《算法设计与分析-PTA 7-5》实验报告

学　号：1004191211　　姓　名：郎文鹏

日　期：2021/10/15　　得　分：

一、实验内容：

算法设计与分析课程作业——PTA 7-5 铺设油井管道。

二、所用算法的基本思想及复杂度分析：

1. 基本思想

由题知油井均垂直向水平方向的主管道修建支线管道，所以与x坐标无关，我们只需要确定主管道y坐标使得所有主管道y坐标到主管道y坐标之和最小。将油井按照y坐标升序排序，当n为奇数时最中间的油井y坐标就是主管道y坐标最优解，当n为偶数时取处于中间位置的两个油井任意一个y坐标作为主管道y坐标最优解均可，根据编号可以确定改题目需要寻找的是油井中第$\frac{n}{2}+1$个大的坐标y。

寻找数列第k大值可以结合快排思想与减治思想进一步降低复杂度值$O(n)$，每次将选取的分界值确定位置后，如果它及左侧数量x少于k个，则需要向右侧方向寻找第$k-x$大，如果恰好等于k个则说明找到答案返回，否则需要进一步在左侧方向寻找第k大。

2. 复杂度分析

经典快排复杂度为$O(nlogn)$，由于这里只考虑一边了所以容易得知该算法复杂度为$O(n)$，但是由于经典快排复杂度最坏时可能上升到$O(n^2)$，在特意卡数据情况下该算法复杂度同样会变为$O(n^2)$。

三、源程序及注释：

```
#include <bits/stdc++.h>
#pragma GCC optimize(2)
#pragma G++ optimize(2)
#define endl "\n"
#define fi first
#define se second
```

```
#define pb push_back
#define all(x) x.begin(), x.end()
#define rep(i, x, y) for (auto i = (x); i != (y + 1); ++i)
#define dep(i, x, y) for (auto i = (x); i != (y - 1); --i)
#ifdef LOCAL
#define de(...) cout << '[' << #__VA_ARGS__ << "] = " << __VA_ARGS__ <<
endl;
#else
#define de(...)
#endif
using namespace std;
typedef long long ll;
typedef pair<ll, ll> pii;
const ll maxn = 2e6 + 5;

struct ios_in {
    inline char gc() {
        const ll MAXN = 1e5 + 100;  //读入字符串的最大长度（根据情况调整)
        static char buf[MAXN], *l, *r;
        return (l == r) && (r = (l = buf) + fread(buf, 1, MAXN, stdin), l
== r) ? EOF : *l++;
    }
    template <typename _Tp>
    inline ios_in &operator>>(_Tp &x) {
        static char ch, sgn;
        for (sgn = 0, ch = gc(); !isdigit(ch); ch = gc()) {
            if (!~ch) return *this;
            sgn |= ch == '-';
        }
        for (x = 0; isdigit(ch); ch = gc())
            x = (x << 1) + (x << 3) + (ch ^ '0');
        sgn && (x = -x);
        return *this;
    }
} Cin;

ll a[maxn];

ll find_k(ll k, ll l, ll r) {
    if (l == r) return a[l];
    swap(a[l + r >> 1], a[l]);
    ll pos = l;  //选中间值为参考，卡最坏?
    for (ll i = l + 1; i <= r; ++i)
        if (a[i] < a[l]) swap(a[++pos], a[i]);
```

```
    swap(a[l], a[pos]);

    ll j = pos - l + 1;
    if (k == j) return a[pos];
    if (k > j) return find_k(k - j, pos + 1, r);
    return find_k(k, l, pos - 1);
}

void solve() {
    ll n;
    // cin >> n;
    Cin >> n;
    for (ll i = 1; i <= n; ++i) {
        ll x;
        Cin >> a[i] >> a[i];
        // cin >> a[i] >> a[i];
    }
    ll mid;
    ll k = n / 2 + 1;
    mid = find_k(k, 1, n);
    ll ans = 0;
    for (ll i = 1; i <= n; ++i) ans += abs(a[i] - mid);
    cout << ans << endl;
}

signed main() {
    ios::sync_with_stdio(false), cin.tie(0);
#ifdef LOCAL
    freopen("IO\\in.txt", "r", stdin);
    freopen("IO\\out.txt", "w", stdout);
    clock_t start, end;
    start = clock();
#endif
    solve();
#ifdef LOCAL
    end = clock();
    cout << endl
         << "Runtime: " << (double)(end - start) / CLOCKS_PER_SEC <<
"s\n";
#endif
    return 0;
}
```

四、运行输出结果：

提交时间	状态	分数	题目	编译器	耗时	用户
2021/09/29 19:24:21	答案正确	详情	编程题	C++ (g++)	221 ms	

五、调试和运行程序过程中产生的问题、采取的措施及获得的相关经验教训：

问题一：代码跑起来非常慢运行超时。

解决措施：发现起初在 n 为偶数的情况下寻找了两边中间位置的数然后取平均值，导致常数太大。

获得教训：进一步分析发现两者取任意一个均可。

问题二：自己加强数据测试发现在有序情况下确实运行效率差距明显。

解决措施：每次选定分界值可以利用随机函数或者取中间位置的数，这样加强代码的稳健性，防止特意卡最坏复杂度情况的数据。

获得教训：算法复杂度分析的重要性，发现了代码确定的病根才能更好的对症下药。

问题三：读入花费了大量时间。

解决措施：$C++$可以关闭同步流加快读入或者自定义函数以字符串形式读入数据后自行处理。

获得教训：C与$C++$语言的缓冲流概念区别。