```
/***********Template Starts Here***********/
#include<bits/stdc++.h>
#define pb push_back
#define nl puts ("")
#define sp printf ( " " )
#define phl printf ( "hello\n" )
#define ff first
#define ss second
#define POPCOUNT __builtin_popcountll
#define RIGHTMOST __builtin_ctzll
#define LEFTMOST(x) (63-__builtin_clzll((x)))
#define MP make_pair
#define FOR(i,x,y) for(int i = (x) ; i <= (y) ; ++i)
#define ROF(i,x,y) for(int i = (y) ; i >= (x) ; --i)
#define CLR(x,y) memset(x,y,sizeof(x))
#define UNIQUE(V) (V).erase(unique((V).begin(),(V).end()),(V).end())
#define MIN(a,b) ((a)<(b)?(a):(b))
#define MAX(a,b) ((a)>(b)?(a):(b))
#define NUMDIGIT(x,y) (((int)(log10((x))/log10((y))))+1)
#define SQ(x) ((x)*(x))
#define ABS(x) ((x)<0?-(x):(x))
#define FABS(x) ((x)+eps<0?-(x):(x))
#define ALL(x) (x).begin(),(x).end()
#define LCM(x,y) (((x)/gcd((x),(y)))*(y))
#define SZ(x) ((int)(x).size())
#define NORM(x) if(x>=mod)x-=mod;

using namespace std;

typedef long long vlong;
typedef unsigned long long uvlong;
typedef pair < int, int > pii;
typedef pair < vlong, vlong > pll;
typedef vector<pii> vii;
typedef vector<int> vi;

const vlong inf = 2147383647;
const double pi = 2 * acos ( 0.0 );
const double eps = 1e-9;

#ifdef forthright48
     #include <ctime>
     clock_t tStart = clock();
     #define debug(args...) {dbg,args; cerr<<endl;}
     #define   timeStamp   printf("Execution   Time:   %.2fs\n",
(double)(clock() - tStart)/CLOCKS_PER_SEC)
#else
    #define debug(args...)  // Just strip off all debug tokens
    #define timeStamp
#endif
```

```
struct debugger{
    template<typename T> debugger& operator , (const T& v){
        cerr<<v<<" ";
        return *this;
    }
}dbg;

inline vlong gcd ( vlong a, vlong b ) {
    a = ABS ( a ); b = ABS ( b );
    while ( b ) { a = a % b; swap ( a, b ); } return a;
}

vlong ext_gcd ( vlong A, vlong B, vlong *X, vlong *Y ){
    vlong x2, y2, x1, y1, x, y, r2, r1, q, r;
    x2 = 1; y2 = 0;
    x1 = 0; y1 = 1;
    for(r2=A,r1=B;r1!=0;r2=r1,r1=r,x2=x1,y2=y1,x1=x,y1=y){
        q = r2 / r1;
        r = r2 % r1;
        x = x2 - (q * x1);
        y = y2 - (q * y1);
    }
    *X = x2; *Y = y2;
    return r2;
}

inline vlong modInv ( vlong a, vlong m ) {
    vlong x, y;
    ext_gcd( a, m, &x, &y );
    if ( x < 0 ) x += m; //modInv is never negative
    return x;
}

inline vlong power ( vlong a, vlong p ) {
    vlong res = 1, x = a;
    while ( p ) {
        if ( p & 1 ) res = ( res * x );
        x = ( x * x ); p >>= 1;
    }
    return res;
}

inline vlong bigmod ( vlong a, vlong p, vlong m ) {
    vlong res = 1 % m, x = a % m;
    while ( p ) {
        if ( p & 1 ) res = ( res * x ) % m;
        x = ( x * x ) % m; p >>= 1;
    }
    return res;
}
/***********Template Ends Here***********/
```

```
struct node { int x, y, next, cap, cost; };

/*
1. Clear graph
2. Add edge
3. Assign source and sink
4. Pass limit - highest index of nodes
*/

#define NODE 30000
#define EDGE 5000000
int backup[NODE];

struct FLOW {
    int source, sink;

    int head[NODE];
    void clear() {
        e = 0;
        CLR(head,-1);
    }

    node edge[EDGE]; int e;
    void addEdge ( int u, int v, int cap ) {
        edge[e].x = u; edge[e].y = v; edge[e].cap = cap;
        edge[e].next = head[u]; head[u] = e; e++;
        edge[e].x = v; edge[e].y = u; edge[e].cap = 0;
        edge[e].next = head[v]; head[v] = e; e++;
    }

    int vis[NODE], q[NODE], now[NODE];
    bool bfs ( ) {
        memset ( vis, -1, sizeof vis );
        vis[source] = 0;
        int ini = 0, qend = 0;
        q[qend++] = source;

        while ( ini < qend && vis[sink] == -1 ) {
            int s = q[ini++];
            int i;
            for (i=head[s];i!=-1;i= edge[i].next){
                int t = edge[i].y;
                if ( vis[t] == -1 && edge[i].cap){
                    vis[t] = vis[s] + 1;
                    q[qend++] = t;
                }
            }
        }
        if ( vis[sink] != -1 ) return true;
        else return false;
    }
```

```
    int dfs ( int s, int f ) {
        if ( f == 0 ) return 0;
        if ( s == sink ) return f;
        for ( int &i=now[s];i!=-1;i=edge[i].next){
            int t = edge[i].y;
            if ( vis[s] + 1 != vis[t] ) continue;
            int pushed=dfs(t,MIN(f,edge[i].cap));
            if ( pushed ) {
                edge[i].cap -= pushed;
                edge[i^1].cap += pushed;
                return pushed;
            }
        }
        return 0;
    }

    int maxFlow ( int limit, int flow ) {
        int res = 0;
        while ( 1 ) {
            if ( flow == 0 ) break;
            if ( bfs () == false ) break;
            int i;
            for ( i=0;i<=limit;i++)now[i]= head[i];
            while (int pushed=dfs(source,flow ) ) {
                res += pushed; //Can overflow depending on Max Flow
                flow -= pushed;
            }
        }
        return res;
    }

}graph;

pair<int,int> dish[110];
int event[10100];

int main ()
{
    #ifdef forthright48
    freopen ( "input.txt", "r", stdin );
    //freopen ( "output.txt", "w", stdout );
    #endif // forthright48

    int n; scanf ( "%d", &n );
    int mx = -1;

    graph.clear();

    int x = 10000;
    graph.source = x + n + 1;
    graph.sink = x + n + 2;
```

```
    FOR(i,1,n) {
        int a, b;
        scanf ( "%d %d", &a, &b );

        FOR(j,a,b-1) {
            graph.addEdge( j, x + i, inf );
        }
    }

    FOR(i,0,x) {
        graph.addEdge( graph.source, i, 1 );
    }

    int e = graph.e;
    memcpy( backup, graph.head, sizeof backup );

    int low = 0, high = 10000;

    while ( low <= high ) {
        int mid = ( low + high ) / 2;

        ///Reset graph
        for ( int i = 0; i < e; i += 2 ) {
            graph.edge[i].cap += graph.edge[i^1].cap;
            graph.edge[i^1].cap = 0;
        }

        graph.e = e;
        memcpy( graph.head, backup, sizeof backup );

        FOR(i,1,n) {
            graph.addEdge( x + i, graph.sink, mid );
        }

        int need = mid * n;

        int f = graph.maxFlow( graph.sink, inf );

        if ( f == need ) {
            low = mid + 1;
        }
        else {
            high = mid - 1;
        }
    }

    printf ( "%d\n", (low-1) * n );

    return 0;
}
```