华东师范大学数据科学与工程学院上机实践报告

课程名称：算法设计与分析	**年级：**19 级	**上机实践成绩：**
指导教师：金澈清	**姓名：**龚敬洋	
上机实践名称：随机选择算法	**学号：**	**上机实践日期：**
	10195501436	2020/10/23
上机实践编号：No.5	**组号：**1-436	

一、目的

1．熟悉算法设计的基本思想
2．掌握随机选择算法（rand select）的方法
3．掌握选择算法（SELECT）的方法

二、内容与设计思想

1. 编写随机整数生成算法，生成 S 到 T 范围内的 N 个随机整数并输出；
2. 编写随机选择算法和 SELECT 算法；
3. 随机生成 1e2、1e3、1e4、1e5、1e6 个数，使用随机选择算法和 SELECT 算法找到第 0.5N 大的数输出，并画图描述不同情况下的运行时间差异；
4. 随机生成 1e6 个数，使用随机选择算法和 SELECT 算法找到第 0.2N、0.4N、0.6N、0.8N 大的数输出，并画图描述不同情况下的运行时间差异；

三、使用环境

推荐使用 C/C++集成编译环境。

四、 实验过程

写出随机选择算法的源代码

```
#include<iostream>
#include<fstream>
#include<cstdlib>
#include<ctime>
using namespace std;
int a[1000005];
void swap(int *m, int *n){
    int tmp;
    tmp = *m;
    *m = *n;
    *n = tmp;
}
int partition(int p, int r){
    int x = a[r];
    int i = p-1;
    for(int j = p;j < r; j++){
        if(a[j] <= x){
            i = i + 1;
            swap(&a[i], &a[j]);
```

```
        }
    }
    swap(&a[i+1], &a[r]);
    return i + 1;
}
int random_partition(int p, int r){
    int t = p + rand() % (r - p + 1);
    swap(&a[t], &a[r]);
    return partition(p, r);
}
int random_select(int p, int r, int i){
    if(p == r) return a[p];
    int pivot = random_partition(p, r);
    int k = pivot - p + 1;
    if(i == k) return a[pivot];
    else if(i < k) return random_select(p, pivot - 1, i);
    else return random_select(pivot + 1, r, i - k);
}
int main(){
    srand(time(0));
    ifstream fin("data.txt");
    int n, i;
    n = 0;
    while(!fin.eof()){
        fin>>a[n];
        n++;
    }
    n--;
    cin>>i;
    cout<<random_select(0, n - 1, i)<<endl;
    fin.close();
    return 0;
}
```

写出 SELECT 算法的源代码

```
#include<iostream>
#include<fstream>
#include<cstdlib>
#include<ctime>
using namespace std;
int a[1000005];
struct I2D{
    int index;
    int value;
};
void swap(int *m, int *n){
    int tmp;
    tmp = *m;
    *m = *n;
    *n = tmp;
}
int partition(int p, int r, int pivot){
    int x = a[pivot];
    int i = p-1;
    for(int j = p;j <= r; j++){
        if(a[j] <= x){
            i = i + 1;
            swap(&a[i], &a[j]);
        }
    }
    swap(&a[i], &a[pivot]);
    return i;
}
```

```
29. void insert_sort(int l, int r){
30.     int key, j;
31.     for(int i = l; i <= r; i++){
32.         key = a[i];
33.         j = i - 1;
34.         while(j >= l && a[j] > key){
35.             a[j + 1] = a[j];
36.             j--;
37.         }
38.         a[j + 1] = key;
39.     }
40. }
41. I2D select(int p, int r, int i){
42.     if(r - p < 140){
43.         insert_sort(p, r);
44.         I2D pack;
45.         pack.index = p + i - 1;
46.         pack.value = a[p + i - 1];
47.         return pack;
48.     }
49.     for(int u = 0; u <= (r - p) / 5; u++){
50.         if(p + u * 5 + 4 > r){
51.             insert_sort(p + u * 5, r);
52.             int mid = p + 5 * u + (r - (p + 5 * u)) / 2;
53.             swap(&a[mid], &a[p + u]);
54.         }
55.         else{
56.             insert_sort(p + u * 5, p + u * 5 + 4);
57.             int mid = p + 5 * u + 2;
58.             swap(&a[mid], &a[p + u]);
59.         }
60.     }
61.     I2D tmpPack = select(p, p + (r - p) / 5, ((r - p) / 5 + 1) / 2);
62.     int tmp = tmpPack.index;
63.     int pivot = partition(p, r, tmp);
64.     int k = pivot - p + 1;
65.     if(i == k) {
66.         I2D subPack;
67.         subPack.index = pivot;
68.         subPack.value = a[pivot];
69.         return subPack;
70.     }
71.     else if(i < k) return select(p, pivot - 1, i);
72.     else return select(pivot + 1, r, i - k);
73. }
74. int main(){
75.     srand(time(0));
76.     ifstream fin("data.txt");
77.     int n, i;
78.     n = 0;
79.     while(!fin.eof()){
80.         fin>>a[n];
81.         n++;
82.     }
83.     n--;
84.     cin>>i;
85.     I2D result = select(0, n-1, i);
86.     cout<<result.value<<endl;
87.     fin.close();
88.     return 0;
89. }
```

截取各个实验的实验结果

算法：rand-select 数据规模：1e2

```
50
447
Total Time: 5.5e-05
```

算法：select 数据规模：1e2

```
50
447
Total Time: 0.000107
```

算法：rand-select 数据规模：1e3

```
500
5248
Total Time: 0.000157
```

算法：select 数据规模：1e3

```
500
5248
Total Time: 0.000288
```

算法：rand-select 数据规模：1e4

```
5000
49965
Total Time: 0.000415
```

算法：select 数据规模：1e4

```
5000
49965
Total Time: 0.001664
```

算法：rand-select 数据规模：1e5

```
50000
500319
Total Time: 0.004772
```

算法：select 数据规模：1e5

```
50000
500319
Total Time: 0.01191
```

算法：rand-select 数据规模：1e6

```
500000
4993029
Total Time: 0.037822
```

算法：select 数据规模：1e6

```
500000
4993029
Total Time: 0.066902
```

算法：rand-select 数据规模：1e6

```
200000
1991866
Total Time: 0.029736
```

```
400000
3996939
Total Time: 0.042312
```

```
600000
5991505
Total Time: 0.027423
```

```
800000
7996541
Total Time: 0.028792
```

算法：select 数据规模：1e6

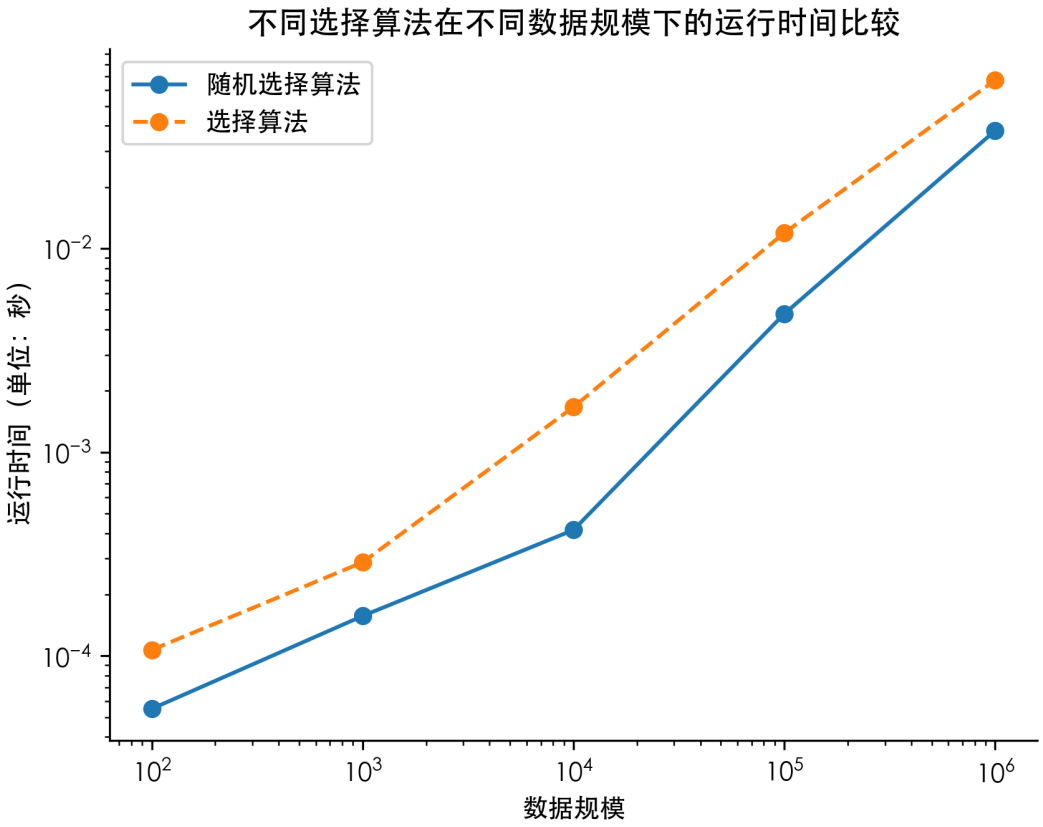
```
200000
1991866
Total Time: 0.070884
```

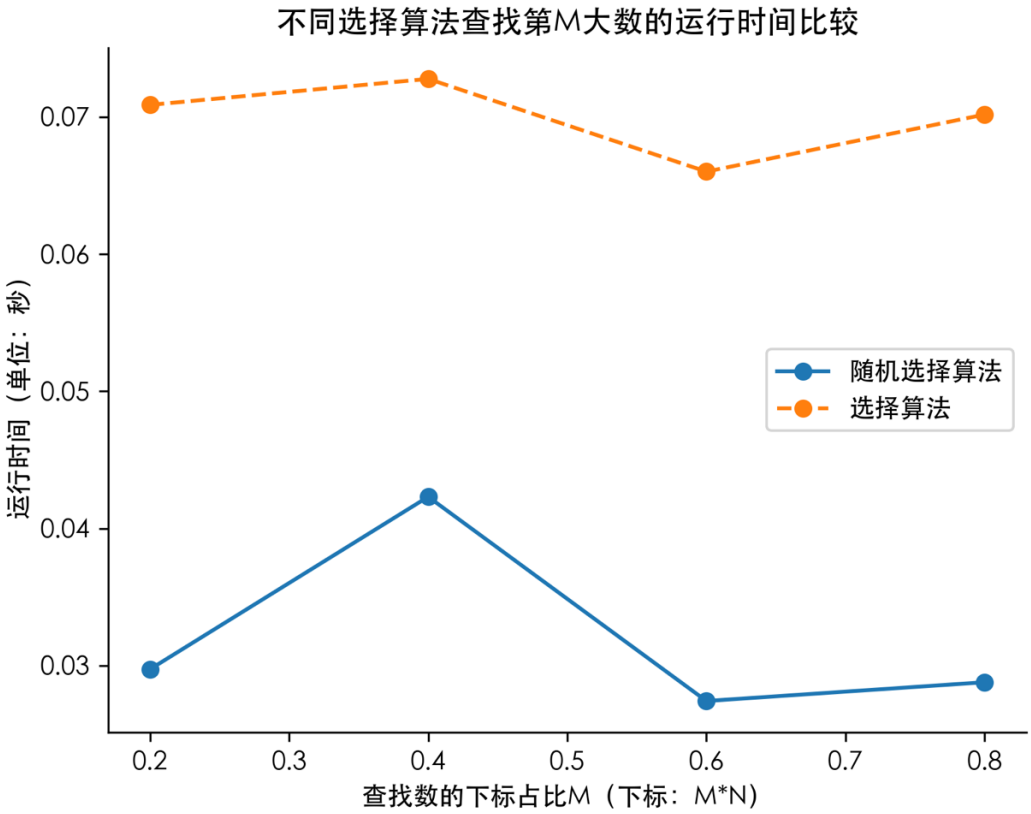
```
400000
3996939
Total Time: 0.07278
```

```
600000
5991505
Total Time: 0.066018
```

```
800000
7996541
Total Time: 0.070183
```

分别画出各个实验结果的折线图


不同选择算法在不同数据规模下的运行时间比较
随机选择算法
选择算法
运行时间（单位：秒）
10^{-2}
10^{-3}
10^{-4}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
数据规模


不同选择算法查找第M大数的运行时间比较
运行时间（单位：秒）
0.07
0.06
0.05
0.04
0.03
随机选择算法
选择算法
0.2
0.3
0.4
0.5
0.6
0.7
0.8
查找数的下标占比M（下标：M*N）

五、总结

对上机实践结果进行分析，问题回答，上机的心得体会及改进意见。

从理论上看，随机选择算法（Rand-Select）的平均时间复杂度为$O(n)$，而在最坏情况下会降为$\Theta(n^2)$；而选择算法（Select）在最坏情况下的时间复杂度也为$O(n)$，优于随机选择算法。然而在实际运行时，随机选择算法直接通过随机数确定了 pivot，而选择算法需要通过插入排序并递归取中位数的方式找出区间最优的 pivot，对数据的预处理耗时较大，故选择算法的实际运行效率反而不如随机选择算法，这也与实验结果相吻合。