实验报告

RSelect核心算法实现

算法流程

首先random数组中的一个pivot，之后对找到的pivot点进行一次partition，找到pivot点对应在数组中的位置，之后根据pivot点的具体位置pos与K的关系，递归调用RSelect(l, pos- 1)或RSelect(pos + 1, r)。RSelect期望时间复杂度为$O(n)$，最差情况为$O(n^2)$。

代码实现

```
int RSelect(vector<int>& w, int l, int r, int k) {
    while (l < r) {
        uniform_int_distribution<int> u(l, r);
        int len = Partition(w, l, r, u(e));
        if (len == k) return w[len];
        else if (len > k) r = len - 1;
        else l = len + 1;
    }
    return w[r];
}
```

DSelect核心算法实现

算法流程

DSelect对于RSelect的优化在于找到pivot点。首先找到找到相邻5个数的中间值组成一个数组mid，之后递归调用RSelect找到mid的中位数，即全局转数组的pivot点，之后利用找到的pivot点进行一次partition，找到pivot点对应在数组中的位置，之后根据pivot点的具体位置pos与K的关系，递归调用RSelect(l, pos- 1)或RSelect(pos + 1, r)。

时间复杂度递推式

$$T(n) = T(\frac{n}{5}) + T(\frac{7n}{10}) + O(n)$$

总时间复杂度为$O(n)$。

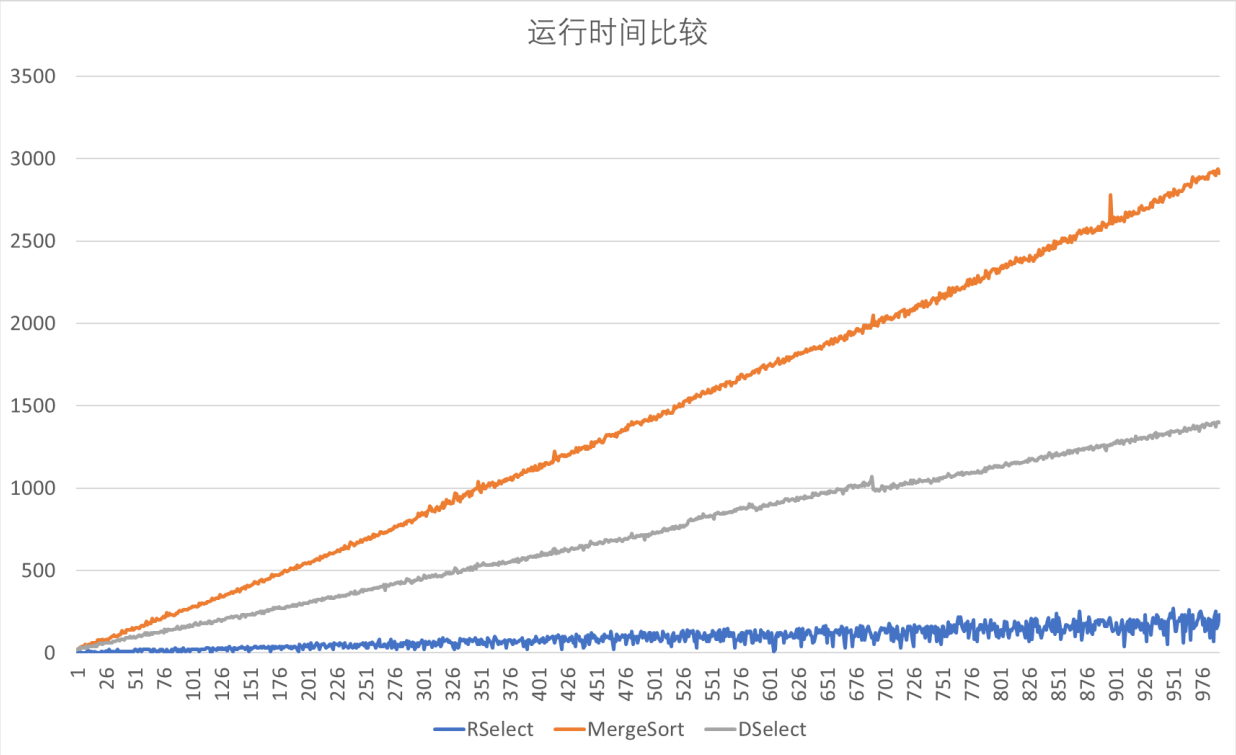
代码实现

```
int RSelect(vector<int>& w, int l, int r, int k) {
    if (r - l + 1 <= LEN) {
        sort(w.begin() + l, w.begin() + r + 1);
        return k;
    }
    vector<int> mid;
    mid.push_back(0);
```

```
    for (auto i = l; i < r; i += 5 ) mid.push_back(midSelect(w, i));
    int size = mid.size();
    int it = FSelect(mid, 1, size - 1, size >> 1), p = 0;
    for (int i = l; i <= r; i ++ )
        if (w[i] == mid[it])
            p = i;
    int len = Partition(w, l, r, p);
    if (len == k) return k;
    else if (len < k) return FSelect(w, len + 1, r, k);
    return FSelect(w, l, len - 1, k);
}
```

运行时间比较

我们比较了数据量在$1e6 - 1e8$的三种求第K大数的算法为KSelect, MergeSort. RSelect运行时间如下



YOJ通过截图

编号	题目名称	状态	分数	总时间	内存	代码	提交者	提交时间
#675360	#909 k小值	Accepted	100	2020 ms	15996 KB	cpp17 / 903 B	胡博文(2021201780)	2023/3/6 21:39

下边这个缩略输入输出，老师同意可以拉开看了^_^ (但考试时候不给看)

测试点 #0	2	得分：100	用时：427 ms	内存：6636 KiB
测试点 #1	2	得分：100	用时：649 ms	内存：11388 KiB
测试点 #2	2	得分：100	用时：944 ms	内存：15996 KiB

C++
GCC 8.2.0

C++ 11
GCC 8.2.0

C++ 17
GCC 8.2.0

C++ (NOI)
GCC 4.8.4 (NOILinux 1.4.1)

C++ 11 (NOI)
GCC 4.8.4 (NOILinux 1.4.1)

C++ 11 (Clang)
Clang 7.0.1

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <random>

using namespace std;

const int N = 4000010, len = 50;

int n, k;
int w[N];
default_random_engine e;

int Partition(int l, int r) {
    uniform_int_distribution<unsigned> u(l, r);
    int i = l, j = r;
    swap(w[i], w[u(e)]);
```