = " << (x) << endl;
6 #define RAYA cerr<<"===== "<<endl;
7 #define RAYAS cerr<<"..... "<<endl;
8 // #define DBG(x) ;
9 // #define RAYA ;
10 // #define RAYAS ;
11
12 //-----SOLBEGIN-----
13 int main() {
14     ios_base::sync_with_stdio(false); cout.tie(NULL); cin.tie(NULL);
15     int tC;
16
17     cin >> tC;
18     while (tC-- > 0) {
19
20     }
21
22 }
23 //-----EOSOLUTION-----
```

2 Data structures

2.1 Simplified DSU (Stolen from GGDem)

```
1 int uf[MAXN];
2 void uf_init(){memset(uf,-1,sizeof(uf));}
3 int uf_find(int x){return uf[x]<0?x:uf[x]=uf_find(uf[x]);}
4 bool uf_join(int x, int y){
5     x=uf_find(x);y=uf_find(y);
6     if(x==y)return false;
7     if(uf[x]>uf[y])swap(x,y);
8     uf[x]+=uf[y];uf[y]=x;
9     return true;
10 }
```

2.2 Disjoint Set Union

```

1 class disjSet {
2     int* sz;
3     int* par;
4 public:
5     int len;
6     disjSet(int tam){
7         sz = new int[tam + 4]();
8         par = new int[tam + 4]();
9         len = 0;
10        for(int i = 0; i<=tam; i++){
11            par[i] = i;
12            sz[i] = 1;
13            len++;
14        }
15    }
16    int finds(int el){
17        if (el == par[el]) return el;
18        return par[el] = finds(par[el]);
19    }
20    void unions(int a, int b){
21        a = finds(a);
22        b = finds(b);
23        if (a == b) return;
24        len--;
25        //se hace que el gde sea padre del pequeno
26        if (sz[a] > sz[b]) swap(a,b);
27        par[a] = b;
28        sz[b] += sz[a];
29    }
30    ~disjSet(){
31        delete[] size;
32        size = nullptr;
33        delete[] parent;
34        parent = nullptr;
35    }
36 };

```

3 Graphs

4 Math

4.1 Identities

$$C_n = \frac{2(2n-1)}{n+1} C_{n-1}$$

$$C_n = \frac{1}{n+1} \binom{2n}{n}$$

$$C_n \sim \frac{4^n}{n^{3/2} \sqrt{\pi}}$$

$$\sigma(n) = O(\log(\log(n))) \text{ (number of divisors of } n)$$

$$F_{2n+1} = F_n^2 + F_{n+1}^2$$

$$F_{2n} = F_{n+1}^2 - F_{n-1}^2$$

$$\sum_{i=1}^n F_i = F_{n+2} - 1$$

$$F_{n+i} F_{n+j} - F_n F_{n+i+j} = (-1)^n F_i F_j$$

(Möbius Inv. Formula) Let $g(n) = \sum_{d|n} f(d)$, then $f(n) = \sum_{d|n} g(d) \mu\left(\frac{n}{d}\right)$.

4.2 Sieve Of Eratosthenes

```

1 #define MAXN 10e6
2 class soe{
3 public:
4     bitset<MAXN> isPrime;
5     soe(){
6         for(int i = 3; i<MAXN; i++) isPrime[i] = (i%2);
7         isPrime[2] = 1;
8         for(int i = 3; i*i<MAXN; i+=2)
9             if(isPrime[i])
10                 for(int j = i*i; j<MAXN; j+=i)
11                     isPrime[j] = 0;
12     }
13 };

```

4.3 Sieve-based Factorization

```

1 #define MAXN 10e6
2 class soe{
3 public:
4     int smolf[MAXN];
5     soe(){
6         for(int i = 2; i<MAXN; i++) smolf[i] = (i%2==0?2:i);
7
8         for(int i = 3; i*i<MAXN; i+=2)

```

```

9         if(smolf[i]==i)
10             for(int j = i*i; j<MAXN; j+=i)
11                 smolf[j] = min(smolf[j],smolf[i]);
12     }
13 };

```

5 Geometry

6 Strings

6.1 Explode by token

```

1 //include <sstream>
2
3 vector<string> explode(string const& s, char delim) {
4     vector<string> result;
5     istringstream iss(s);
6     for (string token; getline(iss, token, delim); )
7     {
8         result.push_back(move(token));
9     }
10    return result;
11 }

```

6.2 Multiple Hashings DS

```

1 struct multhash{
2     unsigned long long int h1,h2,h3;
3     unsigned long long int alf[257];
4     bool operator < (multhash b) const { // override less than operator
5         if (h1 != b.h1) return h1 < other.h1;
6         if (h2 != b.h2) return h2 < other.h2;
7         return h3 < b.h3;
8     }
9     bool operator == (multhash b) const { // override equal operator
10        return (h1== b.h1 && h2== b.h2 && h3== b.h3)
11    }
12 public:
13     string s;
14     multhash(){
15         h1 = 0; h2 = 0;h3 = 0; s = "";
16         for(char l = 'a'; l<='z'; l++) alf[l] = l-'a'+1;
17     }

```

```

18 void innit(){
19     unsigned long long int inf,p,op;
20
21     inf = 66666655557777777;
22     p = 47;op = 47;
23     for(char l: s){
24         h1+=(p*alf[l])%inf;
25         p*=op;
26         p%=inf;
27     }
28
29     inf = 986143414027351997;
30     p = 53;op = 53;
31     for(char l: s){
32         h2+=(p*alf[l])%inf;
33         p*=op;
34         p%=inf;
35     }
36
37     inf = 909090909090909091;
38     p = 79;op = 79;
39     for(char l: s){
40         h3+=(p*alf[l])%inf;
41         p*=op;
42         p%=inf;
43     }
44 }
45 };
46 //VALORES POSIBLES DE INF, MIENTRAS MAS CERCANOS A 10^17 MEJOR
47 //66666655557777777
48 //986143414027351997
49 //974383618913296759
50 //973006384792642181
51 //953947941937929919
52 //909090909090909091
53 //VALORES PARA P, USAR PRIMOS MAYORES A |Alfabeto|
54 //31,47,53,61,79

```

6.3 Permute chars of string

```

1 void permute(string str){
2     // Sort the string in lexicographically
3     // ascennding order

```

```

4  sort(str.begin(), str.end());
5
6  // Keep printing next permutation while there
7  // is next permutation
8  do {
9      cout<<str<<endl;
10 } while (next_permutation(str.begin(), str.end()));
11 }

```

6.4 Longest common subsequence

```

1  //0(|te|*|pa|)
2  //cambiar score para otros problemas, str all match = +2, miss/ins/del =
   -1
3  //usar char que no este en el alfabeto para denotar del/ins
4  string te,pa;
5  long long int ninf = -10e13;
6  long long int score(char a, char b){
7      if(a=='*' || b=='*') return 0;
8      if(a==b) return 1;
9      return ninf;
10 }
11 long long int lcs(){
12     long long int** dp;te = "*" + te; pa = "*" + pa;
13     long long int res = 0;
14
15     dp = new long long int*[te.size()];
16     for(int i = 0; i<te.size(); i++) dp[i] = new long long int[pa.size()
17         ]();
18
19     for(int r = 1; r<te.size(); r++){
20         for(int c = 1; c<pa.size(); c++){
21             dp[r][c] = dp[r-1][c-1]+score(te[r],pa[c]);
22             dp[r][c] = max(dp[r][c-1]+score(te[r],'*'),dp[r][c]);
23             dp[r][c] = max(dp[r-1][c]+score('*',pa[c]),dp[r][c]);
24         }
25     }
26
27     return dp[te.size()-1][pa.size()-1];
28 }

```

7 Flow

8 Miscellaneous

8.1 Bit Manipulation

```

1  #include "bits/stdc++.h"
2  using namespace std;
3  #define endl '\n'
4
5
6  int main() {
7      ios_base::sync_with_stdio(false); cout.tie(NULL); cin.tie(NULL);
8      //Se representan bitmasks de 30 a 62 bits
9      //usando signed int y signed long long int
10     //para evitar problemas con el complemento de dos
11     signed int a, b;
12     //para multiplicar un numero por dos solo es necesario aplicar un
13     //shifteo de sus bits a la izquierda
14     a = 1;
15     a= a << 3;
16     cout << a << endl;
17     //para dividir un numero entre dos es necesario aplicar un
18     //shifteo a la derecha
19     a = 32;
20     a = a >> 3;
21     cout << a << endl;
22     //para encender el bit n de a, solo hay que igualar a = a | pow(2,n-1)
23     //prende el tercer bit
24     a = 1;
25     b = 1 << 2;
26     a = a | b;
27     cout << a << endl;
28     //para apagar el bit n de a, solo hay que a &= ~pow(2,n-1)
29     //prende el tercer bit
30     a = 5;
31     b = 1 << 2;
32     a &= ~b;
33     cout << a << endl;
34     //para revisar si el bit n de a esta encendido
35     //revisa si el tercer bit esta encendido
36     a = 5;

```

```

37 b = 1 << 2;
38 a = a & b;
39 cout << (a?"SI":"NO") << endl;
40 //para volter el bit n de a, solo hay que igualar a = a ^ pow(2,n-1)
41 //apaga el tercer bit
42 a = 5;
43 b = 1 << 2;
44 a = a ^ b;
45 cout << a << endl;
46 //para obtener el bit menos significativo que esta encendido a& -a
47 a = 12;
48 cout << log2(a & ((-1) * a))+1 << endl;
49 //para prender todos los bits hasta n
50 a = (1<<4)-1;
51 cout << a << endl;
52 }
53 //-----EOSOLUTION-----

```

```

1 #include "bits/stdc++.h"
2 using namespace std;
3 #define endl '\n'
4 #pragma GCC optimize("O3")
5 #pragma GCC target("popcnt")
6
7 //no usar con visual c++
8 //solo con g++ like compilers
9 int main() {
10 ios_base::sync_with_stdio(false); cout.tie(NULL); cin.tie(NULL);
11 signed long long int a, b, n;
12 //Obtain the remainder (modulo) of a when it is divided by n (n is a
    power of 2)
13 a = 15; n = 8-1;
14 a &= n;
15 cout << "a%n, a=%15, n=%2^3" << endl;
16 cout << a << endl;
17 //Apaga el bit menos significativo de a
18 a = 14;
19 b = (a & ((-1) * a));
20 a &= ~b;
21 cout << a << endl;
22 //enciende el ultimo cero de a
23 a = 9;
24 b = ~a;

```

```

25 b = (b & ((-1) * b));
26 a = a | b;
27 cout << a<<endl;
28 //contar bits encendidos en a
29 cout << __builtin_popcount(a)<<endl;
30 //chechar la paridad de a
31 cout << (__builtin_parity(a) ? "IMPAR" : "PAR") << endl;
32 //contar leading zeroes en a
33 cout << __builtin_clz(a)<<endl;
34 //contar 9,trailling zeroes en a
35 cout << __builtin_ctz(a)<<endl;
36 }
37 //-----EOSOLUTION-----

```

9 Testing