```cpp
#include <cstdio>
#include <climits>
#include <vector>
#include <algorithm>
#define IL inline
using namespace std;
constexpr int MN(1e6+7);

template <typename vint, typename sint, typename xint = int>
class STree
{
private:
    static constexpr xint ROOT = 1;
    struct Node
    {
        xint l, r;
        sint max, lza;
    } t[MN << 2];
    xint ll, rr, pos;
    vint vv;
#define li i<<1
#define ri i<<1|1
#define t_mid ((t[i].l+t[i].r) >> 1)
#define add_v(i, v) ({t[i].max += v, t[i].lza += v;})

#define pd(i) \
    ({ \
        if (t[i].lza) \
        { \
            add_v(li, t[i].lza); \
            add_v(ri, t[i].lza); \
            t[i].lza = 0; \
        } \
    })

    void build(const xint i, const xint l, const xint r)
    {
        t[i].l = l, t[i].r = r, t[i].lza = 0;
        if (l == r)
            t[i].max = 0;
        else
        {
            build(li, l, t_mid),
            build(ri, t_mid+1, r);
            t[i].max = max(t[li].max, t[ri].max);;
        }
    }
    void add(const xint i)
    {
        if (ll <= t[i].l && t[i].r <= rr) add_v(i, vv);
        else
```

```
        {
            pd(i);
            if (ll <= t_mid)
                add(li);
            if (rr > t_mid)
                add(ri);
            t[i].max = max(t[li].max, t[ri].max);;
        }
    }

public:
    IL void build(const xint l, const xint r)
    {
        build(ROOT, l, r);
    }
    IL void add(const xint l, const xint r, const vint v)
    {
        ll = l, rr = r, vv = v,
        add(ROOT);
    }
    IL sint gmax()
    {
        return t[ROOT].max;
    }
};


int tr[MN];
STree<int, int> st;

struct Node
{
    int l, r, len;
    IL bool operator <(const Node &o) const
    {
        return len < o.len;
    }
} a[MN];
vector<int> vals;

#define ID(x) lower_bound(vals.begin(), vals.end(), x)-vals.begin()

int main()
{
    int n, m;
    scanf("%d %d", &n, &m);

    for (int i=0; i<n; ++i)
    {
        scanf("%d %d", &a[i].l, &a[i].r);
        a[i].len = a[i].r-a[i].l;
```

```
        vals.emplace_back(a[i].l);
        vals.emplace_back(a[i].r);
    }
    sort(a, a+n);

    sort(vals.begin(), vals.end());
    vals.erase(unique(vals.begin(), vals.end()), vals.end());
    for (int i=0; i<n; ++i)
        a[i].l = ID(a[i].l), a[i].r = ID(a[i].r);

    int ans = INT_MAX;
    st.build(0, vals.size()-1);

    int l=0, r=0;
    while (r < n)
    {
        st.add(a[r].l, a[r].r, 1);
        while (st.gmax() >= m)
        {
            ans = min(ans, a[r].len - a[l].len);
            st.add(a[l].l, a[l].r, -1);
            ++l;
        }
        ++r;
    }
    if (ans == INT_MAX)
        puts("-1");
    else
        printf("%d", ans);

    return 0;
}
```