Team notebook

October 9, 2017

Contents

1 Grafos **1**

1.1 BFS . . . 1

1.2 DFS . . . 1

1.3 Dijkstra . . . 2

1.4 Flood fill . . . 2

1.5 Kruskal . . . 3

1.6 Segment Tree I . . . 3

1.7 Segment Tree II . . . 4

1.8 Strongly Connected Graph . . . 5

1.9 UFDS . . . 7

2 Matemáticas **7**

2.1 Bin to Int (bitset) . . . 7

2.2 Fibonacci . . . 7

2.3 Int to Bin (bitset) . . . 8

2.4 MCD and mcm . . . 8

2.5 OBI Sieve - Prime Factorization . . . 8

2.6 Reverse a number . . . 9

2.7 Sieve CSP - Vlada . . . 9

2.8 String to Int . . . 9

3 Otros **10**

3.1 Map Implementation . . . 10

4 Programación Dinámica **10**

4.1 BitMask . . . 10

4.2 Búsqueda binaria . . . 10

4.3 Exponenciación modular . . . 11

4.4 Knapsack . . . 11

1 Grafos

1.1 BFS

```
#include <iostream>
#include <queue>
#include <cstring>
#include <vector>
using namespace std;
const int MAXN = 100005;
vector < vector <int> > g;
int dist[MAXN];

void bfs(int s) {
       queue <int> q;
       int v;
       q.push(s);
       memset(dist, -1, sizeof dist);
       dist[s] = 0;
       while(!q.empty()) {
              s = q.front();
              q.pop();
              for(int i = 0; i < (int)g[s].size(); ++i) {
                     v = g[s][i];

                     if(dist[v] != -1)
                            continue;
                     dist[v] = dist[s] + 1;
                     q.push(v);
              }
       }
}

int main() {
        int v, e, s, f; // vertices, aristas, start y finish
        cin >> v >> e;
        g.assign(v + 5, vector <int>());
        for(int i = 0; i < e; ++i) {
                cin >> s >> f;
                g[s].push_back(f);
                g[f].push_back(s);
        }
        bfs(1); // llamamos a bfs desde el nodo 1, podemos llamar desde
             cualquier nodo.
        return 0;
}
```

1.2 DFS

```
#include <iostream>
#include <cstring>
#include <vector>
using namespace std;
const int MAXN = 100005;
vector < vector <int> > g;
bool vis[MAXN];

void dfs(int u) {
        vis[u] = true;
        int v;
        for(int i = 0; i < (int)g[u].size(); ++i) {
                v = g[u][i];
                if(vis[v] == false)
                        dfs(v);
        }
}

int main() {
        int v, e, s, f;
        cin >> v >> e;
        g.assign(v + 5, vector <int>());
        memset(vis, false, sizeof vis);
        for(int i = 0; i < e; ++i) {
                cin >> s >> f;
                g[s].push_back(f);
                g[f].push_back(s);
        }
        dfs(1);
        return 0;
}
```

1.3 Dijkstra

```
#include <iostream>
#include <queue>
#include <vector>
#include <algorithm>
using namespace std;
const int inf = (1 << 30);
vector < vector < pair <int,int> > > g;
vector <int> dist;

void dijkstra(int b) {
        priority_queue < pair <int,int> > q;
        pair <int,int> s, v;
        q.push(make_pair(0, b));
        dist[b] = 0;
        while(!q.empty()) {
                s = q.top();
                q.pop();
                for(int i = 0; i < (int)g[s.second].size(); ++i) {
                        v = g[s.second][i];
                        if((v.first * -1) + (s.first * -1) <
                             dist[v.second]) {
                                dist[v.second] = (v.first * -1) +
                                     abs(s.first * -1);
                                q.push(make_pair(dist[v.second] * -1,
                                     v.second));
                        }
                }
        }
}

int main() {
        int v, e, s, f, w;
        cin >> v >> e;
        g.assign(v + 5, vector < pair <int,int> >());
        dist.assign(v + 5, inf);

        for(int i = 0; i < e; ++i) {
                cin >> s >> f >> w;
                g[s].push_back(make_pair(-w, f));
                // g[f].push_back(make_pair(-w, s));
        }
        cin >> s >> f;
        dijkstra(s);
        if(dist[f] == inf)
                cout << "NO\n";
        else
                cout << dist[f] << endl;
        return 0;
}
```

1.4 Flood fill

```
#include <iostream>
using namespace std;
char g[105][105];
int r, c;
int dy[] = {1,1,0,-1,-1,-1,0,1};
int dx[] = {0,1,1,1,0,-1,-1,-1};

void floodfill(int rr, int cc) {
        g[rr][cc] = '-';
        int xx, yy;
        for(int i = 0; i < 8; ++i) {
                xx = dx[i] + rr;
                yy = dy[i] + cc;
                if(xx < 0 || xx >= r || yy < 0 || yy >= c || g[xx][yy] !=
                    '@')
                        continue;
                floodfill(xx, yy);
        }
}

int main() {
        int cnt;
        while(cin >> r >> c, r || c) {
                cnt = 0;
                for(int i = 0; i < r; ++i)
                        for(int j = 0; j < c; ++j)
                                cin >> g[i][j];
                for(int i = 0; i < r; ++i) {
                        for(int j = 0; j < c; ++j) {
                                if(g[i][j] == '@') {
                                        ++cnt;
                                        floodfill(i, j);
                                }
                        }
                }
                cout << cnt << endl;
        return 0;
}
```

1.5 Kruskal

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 1000000
int p[MAX], num_sets;
void initSet(int n)
{
    for (int i = 0; i < n ; i++) p[i] = i;
    num_sets = n;
}
int findSet(int i)
{
    return p[i] == i?i:p[i] = findSet(p[i]);
}
bool isSameSet(int i, int j)
{
    return findSet(i) == findSet(j);
}
void unionSet(int i, int j)
{
    if(!isSameSet(i, j)) num_sets--;
    p[findSet(i)] = findSet(j);
}

int main()
{
    int v, e;
    cin>>v>>e;
    vector<pair<int, pair<int,int> > > K;
    int s, f, w;
    for(int i=0; i<e; i++)
    {
        cin >> s >> f >> w;
        K.push_back(make_pair(w, make_pair(s,f)));
    }
    sort(K.begin(), K.end());
    int mst = 0;
    initSet(v);
    for(int i=0; i<e; i++)
    {
        if(!isSameSet(K[i].second.first, K[i].second.second))
        {
            unionSet(K[i].second.first, K[i].second.second);
            mst += K[i].first;
        }
    }
    cout << "mst = " << mst << endl;
    return 0;
}
```

1.6 Segment Tree I

```
#include <iostream>
#include <string>
#include <climits>
using namespace std;
typedef long long i64;
i64 arr[500050];
i64 tree[4*500050];

void init(int node, int a, int b)
{
        if(a == b)
        {
                tree[node] = arr[a];
                return;
        }
        init(2*node+1,a,(a+b)>>1);
        init(2*node+2,((a+b)>>1)+1,b);
        tree[node] = tree[2*node+1] + tree[2*node+2];
}

i64 query(int node, int a, int b, i64 p, i64 q)
{
        // aca viene el valor neutro que les mencione, para que no afecte
            las operaciones ..
        // si es que es un nodo fuera del rango. cambiar para lo que pide
            el problema
        if(q < a || b < p)
                return 0;

        if(p <= a && b <= q)
                return tree[node];
        // aca esta la operacion. se puede cambiar para sacar maximos,
            minimos asi:
        // return max(query(2*node+1,a,(a+b)>>1,p,q),
            query(2*node+2,((a+b)>>1)+1,b,p,q));
        // return min(query(2*node+1,a,(a+b)>>1,p,q),
            query(2*node+2,((a+b)>>1)+1,b,p,q));
        return query(2*node+1,a,(a+b)>>1,p,q) +
            query(2*node+2,((a+b)>>1)+1,b,p,q);
}

void update(int node, int a, int b, i64 p, i64 v)
{
        if(p < a || b < p)
                return;
        if(a == b)
        {
                tree[node] = v;
                return;
        }
        update(2*node+1, a, (a+b)>>1, p, v);
        update(2*node+2, ((a+b)>>1)+1, b, p, v);
        // aca esta la operacion. se puede cambiar para sacar maximos,
            minimos asi:
        // tree[node] = max(tree[2*node+1], tree[2*node+2]); y para minimo
            cambian max.
        tree[node] = tree[2*node+1] + tree[2*node+2];
}

int main()
{
        ios_base::sync_with_stdio(false);
        cin.tie(0);
        int n;
        i64 a, b;
        string s;
        cin >> n;
        for(int i = 0; i < n; i++) cin >> arr[i];
        init(0,0,n-1);
        while(cin >> s)
        {
                cin >> a >> b;
                if(s == "sum")
                {
                        a--, b--;
                        cout << query(0,0,n-1,a,b) << '\n';
                }
                else
                {
                        a--;
                        update(0,0,n-1,a,b);
                }
        }
        return 0;
}
```

1.7 Segment Tree II

```
#include<bits/stdc++.h>
using namespace std;
typedef unsigned long long ui64;
struct nodo
{
    int maximo, minimo, mcd, suma;
    void unir(const nodo &a, const nodo &b)
    {
        // maximo = max(a.maximo, b.maximo);
        // minimo = min(a.minimo, b.minimo);
        // mcd = __gcd(a.mcd, b.mcd);
        suma = a.suma + b.suma;
    }
    void iniciar(int v)
    {
        maximo = minimo = mcd = suma = v;
    }
};

int I, J, V, POS;
int p[205000];
nodo T[805000];
void init(int N, int L, int R)
{
    if (L == R) T[N].iniciar(p[L]);
    else
    {
        int M = (L+R)/2, A = N*2, B = N*2+1;
        init(A, L, M);
        init(B, M+1, R);
        T[N].unir(T[A], T[B]);
    }
}

void update(int N, int L, int R)
{
    if (L == R)
        p[POS] = V, T[N].iniciar(V);
    else
    {
        int M = (L+R)/2, A = N*2, B = N*2+1;
        if (POS <= M) update(A, L, M);
        else update(B, M+1, R);
        T[N].unir(T[A], T[B]);
    }
}

nodo query(int N, int L, int R)
{
    if (I <= L && R <=J) return T[N];
    else
    {
        int M = (L+R)/2, A = N*2, B = A*2 + 1;
        if (J <= M) return query(A, L, M);
        else if (I > M) return query(B, M+1, R);
        else
        {
            nodo ans;
            ans.unir(query(A, L, M), query(B, M+1, R));
            return ans;
        }
    }
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(false);
    int n, t, op;
    cin >> n;
    for (int i = 0; i < n; ++i) cin >> p[i];
    init(1, 0, n-1);
    cin >> t;
    while (t--)
    {
        cin >> op;
        if (op == 1)
        {
            cin >> I >> J;
            I--;
            J--;
            cout << query(1, 0, n-1).suma << endl;
        }
        else
        {
            cin >> POS >> V;
            POS--;
            update(1, 0, n-1);
        }
    }
}
```

1.8 Strongly Connected Graph

```
#include <iostream>
#include <list>
#include <stack>
using namespace std;

class Graph
{
    int V;    // No. of vertices
    list<int> *adj;  // An array of adjacency lists

    // A recursive function to print DFS starting from v
    void DFSUtil(int v, bool visited[]);
public:
    // Constructor and Destructor
    Graph(int V) { this->V = V; adj = new list<int>[V];}
    ~Graph() { delete [] adj; }

    // Method to add an edge
    void addEdge(int v, int w);

    // The main function that returns true if the graph is strongly
    // connected, otherwise false
    bool isSC();

    // Function that returns reverse (or transpose) of this graph
    Graph getTranspose();
};

// A recursive function to print DFS starting from v
void Graph::DFSUtil(int v, bool visited[])
{
    // Mark the current node as visited and print it
    visited[v] = true;

    // Recur for all the vertices adjacent to this vertex
    list<int>::iterator i;
    for (i = adj[v].begin(); i != adj[v].end(); ++i)
        if (!visited[*i])
            DFSUtil(*i, visited);
}

// Function that returns reverse (or transpose) of this graph
Graph Graph::getTranspose()
{
    Graph g(V);
    for (int v = 0; v < V; v++)
    {
        // Recur for all the vertices adjacent to this vertex
        list<int>::iterator i;
        for(i = adj[v].begin(); i != adj[v].end(); ++i)
        {
            g.adj[*i].push_back(v);
        }
    }
    return g;
}

void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w); // Add w to vs list.
}

// The main function that returns true if graph is strongly connected
bool Graph::isSC()
{
    // St1p 1: Mark all the vertices as not visited (For first DFS)
    bool visited[V];
    for (int i = 0; i < V; i++)
        visited[i] = false;

    // Step 2: Do DFS traversal starting from first vertex.
    DFSUtil(0, visited);

     // If DFS traversal doesnt visit all vertices, then return false.
    for (int i = 0; i < V; i++)
        if (visited[i] == false)
             return false;

    // Step 3: Create a reversed graph
    Graph gr = getTranspose();

    // Step 4: Mark all the vertices as not visited (For second DFS)
    for(int i = 0; i < V; i++)
        visited[i] = false;

    // Step 5: Do DFS for reversed graph starting from first vertex.
    // Staring Vertex must be same starting point of first DFS
    gr.DFSUtil(0, visited);

    // If all vertices are not visited in second DFS, then
    // return false
    for (int i = 0; i < V; i++)
        if (visited[i] == false)
             return false;

    return true;
}

// Driver program to test above functions
int main()
{
    // Create graphs given in the above diagrams
    Graph g1(5);
    g1.addEdge(0, 1);
    g1.addEdge(1, 2);
    g1.addEdge(2, 3);
    g1.addEdge(3, 0);
    g1.addEdge(2, 4);
    g1.addEdge(4, 2);
    g1.isSC()? cout << "Yes\n" : cout << "No\n";

    Graph g2(4);
    g2.addEdge(0, 1);
    g2.addEdge(1, 2);
    g2.addEdge(2, 3);
    g2.isSC()? cout << "Yes\n" : cout << "No\n";

    return 0;
}
```

1.9 UFDS

```
#include <iostream>
#include <vector>
using namespace std;
vector <int> finder(1000000);
int numSets;

void init_set(int n) {
        for(int i = 0; i < n; ++i)
                finder[i] = i;
        numSets = n;
}

bool find_set(int i) {
        return (i == finder[i]) ? i : finder[i] = find_set(finder[i]);
}

bool is_same_set(int i, int j) {
        return find_set(i) == find_set(j);
}

void union_set(int i, int j) {
        finder[find_set(i)] = find_set(j);
}

int main() {
        return 0;
}
```

2 Matemáticas

2.1 Bin to Int (bitset)

```
///Convertir de string binario a long int
#include <iostream>
#include <bitset>

using namespace std;

int main()
{
    string s = "110";
    unsigned long int value = bitset<32>(s).to_ulong(); ///Unsigned Long
        Int tiene 8 bytes = 32 bits
    cout << value << endl;
}
```

2.2 Fibonacci

```
///Fibonacci no recursivo
#include <bits/stdc++.h>

using namespace std;

long long int fibo(long long int n){
    long long int a=1, b=1, c=0;
    if(n==1 or n==2) return 1;
    for(int i=3;i<=n;i++){
        c=a+b;
        a=b;
        b=c;
    }
    return c;
}

int main(){
        for(int i=1;i<200;i++){
        cout<<i<<": "<<fibo(i)%10<<endl;
        }
        return 0;
}
```

2.3 Int to Bin (bitset)

```
///Convertir de entero a binario en string
#include <bits/stdc++.h>

using namespace std;

int main()
{
        long long int num;
    cin>>num;
    string binario=bitset<64> (num).to_string(); ///Bajar el tamao del
         bitset dependiendo del tipo de dato
        cout<<binario;
    return 0;
}
```

2.4 MCD and mcm

```
#include <bits/stdc++.h>

using namespace std;

int MCD (int a,int b) {
        if(b==0) return a;
        return MCD (b,a%b) ; // recursion
}
int mcm (int a,int b) {
        return a*(b/MCD(a,b)) ; // recursion
}

int main () {
        cout<<MCD(4,8)<<endl; //4
        cout<<MCD(10,5)<<endl ; // 5
        cout<<endl;
        cout<<mcm(4,8)<<endl ; //8
        cout<<mcm(10,5)<<endl ; //10
        return 0;
}
```

2.5 OBI Sieve - Prime Factorization

```
///Factorizacion LIBRO DE LA PAZ
#include <bits/stdc++.h>

using namespace std;

int C[10000001];
int N=10000000;

void iniciar_criba(){
        for(int i=0;i<=N;i++) C[i]=i;
        C[0]=C[1]=-1;
        for(int i=2;i*i<=N;i++){
                if(C[i]==i){
                        for(int j=i+i;j<=N;j+=i)
                        C[j]=i;
                }
        }
}

void fp (int X){
        if(X <=1) return ;
        int a=1;
        int Y=X/C[X];
                while(C[X]==C[Y]){
                        a++;
                        Y=Y/C[Y];
                }
                fp(Y);
                cout<<C[X]<<"^"<<a<<endl ;
}

int main () {
        int x ;
        iniciar_criba () ;
        while ( cin > > x ) {
        fp ( x ) ;
        }
```

```
        return 0;
}
```

2.6 Reverse a number

```
///Invertir un nmero
#include <bits/stdc++.h>

using namespace std;

long long int inv(long long int l){
    long long int aux=0;
    while(l>0){
        aux*=10;
        aux+=l%10;
        l/=10;
    }
    return aux;
}

int main(){
    long long int x;
        cout<<inv(x)<<endl;
    return 0;
}
```

2.7 Sieve CSP - Vlada

```
///Criba para encontrar los nmeros primos de 0 a n
#include <bits/stdc++.h>
using namespace std;

const long long int n=1000000;
bool criba[n+1];

void gencriba(){
    memset(criba,true,sizeof(criba));
    criba[0]=criba[1]=false;
    for(long long int i=2;i<=n;++i){
        if(criba[i]){
            for(long long int j=i;j<=n/i;++j){
                criba[j*i]=false;
            }
        }
    }
}

int main()
{
    gencriba();
    int c, i=1;
    cin>>c;
    while(i<=c){
        if(criba[i]) cout<<i<<endl;
        i++;
    }
    return 0;
}
```

2.8 String to Int

```
///String a entero
#include <bits/stdc++.h>

using namespace std;

int main()
{
    stringstream x;
    string s;
    long long int n;
    cin>>s;
    x<<s;
    x>>n;
    cout<<n<<endl;
    return 0;
}
```

3 Otros

3.1 Map Implementation

```
///Implementacin de mapas en C++
#include <iostream>
#include <map>
#include <vector>

using namespace std ;

int main ()
{
        map <char,int> apl;
        cin>>apl['a'] //13
        cin>>apl['b'] //98
        //apl.insert ( make_pair ('a', 13) );
        //apl.insert ( make_pair ('b', 98) );
        cout << apl ['a'] <<"\n";
        cout << apl ['b'] <<"\n";
        //Notese que no existe apl['c'] por lo que se creara y pondra como valor 0
        cout << apl ['c'] <<"\n";
        //Acceso a las llaves y valores mediante iteradores
        for(map <char,int>::iterator it=apl.begin();it!=apl.end();it++) {
                cout<<it->first<<" "<<it->second<<"\n";
        }
}
//Resultado: a 13 b 98 c 0
```

4 Programación Dinámica

4.1 BitMask

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n,sum, res = 0, obj;
    cin >> n >> obj;
    int c[n];
    for (int i = 0; i < n; ++i) cin >> c[i];
    for (int mask = 0; mask <= (1<<n)-1; ++mask)
    {
        sum = 1;
        for (int i = 0; i < n; ++i)
```

```
           if (mask&(1<<i)) sum += c[i];
        if (sum == obj)
        {
           res++;
        }
    }
    cout << res << endl;

    return 0;
}
```

4.2 Búsqueda binaria

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
       vector <int> v;
       for(int i = 1; i <= 9; ++i)
              v.push_back(i);
       int lo = 0, hi = 9, mid;
       int x = 6; // vamos a buscar el 6.
       while(hi - lo > 1) {
              mid = lo + (hi - lo) / 2;
              cerr << "middle " << mid << endl;
              if(v[mid] > x)
                     hi = mid;
              else
                     lo = mid;
       }
       cout << x << " esta en la posicion " << lo << endl;
       return 0;
}
```

4.3 Exponenciación modular

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ui64;
const ui64 mod = 1000000007;
ui64 b_pow(ui64 a, ui64 b, ui64 m);
int main()
{
    ui64 n, e;
    while (cin >> n >> e && n != 0)
    {
        cout << b_pow(n, e, mod) << endl;
    }
    return 0;
}

/// Exponente - Base
ui64 b_pow(ui64 a, ui64 b, ui64 mod)
{
    ui64 res = 1;
    while(b > 0)
    {
        if((b&1) == 1) res=(a*res)%mod;
        b >>= 1;
        a = ((a%mod)*(a%mod))%mod;
    }
    return res;
}
```

4.4 Knapsack

```
#include <bits/stdc++.h>
using namespace std;
int mochila(int i, int c);
int maxn(int a, int b) { return (a > b) ? a : b; }
int g[100], p[100];
int mem[100][100];
int n;
int main()
{
    int w;
    memset(mem, -1, sizeof(mem));
    cin >> n >> w;
    for (int i = 0; i < n; ++i) cin >> g[i] >> p[i];
    cout << mochila(0, w);
    return 0;
}
```

```
int mochila(int i, int c)
{
    int ans1 = 0, ans2;
    if (i == n) return 0;
    if (mem[i][c] != -1) return mem[i][c];
    if (p[i] <= c) ans1 = mochila(i+1, c-p[i]) + g[i];
    ans2 = mochila(i+1, c);
    return mem[i][c] = maxn(ans1, ans2);
}
```