Team notebook

Javalieron

August 10, 2018

Contents

1 DP **1**

1.1 Knapsack-BottomTop 1

1.2 Knapsack-TopBottom 2

1.3 Knapsack-TopBottom 3

1.4 Knpsack-BottomTop 4

2 DS **5**

2.1 lazySegtree 5

2.2 simpleSegtree 6

3 Geometria **7**

3.1 convexhull 7

3.2 lines 7

3.3 points 8

4 Grafos **8**

4.1 MaxFlow-Dinic 8

4.2 hopcroft-karp 9

4.3 kosaraju 10

5 MISC **11**

5.1 Difference-Equations 11

5.1.1 Homogeneous Linear Equations 11

5.1.2 Non-Homogeneous Linear Equations 12

5.1.3 Non-linear Equations 12

5.2 Extended-euclides 12

5.3 matrix-operations 13

6 Strings **14**

6.1 LongestCommonSubsequence 14

6.2 pi-function 15

6.3 z-function 15

7 utilities **15**

7.1 linux-utilities 15

1 DP

1.1 Knapsack-BottomTop

```
package Algoritmos;


import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

public class Knapsack_BottonTop {

        //Maximo numero de elementos
        static int N;
        //Tamanho de la maleta
        static int S;

        static int DP [][];
        static List<Obj> elem;

        static class Obj{
                int tam;
                int val;

                public Obj(int pt, int pv) {
                        tam = pt;
                        val = pv;
                }
        }
        static int max(int a, int b) {
                return Math.max(a, b);
        }
        static void dp() {
                Obj o;
                int t,v;
                for(int j=0; j<=S; j++) {
                        for(int i =0; i<=N; i++) {
                                if(j== 0 || i==0) {
                                        DP[i][j]=0;
                                }
                                else {
                                        o = elem.get(i-1);
                                        t = o.tam; v = o.val;
                                        if(t>j) {
                                                DP[i][j] = DP[i-1][j];
                                        }
                                        else {
                                                DP[i][j] = max(DP[i-1][j],
                                                    DP[i-1][j-t]+v);
                                        }

                                }

                        }
                }
        }

        public static void main(String[] args) throws Exception {
                BufferedReader bf = new BufferedReader(new
                    InputStreamReader(System.in));
                String[] data = bf.readLine().split(" ");
                S = Integer.parseInt(data[0]);
                N = Integer.parseInt(data[1]);
                DP = new int[N+1][S+1];
                elem = new ArrayList<>(N);
                for(int i =0; i<N; i++) {
                        data = bf.readLine().split(" ");
                        Obj o = new Obj(Integer.parseInt(data[0]),
                            Integer.parseInt(data[1]));
                        elem.add(o);
                }
                dp();
                int rta = DP[N][S];
                System.out.println(rta);


        }
}
```

1.2 Knapsack-TopBottom

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

using namespace std;


//definir segn problema

const int N_MAX = 10005;

const int S_MAX = 10005;


int DP[N_MAX+1][S_MAX+1];


struct Obj{

    int tam;

    int val;

}elem[N_MAX];



int dp(int n, int s){

    if(DP[n][s]!=-1){
       return DP[n][s];

    }

    else if(n == 0 || s == 0){

      return DP[n][s]=0;

    }

    else{

        Obj o = elem[n-1];

        if(s-o.tam<0){

            return DP[n][s] = dp(n-1,s);

        }

        else{

            return DP[n][s] = max(dp(n-1,s),dp(n-1,s-o.tam)+o.val);

        }

    }

}



int main(){

    int N,S,rta;

    scanf("%d %d", &S, &N);

    for(int i = 0; i<N; i++){

       Obj o;

       scanf("%d %d", &o.tam, &o.val );

       elem[i] =o;

    }

    memset(DP,-1,sizeof(DP));

    rta = dp(N,S);

    printf("%d",rta);

    return 0;

}
```

1.3 Knapsack-TopBottom

```
package Algoritmos;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;


public class Knapsack_TopBottom {

        //Maximo numero de elementos
        static int N;
        //Tamanho de la maleta
        static int S;

        static int DP [][];
        static List<Obj> elem;

        static class Obj{
                int tam;

                int val;

                public Obj(int pt, int pv) {
                        tam = pt;
                        val = pv;
                }
        }
        static int max(int a, int b) {
                return Math.max(a, b);
        }
        static void clearMat() {
                for(int j=0; j<=S; j++) {
                        for(int i=0; i<=N; i++) {
                                DP[i][j]=-1;
                        }
                }
        }
                static int dp(int n,int s) {
                        if(DP[n][s]!=-1) {
                                return DP[n][s];
                        }
                        else if(n== 0 || s == 0) {
                                return DP[n][s] =0;
                        }
                        else {
                                Obj o = elem.get(n-1);
                                if(s<o.tam) {
                                        return DP[n][s] = dp(n-1,s);
                                }
                                else {
                                        int a = dp(n-1,s);
                                        int t = o.tam;
                                        int b = dp(n-1,s-t);
                                        int v = o.val;
                                        return DP[n][s] = max(a,b+v);
                                }
                        }
                }

        public static void main(String[] args) throws Exception {
                BufferedReader bf = new BufferedReader(new
                    InputStreamReader(System.in));
                String[] data = bf.readLine().split(" ");
                S = Integer.parseInt(data[0]);
                N = Integer.parseInt(data[1]);
                DP = new int[N+1][S+1];
                elem = new ArrayList<>(N);
                for(int i =0; i<N; i++) {
                        data = bf.readLine().split(" ");
                        Obj o = new Obj(Integer.parseInt(data[0]),
                            Integer.parseInt(data[1]));
                        elem.add(o);
                }
                clearMat();
                int rta = dp(N,S);
                System.out.println(rta);


        }
}
```

1.4 Knpsack-BottomTop

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

using namespace std;

//Ajusar  segn  problema

const int N_MAX = 100000;

const int S_MAX = 100000;

int N,S;
int DP [S_MAX+1];

struct Obj{

    int tam;

    int val;

}elem[N_MAX];


void dp(){

    Obj o;

    int t,v;

    for(int i=0; i<N; i++){

        for(int j=S; j>0; j--){
            o = elem[i];

            t = o.tam;

            v = o.val;

            if(j-t>=0){

                DP[j] = max(DP[j],DP[j-t]+v);

            }

        }

    }

}

int main(){

    scanf("%d %d", &S, &N);

    for(int i=0; i<N; i++){

        Obj o;

        scanf("%d %d", &o.tam, &o.val);

        elem[i]=o;

    }

    memset(DP, 0, sizeof(DP));

    dp();

    printf("%d", DP[S]);

}
```

2 DS

2.1 lazySegtree

```
#include <bits/stdc++.h>
using namespace std;
#define INF 1e9
const int MAXN = 1e5+5;
int a[MAXN],t[4*MAXN],lazy[4*MAXN];
//funcion para construir el segtree
void build(int v,int tl,int tr){
        memset(lazy,0,sizeof(lazy));
        if(tl==tr)
                t[v]=a[tl];
        else{
                int tm = (tl+tr)/2;
                build(v*2,tl,tm);
                build(v*2+1,tm+1,tr);
                t[v]=max(t[v*2],t[v*2+1]);

        }

}
//funcion para propagar el valor a los hijos del nodo.
void push(int v){
        t[v*2]=lazy[v];
        lazy[v*2]=lazy[v];
        t[v*2+1]=lazy[v];
        lazy[v*2+1]=lazy[v];
        lazy[v]=0;
}
//funcion para hacer update de un rango con un valor dado.
void update(int v,int tl, int tr, int l, int r,int val){
        if(l>r)
                return;
        if(l==tl&&tr==r){
                t[v]=val;
                lazy[v]=val;
        }
        else{
                //solo es necesario propagar el valor si existe un valor
                    lazy guardado.
                if(lazy[v]!=0)
                        push(v);
                int tm=(tl+tr)/2;
                update(v*2,tl,tm,l,min(tm,r),val);
                update(v*2+1,tm+1,tr,max(l,tm+1),r,val);
                t[v]=max(t[v*2],t[v*2+1]);
        }
}
//funcion para realizar range query.
int query(int v, int tl, int tr, int l, int r){
        if(l>r)
                return -INF;
        if(tl==l&&tr==r){
                return t[v];

        }
        if(lazy[v]!=0)
                push(v);
        int tm = (tl+tr)/2;
        return
            max(query(v*2,tl,tm,l,min(r,tm)),query(v*2+1,tm+1,tr,max(tm+1,l),r));
}
int n,m,i,j;
int main(){
        while(scanf("%d%d",&n,&m)==2&&n+m){
                for(int i = 0; i<n; i++){
                        a[i]=i;
                }
                build(1,0,n-1);
                printf("%d\n",query(1,0,n-1,0,n-1));
                update(1,0,n-1,n/2,n-1,0);
                printf("%d\n",query(1,0,n-1,0,n-1));
        }
        return 0;
}
```

2.2 simpleSegtree

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 1e5+5;
int a[MAXN],t[4*MAXN];
void build(int v, int tl, int tr){
        if(tl==tr)
                t[v]=a[tl];
        else{
                int tm = (tl+tr)/2;
                build(v*2,tl,tm);
                build(v*2+1,tm+1,tr);
                //depende la operacin a realizar.
                t[v]=t[v*2]+t[v*2+1];
        }
}
int get(int v, int tl, int tr, int l, int r){
        if(l>r)
                //retornar valor neutro de la opracin.
                return 0;
        if(tl==l&&tr==r)
                return t[v];
        int tm = (tl+tr)/2;
        return get(v*2,tl,tm,l,min(r,tm))+get(v*2,tm+1,tl,max(tm+1,l),r);

}
void update(int v,int tl, int tr, int pos, int new_val){
        if(tl==tr)
                t[v]=new_val;
        else{
                int tm = (tl+tr)/2;
                if(pos<=tm)
                        update(v*2,tl,tm,pos,new_val);
                else
                        update(v*2+1,tm+1,tr,pos,new_val);
                t[v]=t[v*2]+t[v*2+1];
        }
}
int main(){
        int n=100,val=0,pos=5,i=2,j=3;
        //leer arreglo
        //construir el segtree
        build(1,0,n-1);
        int res = get(1,0,n-1,i,j);
        update(1,0,n-1,pos,val);
        res = get(1,0,n-1,0,10);
        printf("%d\n",res);


}
```

3 Geometria

3.1 convexhull

```
#include <bits/stdc++.h>
using namespace std;
#define P(p) const point &p
#define L(p0, p1) P(p0), P(p1)
#define C(p0, r) P(p0), double r
#define PP(pp) pair<point,point> &pp
#define EPS 1e-9
#define MAXN 200005
// se puede definir un punto como un numero complejo
typedef complex<double> point;
// dot product a.x*b.x+a.y*b.y
double dot(P(a), P(b)) { return real(conj(a) * b); }
// cross product a.x*b.y-b.x*a.y
double cross(P(a), P(b)) { return imag(conj(a) * b); }
double cross(P(a),P(b),P(c)){ return cross(b-a,c-a);}
double ccw(P(a), P(b), P(c)) { return cross(b - a, c - b); }
point hull[MAXN];
//convex hull
bool cmp(const point &a, const point &b) {
  return abs(real(a) - real(b)) > EPS ?
    real(a) < real(b) : imag(a) < imag(b); }
int convex_hull(vector<point> p) {
  int n = (int)p.size();
  sort(p.begin(), p.end(), cmp);
  int h = -1;
  for(int i = 0; i<n; i++){
        while(h>=1&&cross(hull[h-1],hull[h],p[i])<=0)--h;
        hull[++h]=p[i];
  }
  int th = h;
  for(int i = n-2; i>=0; i--){
        while(h>th&&cross(hull[h-1],hull[h],p[i])<=0)--h;
        hull[++h]=p[i];
  }
  return h;
}
double dist(point a, point b){return sqrt(dot(b-a,b-a));}
//distancia de el punto p al segmento formado por(a,b)
double dist2(point p,point a, point b){
        point v1 = b-a, v2 = p-a;
        return fabs(cross(v1,v2))/dist(a,b);
}

int main(){
        vector<point>v;
        int h = convex_hull(v);
        int q = 1;
        double ans= 1e10;
        //rotating calipers O(n)
        for(int i = 0; i<h; i++){
                while(cross(hull[i],hull[(i+1)%h],hull[(q+1)%h])>cross(hull[i],hul
                        q = (q+1)%h;
                ans = min(ans,dist2(hull[q],hull[i],hull[(i+1)%h]));

        }
        printf("%lf\n",ans);
        return 0;
}
```

3.2 lines

```
#include <bits/stdc++.h>
using namespace std;
#include "points.cpp"
bool collinear(L(a, b), L(p, q)) {
  return abs(ccw(a, b, p)) < EPS && abs(ccw(a, b, q)) < EPS; }
bool parallel(L(a, b), L(p, q)) {
  return abs(cross(b - a, q - p)) < EPS; }
point closest_point(L(a, b), P(c), bool segment = false) {
  if (segment) {
    if (dot(b - a, c - b) > 0) return b;
    if (dot(a - b, c - a) > 0) return a;
  }
  double t = dot(c - a, b - a) / norm(b - a);
  return a + t * (b - a); }
double line_segment_distance(L(a,b), L(c,d)) {
  double x = INFINITY;
  if (abs(a - b) < EPS && abs(c - d) < EPS) x = abs(a - c);
  else if (abs(a - b) < EPS)
    x = abs(a - closest_point(c, d, a, true));
  else if (abs(c - d) < EPS)
    x = abs(c - closest_point(a, b, c, true));
  else if ((ccw(a, b, c) < 0) != (ccw(a, b, d) < 0) &&
      (ccw(c, d, a) < 0) != (ccw(c, d, b) < 0)) x = 0;
  else {
    x = min(x, abs(a - closest_point(c,d, a, true)));
    x = min(x, abs(b - closest_point(c,d, b, true)));
    x = min(x, abs(c - closest_point(a,b, c, true)));
    x = min(x, abs(d - closest_point(a,b, d, true)));
  }
  return x; }
bool intersect(L(a,b), L(p,q), point &res, bool seg=false) {
  // NOTE: check parallel/collinear before
  point r = b - a, s = q - p;
  double c = cross(r, s),
        t = cross(p - a, s) / c, u = cross(p - a, r) / c;
  if (seg &&
      (t < 0-EPS || t > 1+EPS || u < 0-EPS || u > 1+EPS))
    return false;
  res = a + t * r;
return true; }
```

3.3 points

```
#include <bits/stdc++.h>
using namespace std;
#define pi 2*acos(0)
#define P(p) const point &p
#define L(p0, p1) P(p0), P(p1)
#define C(p0, r) P(p0), double r
#define PP(pp) pair<point,point> &pp
#define EPS 1e-9
typedef complex<double> point;
double dot(P(a), P(b)) { return real(conj(a) * b); }
double cross(P(a), P(b)) { return imag(conj(a) * b); }
point rotate(P(p), double radians = pi / 2,
            P(about) = point(0,0)) {
  return (p - about) * exp(point(0, radians)) + about; }
point reflect(P(p), L(about1, about2)) {
  point z = p - about1, w = about2 - about1;
  return conj(z / w) * w + about1; }
point proj(P(u), P(v)) { return dot(u, v) / dot(u, u) * u; }
point normalize(P(p), double k = 1.0) {
  return abs(p) == 0 ? point(0,0) : p / abs(p) * k; }
double ccw(P(a), P(b), P(c)) { return cross(b - a, c - b); }
bool collinear(P(a), P(b), P(c)) {
  return abs(ccw(a, b, c)) < EPS; }
double angle(P(a), P(b), P(c)) {
  return acos(dot(b - a, c - b) / abs(b - a) / abs(c - b)); }
double signed_angle(P(a), P(b), P(c)) {
  return asin(cross(b - a, c - b) / abs(b - a) / abs(c - b)); }
double angle(P(p)) { return atan2(imag(p), real(p)); }
point perp(P(p)) { return point(-imag(p), real(p)); }
double progress(P(p), L(a, b)) {
  if (abs(real(a) - real(b)) < EPS)
    return (imag(p) - imag(a)) / (imag(b) - imag(a));
else return (real(p) - real(a)) / (real(b) - real(a)); }
```

4 Grafos

4.1 MaxFlow-Dinic

```
#include <bits/stdc++.h>
using namespace std;
//algunas definiciones y constantes
typedef long long F;
const int MAXV = 10005;
const int MAXE = 60005;
const F F_INF = 100000000000000;
//clase para manejar de forma eficiente el maxflow
class MaxFlow {
public:
    int V, E;

    MaxFlow(int V) : V(V), E(0) {
        memset(start, -1, sizeof(start));
    }

    void add_edge(int x, int y, F c) {
        cap[E] = c; flow[E] = 0; v[E] = y; next[E] = start[x]; start[x] =
            E; ++E;
        //para arcos dirigidos c = 0 aca abajo
        cap[E] = c; flow[E] = 0; v[E] = x; next[E] = start[y]; start[y] =
            E; ++E;
    }

    bool BFS(int, int);
    F DFS(int, int, F);
    F maxflow(int, int);

    vector<pair<int, int>> get_flows();

private:
    int start[MAXV], next[MAXE], v[MAXE];
    int used[MAXV], level[MAXV];
    F cap[MAXE], flow[MAXE];
};

vector<pair<int, int>> MaxFlow::get_flows() {
    vector<pair<int, int>> ret;
    for (int i = 0; i < V; ++ i)
        for (int j = start[i]; j != -1; j = next[j])
            if (flow[j] > 0)
                ret.push_back({i, v[j]});
    return ret;
}
//funcion para contruir el grafo de nivel
bool MaxFlow::BFS(int s, int t) {
    memset(level, -1, sizeof(level));
    queue<int> q;
    q.push(s); level[s] = 0;
    while (!q.empty()) {
        int x = q.front(); q.pop();
        for (int i = start[x]; i != -1; i = next[i])
            if (level[v[i]] == -1 && cap[i] > flow[i]) {
                q.push(v[i]);
                level[v[i]] = level[x] + 1;
            }
    }
    return (level[t] != -1);
}

//funcion para hallar el blocking flow
F MaxFlow::DFS(int s, int t, F f) {
    if (s == t) return f;
    for (int &i = used[s]; i != -1; i = next[i])
        //if (level[v[i]] > level[s] && cap[i] > flow[i]) { // should be
            same
        if (level[v[i]] == level[s] + 1 && cap[i] > flow[i]) {
            F temp = DFS(v[i], t, min(f, cap[i] - flow[i]));
            if (temp > 0) {
                flow[i] += temp; flow[i^1] -= temp;
                return temp;
            }
        }
    return 0;
}
// funcion para hallar el maxflow entre s = source y t = target
F MaxFlow::maxflow(int s, int t) {
    while (BFS(s, t)) {
        for (int i = 0; i < V; ++ i)
            used[i] = start[i];
        while (DFS(s, t, F_INF) != 0);
    }
    F ret = 0;
    for (int i = start[s]; i != -1; i = next[i])
        ret += flow[i];
    return ret;
}
```

4.2 hopcroft-karp

```
#include <bits/stdc++.h>
using namespace std;
//hopcorft karp es decir dinic optimizado para redes de cap 1.
// documentacion de las funciones en dinic
const int MAXN1 = 50005;
const int MAXN2 = 50005;
const int MAXM = 150005;

int n1, n2, edges, last[MAXN1], prevs[MAXM], head[MAXM];
int matching[MAXN2], dist[MAXN1], Q[MAXN1];
bool used[MAXN1], vis[MAXN1];

void init(int _n1, int _n2) {
    n1 = _n1;
    n2 = _n2;
    edges = 0;
    fill(last, last + n1, -1);
}

void addEdge(int u, int v) {
    head[edges] = v;
    prevs[edges] = last[u];
    last[u] = edges++;
}
```

```
void bfs() {
    fill(dist, dist + n1, -1);
    int sizeQ = 0;
    for (int u = 0; u < n1; ++u) {
        if (!used[u]) {
            Q[sizeQ++] = u;
            dist[u] = 0;
        }
    }
    for (int i = 0; i < sizeQ; i++) {
        int u1 = Q[i];
        for (int e = last[u1]; e >= 0; e = prevs[e]) {
            int u2 = matching[head[e]];
            if (u2 >= 0 && dist[u2] < 0) {
                dist[u2] = dist[u1] + 1;
                Q[sizeQ++] = u2;
            }
        }
    }
}

bool dfs(int u1) {
    vis[u1] = true;
    for (int e = last[u1]; e >= 0; e = prevs[e]) {
        int v = head[e];
        int u2 = matching[v];
        if (u2 < 0 || (!vis[u2] && dist[u2] == dist[u1] + 1 && dfs(u2))) {
            matching[v] = u1;
            used[u1] = true;
            return true;
        }
    }
    return false;
}

int maxMatching() {
    fill(used, used + n1, false);
    fill(matching, matching + n2, -1);
    for (int res = 0;;) {
        bfs();
        fill(vis, vis + n1, false);
        int f = 0;
        for (int u = 0; u < n1; ++u)
            if (!used[u] && dfs(u))
                ++f;
        if (!f)
            return res;
        res += f;
    }
}
int main(){
        //iniciar el grafo
        init(5,5);
        //agregar arcos
        addEdge(1, 2);
        addEdge(1, 3);
        addEdge(2, 1);
        addEdge(3, 2);
        addEdge(4, 2);
        addEdge(4, 4);
        printf("%d\n",maxMatching());
        //recuperar match maximo
        for(int i=1; i<=4; i++)
                printf("match %d %d\n",matching[i],i);
        return 0;
}
```

4.3 kosaraju

```
#include <bits/stdc++.h>
using namespace std;
//choose MAXN according to the problem.
const int MAXN = 100005;
vector<int> g[MAXN],gr[MAXN];
bool vis[MAXN];
stack<int> tp;
int n,m;
int scc = 0;
void dfs(int x){
        vis[x]=1;
        for(vector<int>::iterator it = g[x].begin(); it!=g[x].end(); ++it){
                int y = *it;
                if(!vis[y])
                        dfs(y);
        }
        tp.push(x);
}
```

```
void dfs2(int x){
        vis[x]=1;
        for(vector<int>::iterator it = gr[x].begin(); it!=gr[x].end();
            ++it){
                int y = *it;
                if(!vis[y])
                        dfs2(y);
        }
}
int main(){
        //read graph.
        //kosaraju
        memset(vis,0,sizeof(vis));
        for(int i = 0; i<n; i++)
                if(!vis[i])
                        dfs(i);
        memset(vis,0,sizeof(vis));
        while(!tp.empty()){
                int x = tp.top();
                tp.pop();
                if(!vis[x]){
                        scc++;
                        dfs2(x);
                        //do extra things like graph condensation.
                }
        }
        return 0;
}
```

5 MISC

5.1 Difference-Equations

F is a vector space defined as,

$$\begin{aligned} F(c\cdot x+y) &= c\cdot F(x)+F(y) \\ (F+G)x &= F(x)+G(x) \\ (c\cdot F)x &= c\cdot F(x) \\ (F\cdot G)x &= F(Gx) \\ F(G+H) &= FG+FH=(G+H)F \end{aligned}$$

Where, I is the identity function, and E is the *advance* operator and the nullifier is defined as,

$$\begin{aligned} P(\lambda) &= (\lambda-\lambda_0)^m \\ P(E)x=0 &\Rightarrow (E-\lambda_0)^m x=0 \\ (E-\lambda_0)^m &\text{ is the nullifier of } n^j\lambda_0^n, 0\leq j<m \\ A(E) \text{ is the nullifier of } a_n &\Rightarrow A(E) \text{ is the nullifier of } c\cdot a_n \end{aligned}$$

5.1.1 Homogeneous Linear Equations

Different roots,

$$\begin{aligned} a_0=0;a_1 &= 1;n\geq 2 \\ a_n-5a_{n-1}+6a_{n-2} &= 0 \\ a_{n+2}-5a_{n+1}+6a_n &= 0 \\ (E^2-5E-6I)a &= 0 \\ (E-3)(E-2)a &= 0 \\ a_n &= A\cdot 3^n+B\cdot 2^n \\ a_0 &= 0=A+B \\ a_1 &= 1=3A+2B \\ A=1;B &= -1 \\ a_n &= 3^n-2^n \end{aligned}$$

Same roots,

$$\begin{aligned} a_0=0;a_1 &= 1;n\geq 0 \\ a_{n+2}-4a_{n+1}+4a_n &= 0 \\ (E^2-4E+4I)a &= 0 \\ (E-2)^2 a &= 0 \\ a_n &= A\cdot 2^n+Bn\cdot 2^n \\ a_0 &= 0=A \\ a_1 &= 1=2A+2B \\ A=0;B &= 1/2 \\ a_n &= \frac{n}{2}\cdot 2^n=n2^{n-1} \end{aligned}$$

5.1.2 Non-Homogeneous Linear Equations

$$\begin{aligned}
a_0 &= 1; a_1 = 3; n \geq 0 \\
a_{n+1} - 2a_n &= 1 \\
(E - 2I)a &= 1 \\
(E - 1) \text{ is the nullifier of } n^0 1^n &= 1 \\
(E - 1)(E - 2)a &= 0 \\
a_n &= A + B \cdot 2^n \\
a_0 &= 1 = A + B \\
a_1 &= 3 = A + 2B \\
A = -1; B &= 2 \\
a_n &= -1 + 2 \cdot 2^n = 2^{n+1} - 1
\end{aligned}$$

$$\begin{aligned}
a_{n+1} + a_n &= 3n + 2^n + 4 \\
(E + 1I)a &= 3n + 2^n + 4 \\
(E - 1)^2 &\text{ is the nullifier of } 2n \text{ and } 4 \\
(E - 2) &\text{ is the nullifier of } 2^n \\
(E - 1)(E - 2)(E + 1)a &= 0 \\
a_n &= A + Bn + C \cdot 2^n + D(-1)^n
\end{aligned}$$

5.1.3 Non-linear Equations

Merge sort example,

$$\begin{aligned}
M(1) &= 0; n > 1 \\
M(n) &= 2M(n/2) + n - 1 \\
x_0 = 1; x_k = n; x_{k+1} &= n/2; x_k = 2x_{k-1} \\
x_{k+1} - 2x_k &= 0 \\
x_k &= \alpha \cdot 2^k; x_0 = 1 = \alpha \\
x_k &= 2^k \\
2^k &= n \\
k &= log(n) \\
y_k &= M(x_k) \\
y_k &= M(n) \\
y_{k-1} &= M(x_{k-1}) = M(n/2) \\
y_k &= 2y_{k-1} + 2^k - 1 \\
y_{k+1} - 2y_k &= 2^{k+1} - 1 \\
(E - 2)y &= 2^{k+1} - 1 \\
(E - 2)^2(E - 1)y &= 0 \\
y_k &= A2^k + Bk2^k + C \\
M(n) &= An + Bnlog(n) + C
\end{aligned}$$

5.2 Extended-euclides

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

using namespace std;



//retorna un arreglo con 3 elementos x,y,gcd(a,b) tq ax+by = gcd(a,b)

int* ExtendedEuclides(int a, int b){

   int r0,r1,r2;

   int s0,s1,s2;
```

```
    int t0,t1,t2;

    int q;

    r0 = a; r1 = b;

    s0 = 1; s1 =0;

    t0 = 0; t1 = 1;

    do{

        q = r0/r1;

        r2 = r0-(q*r1);

        s2 = s0-(q*s1);

        t2 = t0-(q*t1);



r0 = r1; s0 = s1; t0 = t1;

        r1 = r2; s1 = s2; t1 = t2;

    }while(r2!=0);

    int  rta[3];

    rta[0] = r0;

    rta[1] = s0;

    rta[2] = t0;

    return &rta[0];

}



int main(){

    int a, b;

    int * rta;

    while(scanf("%d %d", &a, &b)!=EOF){

        rta = ExtendedEuclides(a,b);

        printf("%d %d %d\n",rta[0],rta[1],rta[2]);

    }

}
```

5.3 matrix-operations

```
#include <bits/stdc++.h>
using namespace std;
#define N 2
#define M 1000000009
//estructura de una matriz.
struct matrix {
        long long m[N][N];
        matrix(){ memset(m,0,sizeof(m));}
        matrix operator *(matrix b){
                matrix c = matrix();
                for (int i = 0; i < N; ++i)
                        for (int k = 0; k < N; ++k)
                            for (int j = 0; j < N; ++j)
                                c.m[i][j] = (((c.m[i][j]%M) +((m[i][k]%M) *
                                    (b.m[k][j]%M))) % M);
                return c;
        }

};
// funcion para la matriz identidad
matrix unit(){
        matrix c = matrix();
        for(int i = 0;i<N; i++)
                c.m[i][i]=1;
        return c;
}
// fast matrix MOD pow
matrix modPow(matrix m,int n){
  if ( n == 0 )
    return unit(); // the unit matrix - that is 1 for principal diagonal
         , otherwise 0
  matrix half = modPow(m,n/2);
  matrix out = half * half;
  if ( n % 2 )
    out = out * m;
  return out;
}
int main(){
        matrix fib=matrix();
        matrix bs = matrix();
        fib.m[0][1]=1;
        fib.m[1][0]=1;
        fib.m[1][1]=1;
        bs.m[0][0]=1;
        bs.m[0][1]=1;
        fib = modPow(fib,48);
        fib = bs*fib;
        printf("%lld\n",fib.m[0][1]);
        return 0;
}
```

6 Strings

6.1 LongestCommonSubsequence

```
package Algoritmos;

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class LongestCommonSubsequence {

        static int L[][];
        static char X[];
        static char Y[];
        //Halla la longitd de la subsecuencia comun ms larga
        static int max(int a, int b) {
                return Math.max(a, b);
        }
        static int lcs1(String s1, String s2) {
                X = s1.toCharArray();
                Y = s2.toCharArray();
                int m = X.length, n = Y.length;
                L = new int[m+1][n+1];

                /*
                 * Se llena la matriz L. L[i][j] =
                     lcs(X[0...i-1],Y[0...j-1])
                 */
                for(int i=0; i<=m; i++) {
                        for(int j =0; j<=n; j++) {
                                if(j==0 || i==0) {
                                        L[i][j] = 0;
                                }
                                else if (X[i-1]==Y[j-1]) {
                                        L[i][j] = L[i-1][j-1]+1;
                                }
                                else {
                                        L[i][j] = max(L[i-1][j],L[i][j-1]);
                                }
                        }
                }
                return L[m][n];
        }
        //Reconstruye la subsecuencia comun ms larga
        static String lcs2(String s1, String s2) {
                int index = lcs1(s1,s2);
                System.out.println(index);
                char [] rta = new char[index];
                int i =X.length, j = Y.length;
                while(i>0 && j>0 && index>0) {
                        if(X[i-1]==Y[j-1]) {

                                rta[index-1] = X[i-1];
                                i--; j--; index--;
                        }
                        else if(L[i-1][j]>L[i][j-1]) {
                                i--;
                        }
                        else {
                                j--;
                        }
                }
                return new String(rta);
        }
        public static void main(String[] args) throws Exception {
                BufferedReader bf = new BufferedReader(new
                    InputStreamReader(System.in));
                String s1 = bf.readLine();
                String s2 = bf.readLine();
                String lcs = lcs2(s1,s2);
                System.out.println(lcs);
        }
}
```

6.2 pi-function

```
#include <bits/stdc++.h>
using namespace std;
//funcion para calcular el prefix function de un string
//prefix("abcabcabc")={0,0,0,1,2,3,1,2,3}
vector<int> pi_function(string s){
        int n = (int)s.length();
        vector<int> pi(n);
        for(int i =1; i<n; i++){
                int j = pi[i-1];
                while(j>0&&s[i]!=s[j])
                        j=pi[j-1];
                if(s[i]==s[j])
                        j++;
                pi[i]=j;
        }
        return pi;
}
int main(){
        string p;
        string t;
        cin>>t>>p;
        int len = (int)p.length();
        // hacer kmp es igual a halla la prefix function de p+"#"+s y
            calcular y un match es equivalente a pi[i]=len(p)
        string kmp = p+"#"+t;
        vector<int> pi = pi_function(kmp);
        for(int i = 0; i<(int)pi.size();i++)
                printf("%d ",pi[i]);
        int ans = 0;
        for(int i = len+1; i<(int)pi.size(); i++)
                if(pi[i]==len)
                        ans++;
        printf("%d\n",ans);
        return 0;
}
```

6.3 z-function

```
#include <bits/stdc++.h>
using namespace std;
//funcion para calcular la z function de un string
vector<int> z_function(string s){
        int n = (int) s.length();
        vector<int> z(n);
        for(int i=1,l=0,r=0; i<n; i++){
                if(i<=r)
                        z[i]=min(r-i+1,z[i-l]);
                while(i+z[i]<n && s[z[i]]==s[i+z[i]])
                        ++z[i];
                if(i+z[i]-1>r)
                        l=i,r=i+z[i]-1;
        }
        return z;
}
int main(){
        string s("abcabcabc");
        vector<int> z = z_function(s);
        //z == {0,0,0,6,0,0,3,0,0}
        for(int i = 0; i<(int)z.size(); i++)
                printf("%d ",z[i]);
        return 0;
}
```

7 utilities

7.1 linux-utilities

```
## correr con un archivo y copiar resulatdo en otro
./file < fileIn > fileOut
## comparar dos archivos
```

```
diff file1 file2
```