

JakeMate14

Contents

1	Busqueda Binaria	1
1.1	LB y UP	1
1.2	Busqueda con numeros reales	1
2	Estructuras de Datos	1
2.1	Segment Tree	1
2.2	Suffix Array	2
3	Grafos	3
3.1	DFS en Grids	3
4	Arboles	3
4.1	Distancia de cada nodo desde la raiz	3
4.2	Distancia de la raiz a cualquier nodo	4
4.3	Diametro de un arbol	4
4.4	Tamaño del subarbol del nodo x	4
5	Math	4
6	DP	4
6.1	Coin Problem	4
6.2	Digitos	5
7	Geometry	6
8	Strings	6
9	Flow	6
10	Other	6
10.1	Template	6

1 Busqueda Binaria

1.1 LB y UP

```
1 //Devuelve el maximo indice tal que v[i]<=x
2 int closestToTheLeft(vector<int>& v, int x) {
3     int l = -1, r = v.size(),m;
4     while (r > l + 1) {
5         m = l+(r - l) / 2;
6         if (v[m] <= x) l = m;
7         else r = m;
8     }
9     return l;
10 }
11
12 //Devuelve el indice del primer elemento v[i]>=x
13 int closestToTheRight(vector<int>& v, int x){
14     int l=-1,r=n,m;
15     while(r>l+1){
16         m = l+(r-l)/2;
17         if(a[m]<x) l=m;
18         else r=m;
19     }
20     return r;
21 }
```

1.2 Busqueda con numeros reales

```
1 //Es asegurado que con 100 funcionara
2 forn(i,100){
3     m = l+(r-l)/2;
4     if(ok(m)) r = m;
5     else l = m;
6 }
7 cout << setprecision(20) << l << "\n";
```

2 Estructuras de Datos

2.1 Segment Tree

```
1 //Para procesar querys de tipo k-esimo es necesario crear un arbol
  binario perfector(llenar con 0's)
2 template<typename T>
```

```

3 struct SegmentTree{
4     int N;
5     vector<T> ST;
6
7     //Creacion a partir de un arreglo O(n)
8     SegmentTree(int N, vector<T> & arr): N(N){
9         ST.resize(N << 1);
10        for(int i = 0; i < N; ++i)
11            ST[N + i] = arr[i]; //Dato normal
12        ST[N + i] = creaNodo(); //Dato compuesto
13        for(int i = N - 1; i > 0; --i)
14            ST[i] = ST[i << 1] + ST[i << 1 | 1]; //Dato normal
15        ST[i] = merge(ST[i << 1] , ST[i << 1 | 1]); //Dato compuesto
16    }
17
18    //Actualizacion de un elemento en la posicion i
19    void update(int i, T value){
20        ST[i += N] = value; //Dato normal
21        ST[i += N] = creaNodo(); //Dato compuesto
22        while(i >= 1)
23            ST[i] = ST[i << 1] + ST[i << 1 | 1]; //Dato normal
24        ST[i] = merge(ST[i << 1] , ST[i << 1 | 1]); //Dato compuesto
25    }
26
27    //query en [l, r]
28    T query(int l, int r){
29        T res = 0; //Dato normal
30        nodo resl = creaNodo(), resr = creaNodo(); //Dato compuesto
31        for(l += N, r += N; l <= r; l >>= 1, r >>= 1){
32            if(l & 1) res += ST[l++]; //Dato normal
33            if(!(r & 1)) res += ST[r--]; //Dato normal
34
35            if(l & 1) resl = merge(resl, ST[l++]); //Dato compuesto
36            if(!(r & 1)) resr = merge(ST[r--], resr); //Dato compuesto
37        }
38        return res; //Dato normal
39        return merge(resl, resr); //Dato compuesto
40    }
41
42    //Para estas querys es necesario que el st tenga el tam de la
43    //siguiente potencia de 2
44    //ll nT = 1;
45    // while(nT<n) nT<<=1;

```

```

45 //vector<int> a(nT,0);
46
47 //Encontrar k-esimo 1 en un st de 1's
48 int Kth_One(int k) {
49     int i = 0, s = N >> 1;
50     for(int p = 2; p < 2 * N; p <= 1, s >= 1) {
51         if(k < ST[p]) continue;
52         k -= ST[p++]; i += s;
53     }
54     return i;
55 }
56
57 //i del primer elemento >= k en todo el arr
58 int atLeastX(int k){
59     int i = 0, s = N >> 1;
60     for(int p = 2; p < 2 * N; p <= 1, s >= 1) {
61         if(ST[p] < k) p++, i += s;
62     }
63     if(ST[N + i] < k) i = -1;
64     return i;
65 }
66
67 //i del primer elemento >= k en [l,fin]
68 //atLeastX(k,l,1,nT)
69 int atLeastX(int x, int l, int p, int s) {
70     if(ST[p] < x or s <= 1) return -1;
71     if((p < 1) >= 2 * N)
72         return (ST[p] >= x) - 1;
73     int i = atLeastX(x, l, p < 1, s >> 1);
74     if(i != -1) return i;
75     i = atLeastX(x, l - (s >> 1), p < 1 | 1, s >> 1);
76     if(i == -1) return -1;
77     return (s >> 1) + i;
78 }
79 };

```

2.2 Suffix Array

```

1 vector<int> sort_cyclic_shifts(string const& s) {
2     int n = s.size();
3     const int alphabet = 256;
4
5     vector<int> p(n), c(n), cnt(max(alphabet, n), 0);

```

```

6   for (int i = 0; i < n; i++)
7       cnt[s[i]]++;
8   for (int i = 1; i < alphabet; i++)
9       cnt[i] += cnt[i-1];
10  for (int i = 0; i < n; i++)
11      p[--cnt[s[i]]] = i;
12  c[p[0]] = 0;
13  int classes = 1;
14  for (int i = 1; i < n; i++) {
15      if (s[p[i]] != s[p[i-1]])
16          classes++;
17      c[p[i]] = classes - 1;
18  }
19
20  vector<int> pn(n), cn(n);
21  for (int h = 0; (1 << h) < n; ++h) {
22      for (int i = 0; i < n; i++) {
23          pn[i] = p[i] - (1 << h);
24          if (pn[i] < 0)
25              pn[i] += n;
26      }
27      fill(cnt.begin(), cnt.begin() + classes, 0);
28      for (int i = 0; i < n; i++)
29          cnt[c[pn[i]]]++;
30      for (int i = 1; i < classes; i++)
31          cnt[i] += cnt[i-1];
32      for (int i = n-1; i >= 0; i--)
33          p[--cnt[c[pn[i]]]] = pn[i];
34      cn[p[0]] = 0;
35      classes = 1;
36      for (int i = 1; i < n; i++) {
37          pair<int, int> cur = {c[p[i]], c[(p[i] + (1 << h)) % n]};
38          pair<int, int> prev = {c[p[i-1]], c[(p[i-1] + (1 << h)) % n]};
39          if (cur != prev)
40              ++classes;
41          cn[p[i]] = classes - 1;
42      }
43      c.swap(cn);
44  }
45  return p;
46 }
47

```

```

48 vector<int> suffix_array_construction(string s) {
49     s += "$";
50     vector<int> sorted_shifts = sort_cyclic_shifts(s);
51     sorted_shifts.erase(sorted_shifts.begin());
52     return sorted_shifts;
53 }
54
55 //Para imprimir el arreglo
56 for(int i,s.size())
57     cout << s.substr(res[i],s.size()-res[i]) << endl;

```

3 Grafos

3.1 DFS en Grids

```

1 void dfs(int x, int y){
2     if(x<0 || y<0 || x>=n || y>=m || vis[x][y] || grafo[x][y] == 0)
3         return;
4
5     vis[x][y] = true;
6
7     dfs(x+1,y);
8     dfs(x,y+1);
9     dfs(x-1,y);
10    dfs(x,y-1);
11
12    return;
13 }

```

4 Arboles

4.1 Distancia de cada nodo desde la raiz

```

1 vector<vector<int>> arbol;
2 vi distInd(n+1);
3
4 void dfs(int nodo, int padre, vi &a){
5     for(auto u: arbol[nodo]){
6         if(u==padre) continue;
7         a[u] = a[nodo]+1;
8         dfs(u,nodo,a);
9     }
10 }

```

```

11
12 dfs(1,0,distInd);

```

4.2 Distancia de la raiz a cualquier nodo

```

1 //Calculo de la distancia mas larga iniciando desde cualquier nodo
2 vector<vector<int>> arbol;
3 vi a(n+1),b(n+1),distInd(n+1);
4
5 void dfs(int nodo, int padre, vi &a){
6     for(auto u: arbol[nodo]){
7         if(u==padre) continue;
8         a[u] = a[nodo]+1;
9         dfs(u,nodo,a);
10    }
11 }
12
13 void distMaximas(){
14     dfs(1,0,distInd);
15     int n1 = max_element(all(distInd))-distInd.begin();
16
17     dfs(n1,0,a);
18     int n2 = max_element(all(a))-a.begin();
19
20     dfs(n2,0,b);
21
22     //La respuesta de cada nodo i es
23     max(a[i],b[i]);
24 }

```

4.3 Diametro de un arbol

```

1 //Es la distancia mas larga entre dos nodos en el arbol
2 vector<vector<int>> arbol;
3
4 void dfs(int nodo, int padre, vi &a){
5     for(auto u: arbol[nodo]){
6         if(u==padre) continue;
7         a[u] = a[nodo]+1;
8         dfs(u,nodo,a);
9     }
10 }
11
12 int diametro(){

```

```

13     vi a(n+1),b(n+1);
14     dfs(1,0,a);
15     int d1 = max_element(all(a))-a.begin();
16
17     dfs(d1,0,b);
18     int d2 = max_element(all(b))-b.begin();
19
20     return max(a[d1],b[d2]);
21 }

```

4.4 Tamaño del subarbol del nodo x

```

1 //Se calcula la cantidad de nodos del subarbol del nodo x
2 const int maxV = 2*1e5+1;
3 vector<vector<int>> arbol;
4 int cantNodos[maxV];
5
6 void dfs(int nodo, int padre){
7     cantNodos[nodo] = 1;
8     for(int u: arbol[nodo]){
9         if(u==padre) continue;
10        dfs(u,nodo);
11        cantNodos[nodo] += cantNodos[u];
12    }
13 }
14
15 dfs(1,0);

```

5 Math

6 DP

6.1 Coin Problem

```

1 int dp[ESTADO_MAX+1];
2 vector<int>coins(ESTADO_MAX+1,INF);
3
4 dp[0] = 0;
5 memset(dp,INF,sizeof(dp));
6 for (int x = 1; x <= n; x++) {
7     value[x] = INF;
8     for (auto c : coins) {
9         if (x-c >= 0) {

```

```

10         dp[x] = min(dp[x], dp[x-c]+1);
11     }
12 }
13 }

```

6.2 Digitos

```

1 res = solve(b) - solve(a-1);
2 vector<int>num;
3 int dp[20][20][2];
4
5 int solve(lli b){
6     num.clear();
7     while(b>0){
8         num.push_back(b%10);
9         b/=10;
10    }
11    reverse(num.begin(), num.end());
12
13    memset(dp, -1, sizeof(dp));
14    lli res = mem(0, 0, 0);
15    return res;
16 }
17
18 //Numeros con a los mas 3 digitos distintos de cero
19 //4, 200000, 10203
20 int mem(int pos, int cant, int goodAll){
21     if(cant>3) return 0;
22     if(pos==num.size()){
23         if(cant<=3) return 1;
24         return 0;
25     }
26
27     int &a = dp[pos][cant][goodAll];
28     if(a!=-1) return a;
29     a = 0;
30
31     int limite = goodAll==0?num[pos]:9;
32     fore(dig,0,limite){
33         int nG = goodAll;
34         int nCant = cant;
35         if(goodAll==0 && dig<limite)    nG=1;
36         if(dig!=0)    nCant++;

```

```

37         if(nCant<=3)    a+=mem(pos+1,nCant,nG);
38     }
39
40     return a;
41 }
42
43 //Numeros donde el digito d ocurre exactamente k veces
44 int call(int pos, int cnt, int f){
45     if(cnt > k) return 0;
46
47     if(pos == num.size()){
48         if(cnt == k)    return 1;
49         return 0;
50     }
51
52     if(DP[pos][cnt][f] != -1)    return DP[pos][cnt][f];
53     int res = 0;
54
55     int LMT;
56
57     if(f == 0)    LMT = num[pos];
58     else        LMT = 9;
59
60     for(int dgt = 0; dgt<=LMT; dgt++){
61         int nf = f;
62         int ncnt = cnt;
63         if(f == 0 && dgt < LMT) nf = 1;
64         if(dgt == d)    ncnt++;
65         if(ncnt <= k)    res += call(pos+1, ncnt, nf);
66     }
67
68     DP[pos][cnt][f] = res;
69     return DP[pos][cnt][f];
70 }

```

7 Geometry

8 Strings

9 Flow

10 Other

10.1 Template

```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 typedef long long int lli;
5 typedef vector<int> vi;
6 #define IO ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
7 #define forn(i,n) for(lli (i)=0; i<n; i++)
8 #define forr(i,a,n) for(lli i=(a); i<n; i++)
9 #define fore(i,a,n) for(lli i=(a); i<=n; i++)
10 #define all(v) v.begin(),v.end()
11 #define borra(s) s.erase(unique(all(s)),s.end())
12 #define YES cout << "YES\n"
13 #define NO cout << "NO\n"
14 #define debug(a) cout << a << "\n"
15
16 void sol(){
17 }
18
19 int main(){IO
20     int t=1;//cin>>t;
21     while(t-->sol());
22 }
```