DLX模板

[一、 精确覆盖 1](#_Toc401600945)

[二、 多重覆盖 4](#_Toc401600946)

[三、 dragon\_DLX 7](#_Toc401600947)

[四、 kuangbin\_DLX 10](#_Toc401600948)

## 精确覆盖

验题: POJ3074

struct DLX{

struct Node{

Node \*L, \*R, \*U, \*D;

int col, row;

} \*head, \*row[Maxn], \*col[Maxm], node[Maxn \* Maxm];

int colsum[Maxm], cnt;

/\* dancing link

\* 精确覆盖问题

\* 可以添加迭代加深优化：

\* 1）枚举深度h；

\* 2）若当前深度+predeep > h return false；

\*/

/\*

int predeep() {

bool vis[Maxm];

memset(vis, 0, sizeof(vis));

int ret = 0;

for (Node \*p = head->R; p != head; p = p->R)

if (!vis[p->col]) {

ret ++ ;

vis[p->col] ++ ;

for (Node \*q = p->D; q != p; q = p->D)

for (Node \*r = q->R; r != q; r = r->R)

vis[r->col] = true;

}

return ret;

}

//\*/

void init(int mat[][Maxm], int n, int m) {

cnt = 0;

memset(colsum, 0, sizeof (colsum) );

head = &node[cnt ++ ];

for(int i = 1; i <= n; i ++ )

row[i] = &node[cnt ++ ];

for(int j = 1; j <= m; j ++ )

col[j] = &node[cnt ++ ];

head->D = row[1], row[1]->U = head;

head->R = col[1], col[1]->L = head;

head->U = row[n], row[n]->D = head;

head->L = col[m], col[m]->R = head;

head->row = head->col = 0;

for(int i = 1; i <= n; i ++ ) {

if (i != n) row[i]->D = row[i + 1];

if (i != 1) row[i]->U = row[i - 1];

row[i]->L = row[i]->R = row[i];

row[i]->row = i, row[i]->col = 0;

}

for(int i = 1; i <= m; i ++ ) {

if (i != m) col[i]->R = col[i + 1];

if (i != 1) col[i]->L = col[i - 1];

col[i]->U = col[i]->D = col[i];

col[i]->col = i, col[i]->row = 0;

}

for(int i = n; i > 0; i -- )

for(int j = m; j > 0; j -- )

if(mat[i][j]) {

Node \*p = &node[cnt ++ ];

p->R = row[i]->R, row[i]->R->L = p;

p->L = row[i], row[i]->R = p;

p->D = col[j]->D, col[j]->D->U = p;

p->U = col[j], col[j]->D = p;

p->row = i;

p->col = j;

colsum[j] ++ ;

}

}

void remove(Node \*c) {

c->L->R = c->R;

c->R->L = c->L;

for(Node \*p = c->D; p != c; p = p->D) {

for(Node \*q = p->R; q != p; q = q->R) {

q->U->D = q->D;

q->D->U = q->U;

colsum[q->col] -- ;

}

}

}

void resume(Node \*c) {

for(Node \*p = c->U; p != c; p = p->U) {

for(Node \*q = p->L; q != p; q = q->L) {

q->U->D = q;

q->D->U = q;

colsum[q->col] ++ ;

}

}

col[c->col]->L->R = col[c->col];

col[c->col]->R->L = col[c->col];

}

int dfs(int deep) {

if(head->R == head) return deep;

Node \*p, \*q = head->R;

for(p = head->R; p != head; p = p->R)

if(colsum[p->col] < colsum[q->col])

q = p;

remove(q);

for(p = q->D; p != q; p = p->D) {

for(Node\* r = p->R; r != p; r = r->R)

if (r->col != 0)

remove (col[r->col]);

/\*--------可修改区域-----------\*/

ans[deep] = p->row;

/\*-----------------------------\*/

int sta = dfs (deep + 1);

if(sta) return sta;

for(Node\* r = p->L; r != p; r = r->L)

if(r->col != 0)

resume (col[r->col]);

}

resume(q);

return false;

}

} dlx;

int mat[Maxn][Maxm];

int mem[Maxn][3]; //记录每行代表哪一格填几

//数独填充(x,y)=v

void addline(int x, int y, int v) {

int i, j;

n++;

mem[n][0] = x;

mem[n][1] = y;

mem[n][2] = v;

for(i = 0; i < Maxm; i++) mat[n][i] = 0;

mat[n][x \* 9 + y + 1] = 1;

mat[n][81 + x \* 9 + v] = 1;

mat[n][162 + y \* 9 + v] = 1;

mat[n][243 + (3 \* (x / 3) + y / 3) \* 9 + v] = 1;

}

## 多重覆盖

验题 HDU3498, HDU5046

struct DLX{

struct Node{

Node \*L, \*R, \*U, \*D;

int col, row;

} \*head, \*row[Maxn], \*col[Maxm], node[Maxn \* Maxm];

int colsum[Maxm], cnt;

/\* dancing link

\* 精确覆盖问题

\* 可以添加迭代加深优化：

\* 1）枚举深度h；

\* 2）若当前深度+predeep > h return false；

\*

\*/

///\*

int predeep(){

bool vis[Maxm];

Node \* p, \*q, \*r;

memset(vis, 0, sizeof(vis));

int ret = 0;

for(p = head->R; p != head; p = p->R) {

if(!vis[p->col]) {

ret++;

vis[p->col]++;

for(q = p->D; q != p; q = q->D) {

for(r = q->R; r != q; r = r->R) {

vis[r->col] = true;

}

}

}

}

return ret;

}

//\*/

void init(int mat[][Maxm], int n, int m) {

cnt = 0;

int i, j;

Node \* p;

memset(colsum, 0, sizeof(colsum));

head = &node[cnt++];

for(i = 1; i <= n; i++) row[i] = &node[cnt++];

for(j = 1; j <= m; j++) col[j] = &node[cnt++];

head->D = row[1], row[1]->U = head;

head->R = col[1], col[1]->L = head;

head->U = row[n], row[n]->D = head;

head->L = col[m], col[m]->R = head;

head->row = head->col = 0;

for(i = 1; i <= n; i++) {

if(i != n) row[i]->D = row[i + 1];

if(i != 1) row[i]->U = row[i - 1];

row[i]->L = row[i]->R = row[i];

row[i]->row = i; row[i]->col = 0;

}

for(i = 1; i <= m; i++) {

if(i != m) col[i]->R = col[i + 1];

if(i != 1) col[i]->L = col[i - 1];

col[i]->U = col[i]->D = col[i];

col[i]->col = i; col[i]->row = 0;

}

for(i = n; i > 0; i--) {

for(j = m; j > 0; j--) {

if(mat[i][j]) {

p = &node[cnt++];

p->R = row[i]->R, row[i]->R->L = p;

p->L = row[i], row[i]->R = p;

p->D = col[j]->D, col[j]->D->U = p;

p->U = col[j], col[j]->D = p;

p->row = i;

p->col = j;

colsum[j]++;

}

}

}

}

void remove(Node \*c) {

Node \* p;

for(p = c->D; row[p->row] != row[c->row]; p = p->D) {

p->R->L = p->L; p->L->R = p->R;

}

}

void resume(Node \*c) {

Node \* p;

for(p = c->U; row[p->row] != row[c->row]; p = p->U) {

p->L->R = p->R->L = p;

}

}

int dfs(int deep) {

if(head->R == head) return deep <= K;

if(deep + predeep() > K) return false;

Node \*p, \*q = head->R, \*r;

for(p = head->R; p != head; p = p->R) {

if(colsum[p->col] < colsum[q->col]) q = p;

}

for(p = q->D; p != q; p = p->D) {

remove(p);

for(r = p->R; r != p; r = r->R) {

if(r->col != 0) remove(r);

}

/\*--------可修改区域-----------\*/

// ans[deep] = p->row;

/\*-----------------------------\*/

int sta = dfs(deep + 1);

if(sta) return sta;

for(r = p->L; r != p; r = r->L) {

if(r->col != 0) resume(r);

}

resume(p);

}

return false;

}

} dlx;

## dragon\_DLX

/\* dancing link

\* 精确覆盖问题

\* 可以添加迭代加深优化：

\* 1）枚举深度h；

\* 2）若当前深度+predeep > h return false；

\*

int predeep() {

bool vis[Maxm];

memset(vis, 0, sizeof(vis));

int ret = 0;

for (Node \*p = head->R; p != head; p = p->R)

if (!vis[p->col]) {

ret ++ ;

vis[p->col] ++ ;

for (Node \*q = p->D; q != p; q = p->D)

for (Node \*r = q->R; r != q; r = r->R)

vis[r->col] = true;

}

return ret;

}

\* \*/

#define Maxn 1010

#define Maxm 1010

struct Node {

Node \*L, \*R, \*U, \*D;

int col, row;

} \*head, \*row[Maxn], \*col[Maxm], node[Maxn \* Maxm];

int colsum[Maxm], cnt;

void init(int mat[][Maxm], int n, int m) {

cnt = 0;

memset(colsum, 0, sizeof(colsum));

head = &node[cnt ++ ];

for (int i = 1; i <= n; i ++ )

row[i] = &node[cnt ++ ];

for (int j = 1; j <= m; j ++ )

col[j] = &node[cnt ++ ];

head->D = row[1], row[1]->U = head;

head->R = col[1], col[1]->L = head;

head->U = row[n], row[n]->D = head;

head->L = col[m], col[m]->R = head;

head->row = head->col = 0;

for (int i = 1; i <= n; i ++ ) {

if (i != n) row[i]->D = row[i + 1];

if (i != 1) row[i]->U = row[i - 1];

row[i]->L = row[i]->R = row[i];

row[i]->row = i, row[i]->col = 0;

}

for (int i = 1; i <= m; i ++ ) {

if (i != m) col[i]->R = col[i + 1];

if (i != 1) col[i]->L = col[i - 1];

col[i]->U = col[i]->D = col[i];

col[i]->col = i, col[i]->row = 0;

}

for (int i = n; i > 0; i -- )

for (int j = m; j > 0; j -- )

if (mat[i][j]) {

Node \*p = &node[cnt ++ ];

p->R = row[i]->R, row[i]->R->L = p;

p->L = row[i], row[i]->R = p;

p->D = col[j]->D, col[j]->D->U = p;

p->U = col[j], col[j]->D = p;

p->row = i;

p->col = j;

colsum[j] ++ ;

}

}

/\*多重覆盖只需删除列，无需对应行删除

void remove(Node \*c) {

for (Node \*p = c->D; p != c; p = p->D) {

p->L->R = p->R;

p->R->L = p->L;

}

}

\*/

void remove(Node \*c) {

c->L->R = c->R;

c->R->L = c->L;

for (Node \*p = c->D; p != c; p = p->D) {

for (Node \*q = p->R; q != p; q = q->R) {

q->U->D = q->D;

q->D->U = q->U;

colsum[q->col] -- ;

}

}

}

void resume(Node \*c) {

for (Node \*p = c->U; p != c; p = p->U) {

for (Node \*q = p->L; q != p; q = q->L) {

q->U->D = q;

q->D->U = q;

colsum[q->col] ++ ;

}

}

col[c->col]->L->R = col[c->col];

col[c->col]->R->L = col[c->col];

}

int ans[Maxm];

int dfs(int deep) {

if (head->R == head) return deep;

Node \*p, \*q = head->R;

for (p = head->R; p != head; p = p->R)

if (colsum[p->col] < colsum[q->col])

q = p;

remove(q);

for (p = q->D; p != q; p = p->D) {

for (Node\* r = p->R; r != p; r = r->R)

if (r->col != 0)

remove(col[r->col]);

/\*--------可修改区域-----------\*/

ans[deep] = p->row;

/\*-----------------------------\*/

int sta = dfs(deep + 1);

if (sta != -1) return sta;

for (Node\* r = p->L; r != p; r = r->L)

if (r->col != 0)

resume(col[r->col]);

}

resume(q);

return -1;

}

## kuangbin\_DLX

const int maxnode = 4000;

const int MaxM = 70;

const int MaxN = 70;

int K;

struct DLX

{

int n,m,size;

int U[maxnode],D[maxnode],R[maxnode],L[maxnode],Row[maxnode],Col[maxnode];

int H[MaxN],S[MaxM];

int ands,ans[MaxN];

void init(int \_n,int \_m)

{

n = \_n;

m = \_m;

for(int i = 0;i <= m;i++)

{

S[i] = 0;

U[i] = D[i] = i;

L[i] = i-1;

R[i] = i+1;

}

R[m] = 0; L[0] = m;

size = m;

for(int i = 1;i <= n;i++)

H[i] = -1;

}

void Link(int r,int c)

{

++S[Col[++size]=c];

Row[size] = r;

D[size] = D[c];

U[D[c]] = size;

U[size] = c;

D[c] = size;

if(H[r] < 0)H[r] = L[size] = R[size] = size;

else

{

R[size] = R[H[r]];

L[R[H[r]]] = size;

L[size] = H[r];

R[H[r]] = size;

}

}

void remove(int c)

{

for(int i = D[c];i != c;i = D[i])

L[R[i]] = L[i], R[L[i]] = R[i];

}

void resume(int c)

{

for(int i = U[c];i != c;i = U[i])

L[R[i]]=R[L[i]]=i;

}

bool v[maxnode];

int f()

{

int ret = 0;

for(int c = R[0];c != 0;c = R[c])v[c] = true;

for(int c = R[0];c != 0;c = R[c])

if(v[c])

{

ret++;

v[c] = false;

for(int i = D[c];i != c;i = D[i])

for(int j = R[i];j != i;j = R[j])

v[Col[j]] = false;

}

return ret;

}

bool Dance(int d)

{

if(d + f() > K)return false;

if(R[0] == 0)return d <= K;

int c = R[0];

for(int i = R[0];i != 0;i = R[i])

if(S[i] < S[c])

c = i;

for(int i = D[c];i != c;i = D[i])

{

remove(i);

for(int j = R[i];j != i;j = R[j])remove(j);

if(Dance(d+1))return true;

for(int j = L[i];j != i;j = L[j])resume(j);

resume(i);

}

return false;

}

};

DLX g;

struct Point

{

int x,y;

void input()

{

scanf("%d%d",&x,&y);

}

}city[MaxM];

long long dis(Point a,Point b)

{

return (long long)abs(a.x-b.x)+(long long)abs(a.y-b.y);

}

int main()

{

//freopen("E.in","r",stdin);

//freopen("E.out","w",stdout);

int T;

int n;

scanf("%d",&T);

int iCase = 0;

while(T--)

{

iCase++;

scanf("%d%d",&n,&K);

assert(n >= 1 && n <= 60 && K >= 1 && K <= n);

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

city[i].input();

assert(abs(city[i].x) <= 1000000000);

assert(abs(city[i].y) <= 1000000000);

}

long long l = 0, r = 100000000000LL;

long long ans = 0;

while(l <= r)

{

long long mid = (l+r)/2;

g.init(n,n);

for(int i = 0;i < n;i++)

for(int j = 0;j < n;j++)

if(dis(city[i],city[j]) <= mid)

g.Link(i+1,j+1);

if(g.Dance(0)){r = mid-1;ans = mid;}

else l = mid+1;

}

printf("Case #%d: %I64d\n",iCase,ans);

}

return 0;

}