

Algoritmos Informáticos Básicos C ++

Frank Rodríguez Siret - Septiembre 2014

Índice de contenido

1. Graph.....	2
1.1. Articulations Points.....	2
1.2. Bridges.....	3
1.3. Strong Connected Components.....	4
1.4. Eulerian Circuit or Path.....	5
1.5. Floyd Warshall.....	6
1.6. DFS.....	6
1.7. Lowest Common Ancestor.....	7
1.8. Prim.....	8
1.9. Kruskal With Disjoin Set.....	9
1.10. Dijkstra.....	10
1.11. BFS.....	11
2. Data Structure.....	12
2.1. Binary Indexed Tree.....	12
2.2. Range Min - Max Quering.....	13
2.3. Segment Tree.....	14
2.4. Trie.....	15
3. String Matching.....	16
3.1. Knuth Morris Pratt.....	16
3.2. Suffix Array.....	17
4. Dynamic.....	18
4.1. Edit Distance.....	18
4.2. Longest Common Subsequence.....	18
4.3. Longest Increasing or Decreasing Subsequence.....	19
4.4. Pack with encore.....	19
4.5. Pack without encore.....	20
4.6. Counting Change.....	20
4.7. Accumulative table.....	21
4.8. Catcher.....	21
5. Geometry.....	22
5.1. Closest Pair Of Points (Convex Hull).....	22
5.2. Closest Pair Of Points (Sweep Line).....	23
5.3. Convex Hull (Graham Scan).....	24
5.4. Union Area (Segment Tree + Sweep Line).....	25
6. Math.....	26
6.1. Gaus Common Divisor.....	26
6.2. Big Mod (b^e) % m.....	26
6.3. Counting Combinations C (n, k).....	27
6.4. Exponentiation n^k	28
6.5. Fórmulas.....	29

1. Graph

1.1. Articulations Points

```
#include <cstdio>
#include <vector>
#include <stack>
#define RANG 1000010
using namespace std;

vector <int> A[RANG];
stack <int> Q;
int v, a, nod, newn, t, DT[RANG], LOW[RANG];
bool MK[RANG];

void AP (int nod) {
    DT[nod] = LOW[nod] = ++t;
    for (vector <int>::iterator newn = A[nod].begin(); newn != A[nod].end();
newn++) {
        if (!LOW[*newn]) {
            AP (*newn);
            LOW[nod] = min (LOW[nod], LOW[*newn]);
            if (!MK[nod] && (DT[nod] != 1 && DT[nod] <= LOW[*newn]) ||
(DT[nod] == 1 && DT[*newn] > 2)) {
                MK[nod] = true;
                Q.push (nod);
            }
        }
        else
            LOW[nod] = min (LOW[nod], DT[*newn]);
    }
}

main() {
    freopen ("AP.in", "r", stdin);
    freopen ("AP.ou", "w", stdout);
    scanf ("%d %d", &v, &a);
    for (int i = 0; i < a; i++) {
        scanf ("%d %d", &nod, &newn);
        A[nod].push_back (newn);
        A[newn].push_back (nod);
    }
    AP (1);
    while (!Q.empty()) {
        printf ("%d\n", Q.top());
        Q.pop();
    }
}
```

1.2. Bridges

```
#include <cstdio>
#include <vector>
#include <stack>
#define RANG 1000010
using namespace std;

struct tri {
    int nod, newn;
    bool marc;
    int nextn (int *a) {
        if (nod == *a)
            return newn;
        return nod;
    }
} A[RANG];
struct two {
    int nod, newn;
};
stack <two> Q;
vector <int> ID[RANG];
int v, a, nod, newn, t, DT[RANG], LOW[RANG];
void Bridges (int nod) {
    DT[nod] = LOW[nod] = ++t;
    for (vector <int>::iterator id = ID[nod].begin(); id != ID[nod].end(); id++) {
        int newn = A[*id].nextn (&nod);
        if (!LOW[newn]) {
            A[*id].marc = true;
            Bridges (newn);
            LOW[nod] <?= LOW[newn];
            if (DT[nod] < LOW[newn])
                Q.push ((two) {nod, newn});
        }
        else
            if (!A[*id].marc)
                LOW[nod] <?= DT[newn];
    }
}
main() {
    freopen ("Bridges.in", "r", stdin);
    freopen ("Bridges.ou", "w", stdout);
    scanf ("%d %d", &v, &a);
    for (int i = 0; i < a; i++) {
        scanf ("%d %d", &nod, &newn);
        A[i] = (tri) {nod, newn, false};
        ID[nod].push_back (i);
        ID[newn].push_back (i);
    }
    Bridges (1);
    while (!Q.empty()) {
        printf ("%d %d\n", Q.top().nod, Q.top().newn);
        Q.pop();
    }
}
```

1.3. Strong Connected Components

```
#include <cstdio>
#include <vector>
#include <stack>
#define RANG 1000010
using namespace std;

vector <int> A[RANG];
stack <int> Q;
int v, a, t, nod, newn, LOW[RANG], DT[RANG];
bool MK[RANG];

void SCC (int nod) {
    DT[nod] = LOW[nod] = ++t;
    Q.push (nod);
    for (vector <int>::iterator newn = A[nod].begin(); newn != A[nod].end();
newn++) {
        if (!LOW[*newn]) {
            SCC (*newn);
            LOW[nod] = min (LOW[nod], LOW[*newn]);
        }
        else
            if (!MK[*newn])
                LOW[nod] = min (LOW[nod], DT[*newn]);
    }
    if (LOW[nod] == DT[nod]) {
        while (Q.top() != nod) {
            printf ("%d ", Q.top());
            MK[Q.top()] = true;
            Q.pop();
        }
        printf ("%d\n", Q.top());
        MK[nod] = true;
        Q.pop();
    }
}

main() {
    freopen ("SCC.in", "r", stdin);
    freopen ("SCC.ou", "w", stdout);
    scanf ("%d %d", &v, &a);
    for (int i = 1; i <= a; i++) {
        scanf ("%d %d", &nod, &newn);
        A[nod].push_back (newn);
    }
    for (int i = 1; i <= v; i++)
        if (!LOW[i])
            SCC (i);
}
```

1.4. Eulerian Circuit Or Path

```
#include <cstdio>
#include <vector>
#include <queue>
#define RANG 1000010
using namespace std;

struct tri {
    int nod, newn;
    bool marc;
    int nextn (int *x) {
        if (*x == nod)
            return newn;
        return nod;
    }
} A[RANG];
vector<int> ID[RANG];
queue<int> Q;
int v, a, nod, newn, impar, start = 1, G[RANG];
void Euler (int nod) {
    for (vector<int>::iterator id = ID[nod].begin(); id != ID[nod].end(); id++) {
        if (!A[*id].marc) {
            A[*id].marc = true;
            Euler (A[*id].nextn (&nod));
        }
    }
    Q.push (nod);
}

main() {
    freopen ("Euler.in", "r", stdin);
    freopen ("Euler.ou", "w", stdout);
    scanf ("%d %d", &v, &a);
    for (int i = 1; i <= a; i++) {
        scanf ("%d %d", &nod, &newn);
        ID[nod].push_back (i);
        ID[newn].push_back (i);
        A[i] = (tri) {nod, newn, false};
        G[nod]++;
        G[newn]++;
    }
    for (int i = 1; i <= v; i++)
        if (G[i] % 2) {
            impar++;
            start = i;
            if (impar > 2) {
                printf ("The Graph is not Eulerian\n");
                return 0;
            }
        }
    Euler (start);
    if (!impar)
        printf ("Eulerian Circuit\n");
    else
        printf ("Eulerian Path\n");
    for (; !Q.empty(); Q.pop())
        printf ("%d ", Q.front());
}
```

1.5. Floyd Warshall

```
#include <cstdio>
#include <algorithm>
#define RANG 310
using namespace std;

int v, a, q, nod, newn, cost, A[RANG][RANG];
main () {
    freopen ("FW.in", "r", stdin);
    freopen ("FW.ou", "w", stdout);
    memset (A, 63, sizeof (A));
    scanf ("%d %d", &v, &a);
    for (int i = 0; i < a; i++) {
        scanf ("%d %d %d", &nod, &newn, &cost);
        A[nod][newn] <?= cost;
    }
    for (int i = 1; i <= v; i++)
        for (int j = 1; j <= v; j++)
            for (int k = 1; k <= v; k++)
                A[i][j] <?= A[i][k] + A[k][j];
    scanf ("%d", &q);
    while (q--) {
        scanf ("%d %d", &nod, &newn);
        printf ("%d\n", A[nod][newn]);
    }
}
```

1.6. DFS

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

int G[100000][2], dist[100000], N, C;
vector <int> G1[100000];

void DFS ( int start )
{
    if ( G1[start].empty() )
        return;
    int tam = G1[start].size();
    for ( int i = 0; i < tam; i ++ )
    {
        int ady = G1[start][i];
        dist[ady] = dist[start] + 1;
        dfs ( ady );
    }
}

int main() {
    freopen ("DFS.in", "r", stdin);
    freopen ("DFS.out", "w", stdout);
    int a, b, c;
    scanf ( "%d%d", &N, &C );
    for ( int i = 0; i < C; i ++ ) {
        scanf ("%d%d%d", &a, &b, &c);
        G1[a].push_back(b);
        G1[a].push_back(c);
    }
    dist[1] = 1;
    DFS ( 1 );
    for ( int i = 1; i <= N; i ++ )
        printf ("%d\n", dist[i]);
}
```

1.7. Lowest Common Ancestor

```
#include <cstdio>
#include <vector>
#include <cmath>
#define RANG 100010
using namespace std;

vector <int> A[RANG];
int v, a, q, nod, newn, maxlog, LV[RANG], T[RANG][20];

void DFS (int nod, int lv) {
    LV[nod] = lv;
    maxlog = (int) log2 (lv);
    for (int i = 1; i <= maxlog; i++)
        T[nod][i] = T[T[nod][i - 1]][i - 1];
    for (vector <int>::iterator newn = A[nod].begin(); newn != A[nod].end();
newn++)
        DFS (*newn, lv + 1);
}

int search (int *nod, int *newn) {
    if (LV[*nod] < LV[*newn])
        swap (nod, newn);

    maxlog = (int) log2 (LV[*nod]);
    for (int i = maxlog; i >= 0; i--)
        if (LV[*nod] - (1 << i) >= LV[*newn])
            *nod = T[*nod][i];
    if (*nod == *newn)
        return *nod;
    maxlog = (int) log2 (LV[*nod]);
    for (int i = maxlog; i >= 0; i--)
        if (T[*nod][i] != T[*newn][i] && T[*nod][i]) {
            *nod = T[*nod][i];
            *newn = T[*newn][i];
        }
    return T[*nod][0];
}

main () {
    freopen ("LCA.in", "r", stdin);
    freopen ("LCA.ou", "w", stdout);
    scanf ("%d %d", &v, &a);
    for (int i = 0; i < a; i++) {
        scanf ("%d %d", &nod, &newn);
        A[nod].push_back (newn);
        T[newn][0] = nod;
    }
    DFS (1, 1);
    scanf ("%d", &q);
    while (q--) {
        scanf ("%d %d", &nod, &newn);
        printf ("%d\n", search (&nod, &newn));
    }
}
```

1.8. Prim

```
#include <cstdio>
#include <vector>
#include <queue>
#define RANG 100
using namespace std;

typedef pair <int, int> two;
vector <two> A[RANG];
priority_queue <two, vector <two>, greater <two> > Q;
bool M[RANG];
int n, a, newn, cost, nod, sol;

main() {
    freopen ("prim.in", "r", stdin);
    freopen ("prim.out", "w", stdout);
    scanf ("%d %d", &n, &a);
    for (int i = 1; i <= a; i++) {
        scanf ("%d %d %d", &nod, &newn, &cost);
        A[nod].push_back (two (newn, cost));
        A[newn].push_back (two (nod, cost));
    }
    Q.push(two (0, 1));
    while (!Q.empty()) {
        nod = Q.top().second;
        cost = Q.top().first;
        Q.pop();
        if (!M[nod]) {
            M[nod] = true;
            sol += cost;
            for (vector<two>::iterator i = A[nod].begin(); i !=
A[nod].end(); i++)
                if (!M[i->first])
                    Q.push(two (i->second, newn));
        }
    }
    printf ("%d\n", sol);
}
```


1.9. Kruskal With Disjoin Set

```
#include <cstdio>
#include <algorithm>
#define RANG 100
using namespace std;

int v, a, nod, newn, cost, setnod, setnewn, sol, SET[RANG], R[RANG];
struct tri {
    int nod, newn, cost;
    bool operator < (const tri &p) const {
        return cost < p.cost;
    }
} A[RANG];
void make_set (int i) {
    SET[i] = i;
    R[i] = 1;
}
int find_set (int nod) {
    if (SET[nod] != nod)
        SET[nod] = find_set (SET[nod]);
    return SET[nod];
}
void join_set (int nod, int newn) {
    if (R[nod] > R[newn]) {
        SET[newn] = nod;
        R[nod]++;
    }
    else {
        SET[nod] = newn;
        R[newn]++;
    }
}
main() {
    freopen ("kruskal.in", "r", stdin);
    freopen ("kruskal.out", "w", stdout);
    scanf ("%d %d", &v, &a);
    for (int i = 0; i < a; i++) {
        scanf ("%d %d %d", &nod, &newn, &cost);
        A[i] = (tri) {nod, newn, cost};
    }
    sort (A, A + a);
    for (int i = 1; i <= v; i++)
        make_set (i);
    for (int i = 0; i < a; i++) {
        setnod = find_set (A[i].nod);
        setnewn = find_set (A[i].newn);
        if (setnod != setnewn) {
            sol += A[i].cost;
            join_set (setnod, setnewn);
        }
    }
    printf ("%d\n", sol);
}
```

1.10. Dijkstra

```
#include <cstdio>
#define maxint 1 << 30
#include <queue>
#include <vector>
#include <algorithm>
using namespace std;

int N, dist[1000], may;
typedef pair <int, int> par;
vector <par> G[1000];
priority_queue <par, vector<par>, greater<par> > Qp;

void dijkstra ( int start ) {
    int costo, nodo, newc, ady;

    fill ( dist + 1, dist + N + 1, maxint );
    Qp.push(par ( 0, start ) );

    while ( !Qp.empty() ) {
        nodo = Qp.top().second, costo = Qp.top().first;
        Qp.pop();

        for ( int i = 0; i < G[nodo].size(); i ++ ) {
            ady = G[nodo][i].first, newc = G[nodo][i].second + costo;

            if ( dist[ady] > newc ) {
                dist[ady] = newc;
                Qp.push( par ( newc, ady ) );
            }
        }
    }
}

int main() {
    freopen ( "dijkstra.in", "r", stdin );
    freopen ( "dijkstra.out", "w", stdout );
    scanf ( "%d", &N );

    int a, b, c;
    for ( int i = 1; i <= N; i ++ ) {
        scanf ( "%d%d%d", &a, &b, &c );

        G[a].push_back( par ( b, c ) );
        G[b].push_back( par ( a, c ) );
    }

    dijkstra( 1 );

    printf ( "%d", dist[N] );
}
```

1.11. BFS

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

bool G[100][100], mark[901];
int N, C, parent[100];
queue <int> Q;
void printcm ( int ini, int fin )
{
    if ( ini == fin || parent[fin] == 0 )
        printf ("%d ", ini );
    else
    {
        printcm ( ini, parent[fin] );
        printf ("%d ", fin );
    }
}

void BFS ( int start )
{
    int nodo, ady;
    Q.push(start);
    mark[start] = true;
    while ( !Q.empty() )
    {
        nodo = Q.front();
        Q.pop();
        for ( int i = 1; i <= N; i ++ )
        {
            if ( !G[nodo][i] )
                continue;
            ady = i;
            if ( mark[ady] )
                continue;
            mark[ady] = true;
            parent[ady] = nodo;
            Q.push(ady);
        }
    }
}

int main() {
    freopen ("BFS.in", "r", stdin);
    freopen ("BFS.out", "w", stdout);
    int a, b, ini, fin;
    scanf ("%d%d", &N, &C);
    for ( int i = 0; i < C; i ++ ) {
        scanf ("%d%d", &a, &b);
        G[a][b] = true;
        G[b][a] = true;
    }
    BFS( 1 );
    printf ("%d%d", ini, fin);
    printcm ( ini, fin );
}
```

2. Data Structure

2.1. Binary Indexed Tree

```
#include <stdio>
#define RANG 1000010
using namespace std;

struct bit {
    int l, T[RANG];
    void add (int *x, int *n) {
        for (int i = *x; i <= l; i += i & -i)
            T[i] += *n;
    }
    int sum (int *x) {
        int sum = 0;
        for (int i = *x; i; i -= i & -i)
            sum += T[i];
        return sum;
    }
    void update (int *x, int *n) {
        int lastx = *x - 1;
        int sumx = sum (x) - sum (&lastx);
        sumx = *n - sumx;
        add (x, &sumx);
    }
} BIT;
int q, x, n, sol;
char qt;
main() {
    freopen ("BIT.in", "r", stdin);
    freopen ("BIT.ou", "w", stdout);
    scanf ("%d %d\n", &BIT.l, &q);
    while (q--) {
        scanf ("%c ", &qt);
        if (qt == 'a') {
            scanf ("%d %d\n", &x, &n);
            BIT.add (&x, &n);
            continue;
        }
        if (qt == 'u') {
            scanf ("%d %d\n", &x, &n);
            BIT.update (&x, &n);
            continue;
        }
        scanf ("%d %d\n", &x, &n);
        sol = BIT.sum (&n) - BIT.sum (&(--x));
        printf ("%d\n", sol);
    }
}
```

2.2. Range Min - Max Quering

```
#include <cstdio>
#include <algorithm>
#include <cmath>
#define RANG 1000000
using namespace std;
int n, c, p, q, a, b;
struct two {
    int min, max;
} T[RANG][19];
main() {
    freopen ("RMQ.in", "r", stdin);
    freopen ("RMQ.ou", "w", stdout);
    scanf ("%d", &n);
    for (int i = 1; i <= n; i++) {
        scanf ("%d", &T[i][0].min);
        T[i][0].max = T[i][0].min;
    }
    c = (int) log2 (n);
    a = n;
    for (int j = 1; j <= c; j++) {
        p = 1 << j - 1;
        a -= p;
        for (int i = 1; i <= a; i++) {
            T[i][j].min = min (T[i][j - 1].min, T[i + p][j - 1].min);
            T[i][j].max = max (T[i][j - 1].max, T[i + p][j - 1].max);
        }
    }
    scanf ("%d", &q);
    while (q--) {
        scanf ("%d %d", &a, &b);
        c = (int) log2 (b - a);
        b = b - (1 << c) + 1;
        printf ("%d %d\n", min (T[a][c].min, T[b][c].min), max (T[a][c].max,
T[b][c].max));
    }
    c = (int) log2 (n);
    for (int i = 1; i <= n; i++) {
        for (int j = 0; j <= c; j++)
            printf ("%d/%d ", T[i][j].min, T[i][j].max);
        printf ("\n");
    }
}
```

2.3. Segment Tree

```
#include <cstdio>
#include <algorithm>
#define RANG 1000010
#define oo 1 << 30
using namespace std;

int a, b, q, N[RANG];
char qt;
struct st {
    int l, T[RANG];
    int build (int x, int xend, int lv) {
        if (x == xend)
            return T[lv] = N[x];
        int piv = (x + xend) / 2;
        return T[lv] = min (build (x, piv, lv * 2), build (piv + 1, xend, lv
* 2 + 1));
    }
    int update (int x, int xend, int lv) {
        if (x > a || xend < a)
            return T[lv];
        if (x == xend)
            return N[x];
        int piv = (x + xend) / 2;
        return T[lv] = min (update (x, piv, lv * 2), update (piv + 1, xend, lv *
2 + 1));
    }
    int query (int x, int xend, int lv) {
        if (a > xend || b < x)
            return oo;
        if (a <= x && b >= xend)
            return T[lv];
        int piv = (x + xend) / 2;
        return min (query (x, piv, lv * 2), query (piv + 1, xend, lv * 2 +
1));
    }
} ST;
main () {
    freopen ("ST.in", "r", stdin);
    freopen ("ST.ou", "w", stdout);

    scanf ("%d", &ST.l);
    for (int i = 1; i <= ST.l; i++)
        scanf ("%d", &N[i]);
    ST.build (1, ST.l, 1);
    scanf ("%d\n", &q);
    while (q--) {
        scanf ("%c %d %d\n", &qt, &a, &b);
        if (qt == 'q')
            printf ("%d\n", ST.query (1, ST.l, 1));
        else {
            N[a] = b;
            ST.update (1, ST.l, 1);
        }
    }
}
```

2.4. Trie

```
#include <cstdio>
#include <algorithm>
#define RANG 256
using namespace std;

struct trie {
    bool marc;
    trie *next[RANG];
} TRIE, *P;
int l, k, q, lW;
char W[RANG];

main () {
    freopen ("Trie.in", "r", stdin);
    freopen ("Trie.ou", "w", stdout);
    scanf ("%d", &l);
    for (int i = 0; i < l; i++) {
        scanf ("%s", &W);
        P = &TRIE;
        lW = strlen (W);
        for (int j = 0; j < lW; j++) {
            if (P -> next[W[j]] == NULL) {
                P -> next[W[j]] = new
                    trie();
                P = P -> next[W[j]];
            }
            else
                P = P -> next[W[j]];
        }
        P -> marc = true;
    }
    scanf ("%d", &q);
    while (q--) {
        scanf ("%s", &W);
        P = &TRIE;
        lW = strlen (W);
        for (k = 0; k < lW; k++) {
            if (P -> next[W[k]] == NULL)
                break;
            P = P -> next[W[k]];
        }
        if (k == lW && P -> marc)
            printf ("YES\n");
        else
            printf ("NO\n");
    }
}
```

3. String Matching

3.1. Knuth Morris Pratt

```
#include <stdio>
#include <string>
#define RANG 1000010
using namespace std;

int lA, lB, mf, F[RANG];
char A[RANG], B[RANG];

int main() {
    freopen ("KMP.in", "r", stdin);
    freopen ("KMP.ou", "w", stdout);
    scanf ("%s", A + 1);
    scanf ("%s", B + 1);
    lA = strlen (A + 1);
    lB = strlen (B + 1);
    //printf ("0 ");
    for (int i = 2; i <= lA; i++) {
        while (mf > 0 && A[i] != A[mf + 1])
            mf = F[mf];
        if (A[i] == A[mf + 1])
            mf++;
        F[i] = mf;
        //printf ("%d ", F[i]);
    }
    for (int i = 1, mf = 0; i <= lB; i++) {
        while (mf > 0 && A[mf + 1] != B[i])
            mf = F[mf];
        if (A[mf + 1] == B[i])
            mf++;
        if (mf == lA) {
            printf ("%d\n", i - lA + 1);
            mf = F[mf];
        }
    }
}
```


3.2. Suffix Array

```
# include <cstdio>
# include <algorithm>
# include <cstring>
using namespace std;
# define MAXN 1010

int N, K, k;
int pos[MAXN], suf[MAXN], T[MAXN], LCP[MAXN];
char word[MAXN];
bool cmp(const int &a, const int &b) {
    if (pos[a] != pos[b])
        return pos[a] < pos[b];
    if (a + K < N && b + K < N)
        return pos[a + K] < pos[b + K];
    return a > b;
}

int main() {
    freopen("suffixarr.in", "r", stdin);
    freopen("suffixarr.out", "w", stdout);
    scanf("%s", &word);
    N = strlen(word);
    for (int i = 0; i < N; i++) {
        suf[i] = i;
        pos[i] = word[i];
        for (K = 0; K < N; K ? K *= 2 : K++) {
            sort(suf, suf + N, cmp);
            for (int i = 1; i < N; i++)
                T[i] = T[i - 1] + cmp(suf[i - 1], suf[i]);
            for (int j = 0; j < N; j++)
                pos[suf[j]] = T[j];
        }
        for (int i = k = 0; i < N; i++) {
            if (pos[i] == N - 1) continue;
            for (int j = suf[pos[i] + 1];
                 j + k < N &&
                 i + k < N &&
                 word[j + k] == word[i + k]; k++);
            LCP[pos[i]] = k;
        }
        for (int i = 0; i < N; i++)
            printf("%d - %s\n", LCP[i], word + suf[i]);
    }
}
```

4. Dynamic

4.1. Edit Distance

```
#include <stdio>
#include <algorithm>
#define RANG 5010
using namespace std;

int lA, lB, maxl, s, T[RANG][RANG];
char A[RANG], B[RANG];

int main () {
    freopen ("ED.in", "r", stdin);
    freopen ("ED.ou", "w", stdout);
    scanf ("%s\n %s\n", A + 1, B + 1);
    lA = strlen (A + 1);
    lB = strlen (B + 1);
    maxl = max (lA, lB);
    for (int i = 0; i <= maxl; i++)
        T[i][0] = T[0][i] = i;
    for (int i = 1; i <= lA; i++)
        for (int j = 1; j <= lB; j++) {
            s = 1;
            if (A[i] == B[j])
                s = 0;
            T[i][j] = min (min (T[i][j - 1] + 1, T[i - 1][j] + 1), T[i -
1][j - 1] + s);
        }
    printf ("%d\n", T[lA][lB]);
}
```

4.2. Longest Comun Subsequence

```
#include <stdio>
#include <cstring>
#define RANG 100
using namespace std;

int lA, lB, T[RANG][RANG];
char A[RANG], B[RANG];

int main() {
    freopen ("LCS.in", "r", stdin);
    freopen ("LCS.out", "w", stdout);
    scanf ("%s\n", A + 1);
    scanf ("%s", B + 1);
    lA = strlen (A + 1);
    lB = strlen (B + 1);
    for (int i = 1; i <= lB; i++)
        for (int j = 1; j <= lA; j++)
            if (B[i] == A[j])
                T[i][j] = T[i - 1][j - 1] + 1;
            else
                T[i][j] = max (T[i - 1][j], T[i][j - 1]);
    printf ("%d\n", T[lB][lA]);
}
```

4.3. Longest Increasing Or Decreasing Subsequence

```
(<) --> lower_bound
(<=) --> upper_bound
#include <cstdio>
#include <algorithm>
#define RANG 100
using namespace std;

int n, m, up, N[RANG], SOL[RANG], ID[RANG], L[RANG];
void write (int ID) {
    if (ID) {
        write (L[ID]);
        printf ("%d ", N[ID]);
    }
}

int main() {
    freopen ("LIS.in", "r", stdin);
    freopen ("LIS.ou", "w", stdout);
    scanf ("%d", &n);
    for (int i = 1; i <= n; i++)
        scanf ("%d", &N[i]);
    for (int i = 1; i <= n; i++) {
        if (SOL[m] <= N[i]) {
            SOL[++m] = N[i];
            ID[m] = i;
            L[i] = ID[m - 1];
        }
        else {
            up = upper_bound (SOL + 1, SOL + m + 1, N[i]) - SOL;
            SOL[up] = N[i];
            ID[up] = i;
            L[i] = ID[up - 1];
        }
    }
    printf ("%d\n", m);
    write (ID[m]);
}
```

4.4. Pack with encore

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

int N, cant, i, j, tam[10000], val[10000], cos[10000];
bool Dp[10000];
int main () {
    freopen ( "mochila.in", "r", stdin );
    freopen ( "mochila.out", "w", stdout );
    scanf ( "%d %d", &cant, &N );
    for ( i = 1; i <= cant; i++ )
        scanf ( "%d %d", &tam[i], &val[i] );
    Dp[0] = true;
    for ( i = 1; i <= cant; i++ )
        for ( j = tam[i]; j <= N; j++ )
            if ( Dp[j - tam[i]] ) {
                Dp[j] = true;
                //cos[j] >?= cos[j - tam[i]] + val[i];
                if ( cos[j] < cos[j - tam[i]] + val[i] ) {
                    cos[j] = cos[j - tam[i]] + val[i];
                }
            }
    printf ( "%d", cos[N] );
}
```

4.5. Pack without encore

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

int N, cant, i, j, tam[1000], val[1000], cos[1000];
bool Dp[1000];

int main () {
    freopen ( "mochila.in", "r", stdin );
    freopen ( "mochila.out", "w", stdout );

    scanf ( "%d %d", &N, &cant );
    for ( i = 1; i <= cant; i++ )
        scanf ( "%d %d", &tam[i], &val[i] );
    Dp[0] = true;
    for ( i = 1; i <= cant; i++ )
        for ( j = N; j >= tam[i]; j-- )
            if ( Dp[j - tam[i]] ) {
                Dp[j] = true;
                //cos[j] >?= cos[j - tam[i]] + val[i];
                if ( cos[j] < cos[j - tam[i]] + val[i] )
                    cos[j] = cos[j - tam[i]] + val[i];
            }
    printf ( "%d", cos[N] );
}
```

4.6. Counting Change

```
#include <cstdio>
#define MAXTOTAL 10000
using namespace std;

long long nway[MAXTOTAL+1];
//Asuma que tienes 5 tipos diferentes de dinero
int coin[1000], i,j,n,v,c;

int main() {
    freopen ( "cchange.in", "r", stdin );
    freopen ( "cchange.out", "w", stdout );

    scanf ( "%d %d", &n, &v );
    for ( i = 0; i < v; i++ )
        scanf ("%d", &coin[i] );
    nway[0] = 1;
    for ( i = 0; i < v; i ++ ) {
        c = coin[i];
        for ( j = c; j <= n; j++ )
            nway[j] += nway[j-c];
    }
    printf ( "%lld\n", nway[n] );
}
```

4.7. Accumulative Table

```
#include <cstdio>
#define RANG 100
using namespace std;

int f, c, T[RANG][RANG];

int main() {
    freopen ( "tabla.in", "r", stdin );
    freopen ( "tabla.out", "w", stdout );

    scanf ("%d %d", &f, &c);
    for ( int i = 1; i <= f; i++ )
        for ( int j = 1; j <= c; j++ ) {
            scanf ("%d", &T[i][j]);
            T[i][j] += T[i - 1][j] + T[i][j - 1] - T[i - 1][j - 1];
        }
    for ( int i = 1; i <= f; i++ ) {
        for ( int j = 1; j < c; j++ )
            printf ("%d ", T[i][j]);
        printf ("%d\n", T[i][c]);
    }
}
```

4.8. Catcher

```
#include <cstdio>
#include <algorithm>
#define RANG 100
using namespace std;

int cn, sol, N[RANG], C[RANG];

int main () {
    freopen ("catcher.in", "r", stdin);
    freopen ("catcher.out", "w", stdout);

    scanf ("%d", &cn);
    for ( int i = 0; i < cn; i++ )
        scanf ( "%d", &N[i] );
    int parent[100], pos;
    C[cn - 1] = 1;
    for ( int i = cn - 2; i >= 0; i-- ) {
        for ( int j = i + 1; j < cn; j++ )
            if (N[i] > N[j] && C[i] < C[j]) {
                C[i] = C[j];
                parent[i] = j;
            }
        ++C[i];
        if ( sol < C[i] ) {
            sol = C[i];
            pos = i;
        }
    }

    printf ( "%d\n", sol );
    printf ( "%d ", N[pos] );
    for ( int j = 0; j < sol - 1; j ++ ) {
        printf ( "%d ", N[parent[pos]] );
        pos = parent[pos];
    }
}
```

5. Geometry

5.1. Closest Pair Of Points (Convex Hull)

```
#include <cstdio>
#include <cmath>
#include <algorithm>
#define RANG 1000010
#define oo 1 << 30
using namespace std;

struct two {
    double x, y;
    bool operator < (const two &p) const {
        if (x != p.x)
            return x < p.x;
        return y < p.y;
    }
} P[RANG], T[RANG];
int l, lim = 1, top;
double dsol = oo;
double cross (const two &pf, const two &p1, const two &p2) {
    double m1 = (p2.y - pf.y) * (p1.x - pf.x);
    double m2 = (p2.x - pf.x) * (p1.y - pf.y);
    return m1 - m2;
}
double dist (const two &p1, const two &p2) {
    return sqrt ((p2.y - p1.y) * (p2.y - p1.y) + (p2.x - p1.x) * (p2.x -
p1.x));
}
main () {
    freopen ("CPP.in", "r", stdin);
    freopen ("CPP.ou", "w", stdout);
    scanf ("%d", &l);
    for (int i = 0; i < l; i++)
        scanf ("%lf %lf", &P[i].x, &P[i].y);
    sort (P, P + l);
    T[++top] = P[0];
    T[++top] = P[1];
    for (int i = 2; i < l; i++) {
        while (top > lim && cross (T[top - 1], T[top], P[i]) < 0)
            top--;
        T[++top] = P[i];
        dsol = min (dsol, dist (T[top - 1], T[top]));
    }
    lim = top;
    T[++top] = T[l - 2];
    T[++top] = T[l - 3];
    for (int i = l - 4; i >= 0; i--) {
        while (top > lim && cross (T[top - 1], T[top], P[i]) < 0)
            top--;
        T[++top] = P[i];
        dsol = min (dsol, dist (T[top - 1], T[top]));
    }
    printf ("%lf\n", dsol);
}
```

5.2. Closest Pair Of Points (Sweep Line)

```
#include <cstdio>
#include <algorithm>
#include <cmath>
#include <set>
#define RANG 1000010
#define oo 1 << 30
using namespace std;
struct two {
    double x, y;
} P[RANG], *last = P;
struct cmp_x {
    bool operator () (const two &p1, const two &p2) const {
        return p1.x < p2.x;
    }
};
struct cmp_y {
    bool operator () (const two &p1, const two &p2) const {
        return p1.y < p2.y;
    }
};
multiset <two, cmp_y> Q;
multiset <two, cmp_y>::iterator lo, hi;
double dsol = oo;
int l;
double dist (const two &p1, const two &p2) {
    return sqrt ((p2.y - p1.y) * (p2.y - p1.y) + (p2.x - p1.x) * (p2.x -
p1.x));
}

main () {
    freopen ("CPP.in", "r", stdin);
    freopen ("CPP.ou", "w", stdout);
    scanf ("%d", &l);
    for (int i = 0; i < l; i++)
        scanf ("%lf %lf", &P[i].x, &P[i].y);
    sort (P, P + l, cmp_x());
    for (two *i = P; i < &P[l]; i++) {
        while (i -> x - last -> x >= dsol)
            Q.erase (Q.find (*last++));
        lo = Q.lower_bound ((two) {i -> x, i -> y - dsol});
        hi = Q.upper_bound ((two) {i -> x, i -> y + dsol});
        for (; lo != hi; lo++)
            dsol = min (dsol, dist (*lo, *i));
        Q.insert (*i);
    }
    printf ("%lf\n", dsol);
}
```

5.3. Convex Hull (Graham Scan)

```
#include <cstdio>
#include <algorithm>
#define RANG 100010
using namespace std;
double x, y;
int l, top, lim = 1;
struct two {
    double x, y;
    bool operator < (const two &p) const {
        if (x != p.x)
            return x < p.x;
        return y < p.y;
    }
} P[RANG], T[RANG];
double cross (const two &pf, const two &p1, const two
&p2) {
    double m1 = (p2.y - pf.y) * (p1.x - pf.x);
    double m2 = (p2.x - pf.x) * (p1.y - pf.y);
    return m1 - m2;
}
main () {
    freopen ("convex_hull.in", "r", stdin);
    freopen ("convex_hull.ou", "w", stdout);
    scanf ("%d", &l);
    for (int i = 0; i < l; i++) {
        scanf ("%lf %lf", &x, &y);
        P[i] = (two) {x, y};
    }
    sort (P, P + l);
    for (int i = 0; i < l; i++) {
        while (top > lim && cross (T[top - 1],
T[top], P[i]) <= 0)
            top--;
        T[++top] = P[i];
    }
    lim = top;
    for (int i = l - 1; i >= 0; i--) {
        while (top > lim && cross (T[top - 1], T[top],
P[i]) <= 0)
            top--;
        T[++top] = P[i];
    }
    printf ("%d\n", top);
    for (int i = 1; i <= top; i++)
        printf ("%01f %01f\n", T[i].x, T[i].y);
}
```


5.4. Union Area (Segment Tree + Sweep Line)

```
# include <cstdio>
# include <vector>
# include <algorithm>
# define MAXN 10010
# define MAXC 30010
using namespace std;

struct event {
    int start, lo, hi, flag;
    event(int x, int i, int j, int s) {
        start = x;
        lo = i;
        hi = j;
        flag = s;
    }
    bool operator <(const event &q)
    const {return start < q.start;}
};

int N, last, sol;
int tree[MAXC * 3], cant[MAXC * 3];
vector <event> L;
void update (int n, int lo, int hi, int s, int f, int
val) {
    if (lo > f || hi < s || lo > hi)
        return ;
    if (lo >= s && hi <= f)
        tree[n] += val;
    else {
        int mid = (lo + hi)/2;
        update(n * 2, lo, mid, s, f, val);
        update(n * 2 + 1, mid + 1, hi, s, f, val);
    }
    if (!tree[n])
        if (lo == hi)
            cant[n] = 0;
        else cant[n] = cant[n * 2] + cant[n * 2 + 1];
    else cant[n] = (hi - lo) + 1;
}

int main() {
    freopen("unionarea.in", "r", stdin);
    freopen("unionarea.out", "w", stdout);
    scanf("%d", &N);
    for (int i = 1; i <= N; i++) {
        int a, b, c, d;
        scanf("%d%d%d%d", &a, &b, &c, &d);
        if (b > d)
            swap(b, d);
        L.push_back(event(a, b, d - 1, 1));
        L.push_back(event(c, b, d - 1, -1));
    }
    sort(L.begin(), L.end());
    last = L[0].start;
    for (int i = 0; i < L.size(); i++) {
        sol += (L[i].start - last) * cant[1];
        last = L[i].start;
        update(1, 0, MAXC, L[i].lo, L[i].hi,
            L[i].flag);
    }
    printf("%d\n", sol);
}
```

6. Math

6.1. Gaus Comun Divisor

```
#include <stdio>
#include <algorithm>
using namespace std;

int a , b;
int GCD (int a, int b) {
    while (a) {
        swap (a, b);
        a %= b;
    }
    return b;
}

main () {
    freopen ("GCD.in", "r", stdin);
    freopen ("GCD.ou", "w", stdout);
    scanf ("%d %d", &a, &b);
    printf ("MCD es %d\n", GCD (a, b));
    printf ("MCM es %d\n", a * b / GCD (a, b));
}
```

6.2. Big Mod (b^e) % m

```
#include <stdio>
using namespace std;

long long q, b, e, m, sq;
long long square (long long n) {
    return n * n;
}

long long big_mod (int b, int e, int m) {
    if (!e)
        return 1;
    if (e % 2 == 0)
        return square (big_mod (b, e / 2, m)) % m;
    return (b % m * big_mod (b, e - 1, m)) % m;
}

main () {
    scanf ("%d", &q);
    while (q--) {
        scanf ("%I64d %I64d %I64d", &b, &e, &m);
        printf ("%I64d\n", big_mod (b, e, m));
    }
}
```

6.3. Counting Combinations $C(n, k)$

```
#include <cstdio>
#define RANG 110
using namespace std;

long long q, n, k, T[RANG][RANG];
//O (n)
double comb (long long n, long long k) {
    double comb = 1;
    if (n - k < k)
        k = n - k;
    for (int i = 2; i <= k; i++)
        comb /= i;
    k = n - k;
    for (int i = n; i > k; i--)
        comb *= i;
    return comb + 0.01;
}
//O (k)
double combfast (long long n, long long k) {
    double comb = 1;
    if (n - k < k)
        k = n - k;
    for (int i = 1; i <= k; i++)
        comb = comb * (n - k + i) / i;
    return comb + 0.01;
}
//O (1)
long long pascal_tri (long long n, long long k) {
    return T[n][k];
}

main () {
    scanf ("%d", &q);
    for (int i = 0; i < RANG; i++)
        T[i][0] = T[i][i] = 1;
    for (int i = 1; i < RANG; i++)
        for (int j = 1; j < RANG; j++)
            T[i][j] = T[i - 1][j] + T[i - 1][j - 1];
    while (q--) {
        scanf ("%lld %lld", &n, &k);
        printf ("%0.1f\n", comb (n, k));
        printf ("%0.1f\n", combfast (n, k));
        printf ("%lld\n", pascal_tri (n, k));
    }
}
```

6.4. Exponentiation n^k

```
#include <cstdio>
#include <cmath>
using namespace std;

long long q, n, k;
long long square (long long n) {
    return n * n;
}
//O (N)
long long slowexp (long long n, long long k) {
    long long sol = 1;
    for (long long i = 1; i <= k; i++)
        sol *= n;
    return sol;
}
//O (log2 (N))
long long fastexpr (long long n, long long k) {
    if (!k)
        return 1;
    if (k % 2 == 0)
        return square (fastexpr (n, k / 2));
    return n * fastexpr (n, k - 1);
}
//O (log2 (N))
long long fastexpi (long long n, long long k) {
    long long sol = 1;
    while (k) {
        if (k & 1)
            sol *= n;
        n *= n;
        k >>= 1;
    }
    return sol;
}

main () {
    scanf ("%lld", &q);
    while (q--) {
        scanf ("%lld %lld", &n, &k);
        printf ("%lld\n", slowexp (n, k));
        printf ("%lld\n", fastexpr (n, k));
        printf ("%lld\n", fastexpi (n, k));
        printf ("%0.1f\n", pow (n, (double) k));
        printf ("%0.1f\n", exp (log (n) * k));
    }
}
```

6.5. Fórmulas

$$Pi = 4 * (2/3 * 4/3 * 4/5 * 6/5 * ...) = 3,14159265358979$$

$$Pi = 2 * \arccos(0);$$

$$\text{Golden Number} = \frac{\sqrt{5} + 1}{2} = 1,61803398874989$$

Fibonacci Number

$$F(n) = F(n - 1) * \text{Golden Number}$$

Sumatoria de n

$$S(n) = \frac{n * (n + 1)}{2}$$

Números Catalan

$$C(n) = C(n, 2n) - C(n - 1, 2n) = \frac{(2n)!}{n! * (n + 1)!}$$

Variaciones

$$V(n, p) = \frac{n!}{(n - p)!}$$

$$V'(n, p) = n^p;$$

Permutaciones

$$P(n) = n!$$

$$P'n(n_1, n_2, \dots, n_k) = \frac{n!}{n_1! * n_2! * \dots * n_k!}$$

Donde n es el total de elementos y n(k) la cantidad de repeticiones

Combinaciones

$$C(n, p) = C(n - 1, p - 1) + C(n - 1, p) = \frac{n!}{(n - p)! * p!}$$

Mínimo Común Múltiplo

$$MCM(a, b) = \frac{a * b}{MCD(a, b)}$$

Carmichael Numbers >= 3 Primes Factors

561, 1105, 1729, 2465, 2821, 6601, 8911, 10585, 15841, 29341, 41041, 466

57, 52633, 62745, 6397

3

Sumatoria de n^2

$$1^2 + 2^2 + \dots + n^2 = n * (n + 1) * (2n + 1) / 6;$$

Sumatoria de x^n

$$x^0 + x^1 + \dots + x^n = (x^{n+1} - 1) / (x - 1);$$

Sumatoria de los divisores de un número

$$N = p_1^{a_1} + p_2^{a_2} + \dots + p_k^{a_k}$$

sea $p_1 < p_2 < \dots < p_k$ números primos.

$$S(N) = \frac{p_1^{a_1+1} - 1}{p_1 - 1} * \dots * \frac{p_k^{a_k+1} - 1}{p_k - 1}$$