Contents

1 Basic

1.1 .vimrc

1.2 Increase Stack Size

```
//stack resize
asm( "mov %0,%%esp\n" ::"g"(mem+10000000) );
//change esp to rsp if 64-bit system

//stack resize (linux)
#include <sys/resource.h>
void increase_stack_size() {
  const rlim_t ks = 64*1024*1024;
  struct rlimit rl;
  int res=getrlimit(RLIMIT_STACK, &rl);
  if(res==0){
    if(rl.rlim_cur<ks){
      rl.rlim_cur=ks;
      res=setrlimit(RLIMIT_STACK, &rl);
    }
  }
}
```

2 Graph

2.1 HLD

```
struct segment_tree{
    #define MAXN 100100
    #define right(x) x << 1 | 1
    #define left(x) x << 1
    int* arr;
    LL sum[4*MAXN];
    const int inf = 1e9;

    void pull(int ind) {
        sum[ind] = sum[right(ind)]+sum[left(ind
            )];
    };
    /// root => 1
    void build(int ind, int l, int r) {

        if( r - l == 1) {
            sum[ind] = 0;
            return;
        }
        int mid = (l+r)>>1;
        build( left(ind), l, mid );
        build( right(ind), mid, r );
        pull(ind);
    }
    LL query_sum(int ind, int L, int R, int ql,
         int qr) {
        if( L >= qr || R <= ql ) return 0;
        if( R <= qr && L >= ql ) {
            return sum[ind];
        }
        int mid = (L+R)>>1;
        return query_sum(left(ind), L, mid, ql,
             qr) + query_sum(right(ind), mid, R
            , ql, qr);
    }
    void modify(int ind, int L, int R, int ql,
        int qr, int x) {
        if( L >= qr || R <= ql ) return;
        if( R <= qr && L >= ql ) {
            sum[ind] = x;
            return;
        }
        int mid = (L+R)>>1;
        modify(left(ind), L, mid, ql, qr, x);
        modify(right(ind), mid, R, ql, qr, x);
        pull(ind);
    }
};

struct Tree{
    segment_tree seg;
    #define MAXN 100010
    #define maxm (maxn<<1)
    int n;
    struct edge { int u, v; };
    vector<edge> e;
    void addedge(int x, int y) {
        G[x].pb( SZ(e) );
        G[y].pb( SZ(e) );
        e.pb( edge{x, y}  );
    }
    int siz[MAXN],max_son[MAXN],pa[MAXN],dep[
        MAXN];
    /*size of  subtreeindex of max_son, parent
         indexdepth*/
    int link_top[MAXN],link[MAXN],Time;
    /*chain  topindex in  segtreetime stamp*/
    std::vector<int >G[MAXN];

    void init(int N) {
        n = N;
        e.clear();
        for(int i = 1; i <= n; i++) G[i].clear
            ();
    }
    void find_max_son(int x){
        siz[x]=1;
        max_son[x]=-1;
        for(int e_ind : G[x]) {
            int v = e[e_ind].u == x ? e[e_ind].
                v : e[e_ind].u ;
            if( v == pa[x] )continue;
            pa[v] = x; dep[v] = dep[x] + 1;
            find_max_son(v);
```

```
            if(max_son[x] == -1 || siz[v] > siz
                [max_son[x]])
                max_son[x] = v;
            siz[x] += siz[v];
        }
    }
    void build_link(int x,int top){
        link[x] = ++Time;/*    x*/
        link_top[x] = top;
        if(max_son[x] == -1)return;
        build_link( max_son[x], top);/*    */
        for(int e_ind : G[x]) {
            int v = e[e_ind].u == x ? e[e_ind].
                v : e[e_ind].u ;
            if( v == max_son[x] || v == pa[x] )
                continue;
            build_link(v, v);
        }
    }
    inline int lca(int a,int b){
        /* ,       LCA*/
        int ta=link_top[a],tb=link_top[b];
        while(ta != tb){
            if(dep[ta]<dep[tb]){
                std::swap(ta,tb);
                std::swap(a,b);
            }
            //interval [ link[ta], link[a] ]
            a = pa[ta];
            ta = link_top[a];
        }
        return dep[a] < dep[b] ? a:b;
    }
    int query(int a,int b){
        int ret = 0;
        int ta=link_top[a],tb=link_top[b];
        while(ta != tb){
            if(dep[ta]<dep[tb]){
                std::swap(ta,tb);
                std::swap(a,b);
            }
            //interval [ link[ta],link[a] ]
            a = pa[ta];
            ta = link_top[a];
        }
        if( a == b ) return ret;
        else {
            if(dep[a]>dep[b])
                swap(a,b);
            //interval [ link[a],link[b] ]
            // if operate on edges ==> [ link[
                max_son[ta] ], link[b] ]
        }
    }
    /// Heavy Light Decomposition
    void HLD() {
        // root is indexed 1 here !
        find_max_son(1);
        build_link(1, 1);
    }
     void modify(int a, int b, int x) {
        // modify the path from a -> b to x
        //( which is [ link[a] .. link[b] ] on
            the segment tree)
        seg.modify(1, 1, n+1, link[a], link[b
            ]+1, x);
        // this segment tree uses [ 1 ..n+1 )
    }
}tree;
```

2.2 bipartite graph matching

2.3 Hungarian

```
// edge and node index starting from 0
// dfs  version below
/* to do
#define __maxNodes
num_left = ?
*/
struct Edge {
    int from;
    int to;
    int weight;
    Edge(int f, int t, int w):from(f), to(t),
        weight(w) {}
};
vector<int> G[__maxNodes]; /* G[i]    i      */
vector<Edge> edges;
int num_nodes;
int num_left;
int num_right;
int num_edges;
int matching[__maxNodes]; /* matching result */
int check[__maxNodes];

bool dfs(int u) {
    for (auto i = G[u].begin(); i != G[u].end()
        ; ++i) { //   u
        int v = edges[*i].to;
        if (!check[v]) {     //
            check[v] = true; //
            if (matching[v] == -1 || dfs(
                matching[v])) {
                //    ,     ,  ,
                matching[v] = u;
                matching[u] = v;
                return true;
            }
        }
    }
    return false; //    ,
}
int hungarian() {
    int ans = 0;
    memset(matching, -1, sizeof(matching));
    for (int u=0; u < num_left; ++u) {
        if (matching[u] == -1) {
            memset(check, 0, sizeof(check));
            if (dfs(u)) ++ans;
        }
    }
    return ans;
}
```

2.4 KM

```
//    ~  km
//http://acm.csie.org/ntujudge/contest_view.php
    ?id=836&contest_id=449
#include <bits/stdc++.h>
using namespace std;

struct bipartite {
    #define maxn 602
    #define INF 0xffffff
    int sx[maxn], sy[maxn], mat[maxn][maxn];
    int x[maxn], y[maxn], link[maxn];
    int N, M, slack;

    int DFS(int t) {
        int tmp;
        sx[t] = 1;
        for (int i = 0; i < M; i++) {
            if (!sy[i]) {
                tmp = x[t] + y[i] - mat[t][i];
                if (tmp == 0) {
                    sy[i] = 1;
                    if (link[i] == -1 || DFS(
                        link[i])) {
                        link[i] = t;
                        return 1;
                    }
                }
                else if (tmp < slack) slack =
                    tmp;
            }
        }
        return 0;
    }
    int KM() {
        for (int i = 0; i < N; i++) {
            x[i] = 0;
            for (int j = 0; j < M; j++) {
                if (mat[i][j] > x[i]) x[i] =
                    mat[i][j];
            }
        }
        for (int j = 0; j < M; j++) { y[j] = 0;
         }
        memset(link, -1, sizeof(link));
        for (int i = 0; i < N; i++) {
            while (1) {
                memset(sx, 0, sizeof(sx));
                memset(sy, 0, sizeof(sy));
                slack = INF;
                if (DFS(i)) break;
                for (int j = 0; j < N; j++) {
                    if (sx[j]) x[j] -= slack;
                }
                for (int j = 0; j < M; j++) {
                    if (sy[j]) y[j] += slack;
                }
            }
        }

        int ans = 0;
        int cnt = 0;
        int t;
        for (int i = 0; i < M; i++)
        {
            t = link[i];
            if (t >= 0 && mat[t][i] != -INF)
            {
                cnt ++;
                ans += mat[t][i];
            }
        }
        //    :
        return -ans;
    }
    void init(int n,int m) {
        N = n, M = m;
        for (int i = 0; i < N; i++)
            for (int j = 0; j < M; j++)
                mat[i][j] = -INF;
    }
    void input() {
        for(int i = 0; i < N; i++)
            for(int j =0;j<M;j++) {
                // fill in mat[i][j]
                // stands for the weighting ,
                    but negative sign !
                // if    :
            }

    }
}km;

int main(){
    int n,E;
    while (scanf("%d", &n) != EOF)
    {
        km.init(n, n);
        km.input();
        cout<< km.KM() <<endl;
    }
    return 0;
```

```
}
```

3 Data Structure

3.1 Disjoint Set

4 Math

4.1 Prime Table

```
#include <bits/stdc++.h>
using namespace std;
struct Prime_table {

    int prime[1000000]={2,3,5,7};
    int sz=4;
    // biggest prime < ub
    int ub=(1<<20);

    int check(int num){
        int k = 0;
        for(k = 0; k < sz && prime[k]*prime[k]
            <= num; k++){
            if( num % prime[k]==0)  return 0;
        }
        return 1;
    }
    void buildprime(){
        int currentPrime=7;
        int j=4;
        for(sz=4,j=4; currentPrime<ub; sz++, j
            =6-j){
             currentPrime=currentPrime+j;
             if (check(currentPrime)) {
                prime[sz] = currentPrime;
             }
             else{
                sz--;
             }
        }
     }
}ptable;
```

4.2 Miller Rabin(prime test)

```
#include <cstdio>
#include <vector>
#include <map>
#include <algorithm>
using namespace std;


long long mul(unsigned long long a, unsigned
    long long b, unsigned long long mod) {
    long long ret = 0;
    for (a %= mod, b %= mod; b != 0; b >>= 1, a
         <<= 1, a = a >= mod ? a - mod : a) {
        if (b&1) {
            ret += a;
            if (ret >= mod) ret -= mod;
        }
    }
    return ret;
}

long long mpow2(long long x, long long y, long
    long mod) {
    long long ret = 1;
    while (y) {
        if (y&1)
            ret = mul(ret, x, mod);
        y >>= 1, x = mul(x, x, mod);
    }
    return ret % mod;
}
int isPrime(long long p, int it) { //
    implements by miller-babin
    if (p < 2)  return 0;
    if (p == 2) return 1;
    if (!(p&1)) return 0;
    long long q = p-1, a, t;
    int k = 0, b = 0;
    while (!(q&1))  q >>= 1, k++;

    while(it--) {
        a = rand()%(p-4) + 2;
        t = mpow2(a, q, p);
        b = (t == 1) || (t == p-1);
        for (int i = 1; i < k && !b; i++) {
            t = mul(t, t, p);
            if (t == p-1)
                b = 1;
        }
        if (b == 0)
            return 0;
    }

    return 1;
}


int main() {

    int testcase;
    scanf("%d", &testcase);
    while (testcase--) {
        long long n;
        scanf("%lld", &n);
        puts(isPrime(n, 1000)?"YES":"NO");
    }
    return 0;
}
```

4.3 Extended Euclidean Algorithm

```
/** normal gcd function using recursion **/
int gcd(int a, int b){
    if(b == 0) return a;
    return gcd(b, a%b);
}
// Find solution of ax + by = gcd(a, b)
// ps : x, y may be negative
int extgcd(int a, int b, int& x, int& y){
    int d = a;
    if(b != 0) {
        d = extgcd(b, a%b, y, x);
        y -= (a/b) * x;
    }else {
        x = 1, y = 0;
    }
    return d;
}
```

4.4 Matrix Fast Power

```
typedef vector<int> vec;
typedef vector<vec> mat;
typedef long long LL;
const int mod = 10000;

mat mul(mat &A, mat &B){
    mat C(A.size(), vec(B[0].size()));
    // initialize size of matrix C => A.size()
         vector with B[0].size()
    for(int i = 0; i < A.size(); i++) {
        for(int k = 0; k < B.size(); k++) {
            for(int j = 0; j < B[0].size(); j
                ++) {
                C[i][j] = (C[i][j] +A[i][k]*B[k
                    ][j]) % mod;
            }
        }
    }
    return C;
}
mat pow(mat A, LL n){
    mat B(A.size(), vec(A.size()));
    for(int i = 0; i < A.size(); i++)
        B[i][i] = 1;
    while(n > 0){
        if(n%2) B = mul(B, A);
        A = mul(A, A);
        n >>= 1;
    }
    return B;
}
int main(){
    LL n;cin>>n;
    mat A(2, vec(2));
    A[0][0] = A[0][1] = A[1][0] = 1, A[1][1] = 0;
    A = pow(A, n);
    //fibonacci number n term
    printf("%d", A[1][0]);
}
```

4.5 Gauss Elimination

```
// solving linear equations with gauss elimination
#include <iostream>
#include <cmath>
#include <vector>

using namespace std;

void print(vector< vector<double> > A) {
    int n = A.size();
    for (int i=0; i<n; i++) {
        for (int j=0; j<n+1; j++) {
            cout << A[i][j] << "\t";
            if (j == n-1) {
                cout << "| ";
            }
        }
        cout << "\n";
    }
    cout << endl;
}

vector<double> gauss(vector< vector<double> > A) {
    int n = A.size();

    for (int i=0; i<n; i++) {
        // Search for maximum in this column
        double maxEl = abs(A[i][i]);
        int maxRow = i;
        for (int k=i+1; k<n; k++) {
            if (abs(A[k][i]) > maxEl) {
                maxEl = abs(A[k][i]);
                maxRow = k;
            }
        }

        // Swap maximum row with current row (column by column)
        for (int k=i; k<n+1;k++) {
            double tmp = A[maxRow][k];
            A[maxRow][k] = A[i][k];
            A[i][k] = tmp;
        }

        // Make all rows below this one 0 in current column
        for (int k=i+1; k<n; k++) {
            double c = -A[k][i]/A[i][i];
            for (int j=i; j<n+1; j++) {
                if (i==j) {
                    A[k][j] = 0;
                } else {
                    A[k][j] += c * A[i][j];
                }
            }
        }
    }

    // Solve equation Ax=b for an upper triangular matrix A
    vector<double> x(n);
    for (int i=n-1; i>=0; i--) {
        x[i] = A[i][n]/A[i][i];
        for (int k=i-1;k>=0; k--) {
            A[k][n] -= A[k][i] * x[i];
        }
    }
    return x;
}

int main() {
    int n;
    cin >> n;

    vector<double> line(n+1,0);
    vector< vector<double> > A(n,line);

    // Read input data
    for (int i=0; i<n; i++) {
        for (int j=0; j<n; j++) {
            cin >> A[i][j];
        }
    }

    for (int i=0; i<n; i++) {
        cin >> A[i][n];
    }

    // Print input
    print(A);

    // Calculate solution
    vector<double> x(n);
    x = gauss(A);

    // Print result
    cout << "Result:\t";
    for (int i=0; i<n; i++) {
        cout << x[i] << " ";
    }
    cout << endl;
}
```