Homework 7: Median

Part 1: 实验步骤

1. 设计实验用例，你可以从以下几个角度设计几组对照数据进行实验：

   a) 数据规模：设计至少三组不同规模的数据集，比较数据集大小不同时，两者算法的差异；

   b) 乱序性：比较数据集顺序和乱序的情况下，两者算法的差异；（可以自行设计一些容易导致性能最坏

   情况出现的数据集）

2. 在 Linear Select 算法中，需要将数据分组，每组大小固定为 |Q|，请自行设计实验，探究 |Q| 对此算

法性能的影响

Part 2: 实验与结果分析

1. 算法分析：

1. quickSelect基本思路是选择一个随机主元素，然后将数组划分为比主元素小的左子数组、等于主元素的中间子数组和比主元素大的右子数组。接着根据 k 和左、中、右子数组的长度关系，选择递归查找左子数组或右子数组，或者直接返回中间子数组中的元素。
2. linearSelect基本思路是将原始数组划分为若干个大小为 5 的子数组，然后计算每个子数组的中位数，并在这些中位数中选择一个作为主元素。根据主元素将原始数组划分为比主元素小的左子数组、等于主元素的中间子数组和比主元素大的右子数组。接着根据 k 和左、中、右子数组的长度关系，选择递归查找左子数组或右子数组，或者直接返回中间子数组中的元素。

2. 结果分析

1. 数据规模

```
数据规模为： 500
The 250th smallest element in quickSelect is 2433
quickSelect time: 39
The 250th smallest element in linearSelect is 2433
linearSelect time: 213

Process finished with exit code 0
```

```
数据规模为： 1000
The 500th smallest element in quickSelect is 4449
quickSelect time: 82
The 500th smallest element in linearSelect is 4449
linearSelect time: 371

Process finished with exit code 0
```

数据规模为： 2000
The 1000th smallest element in quickSelect is 8336
quickSelect time: 103
The 1000th smallest element in linearSelect is 8336
linearSelect time: 720

Process finished with exit code 0

数据规模为： 5000
The 2500th smallest element in quickSelect is 16676
quickSelect time: 253
The 2500th smallest element in linearSelect is 16676
linearSelect time: 1728

Process finished with exit code 0

数据规模为： 10000
The 5000th smallest element in quickSelect is 16413
quickSelect time: 443
The 5000th smallest element in linearSelect is 16413
linearSelect time: 3471

Process finished with exit code 0

数据规模为： 20000
The 10000th smallest element in quickSelect is 16441
quickSelect time: 890
The 10000th smallest element in linearSelect is 16441
linearSelect time: 5325

Process finished with exit code 0

数据规模为： 50000
The 25000th smallest element in quickSelect is 16421
quickSelect time: 2807
The 25000th smallest element in linearSelect is 16421
linearSelect time: 15852

Process finished with exit code 0

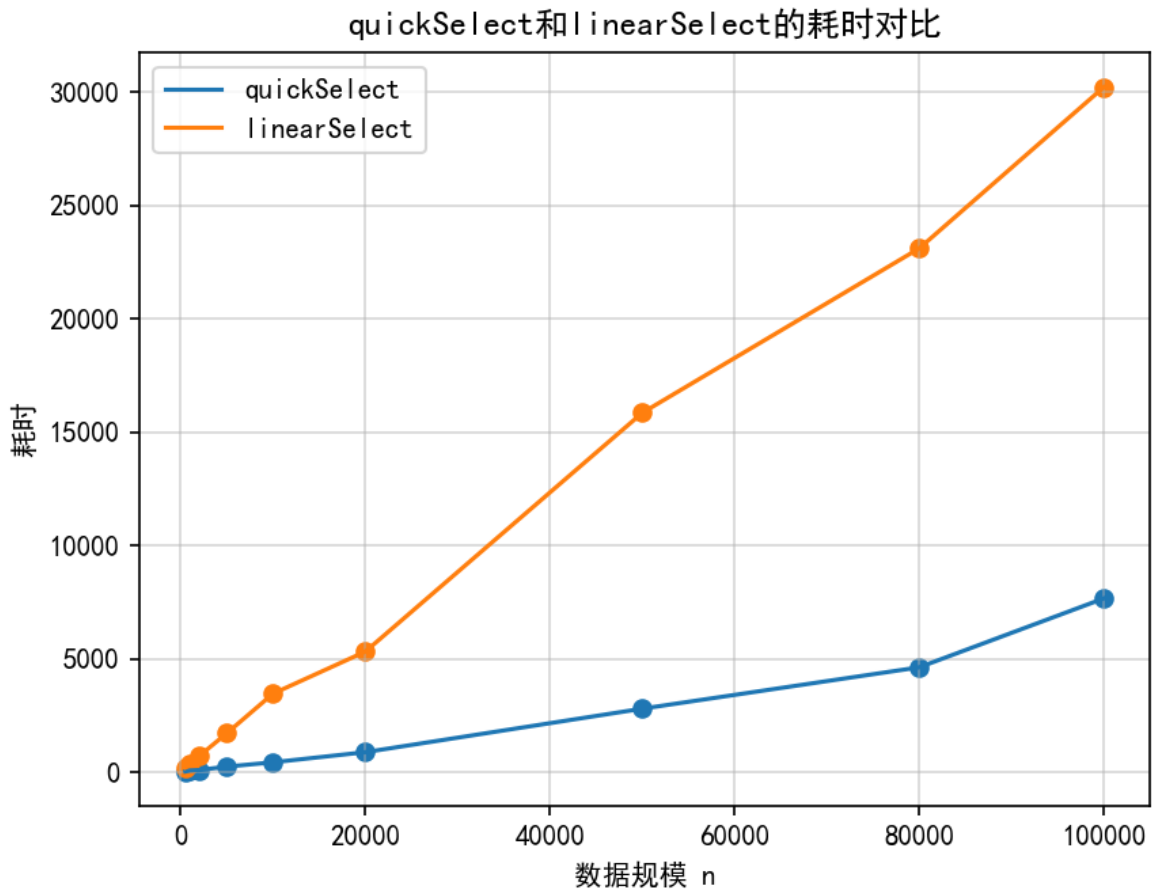
数据规模为： 80000
The 40000th smallest element in quickSelect is 16247
quickSelect time: 4631
The 40000th smallest element in linearSelect is 16247
linearSelect time: 23104

Process finished with exit code 0

数据规模为： 100000
The 50000th smallest element in quickSelect is 16336
quickSelect time: 7669
The 50000th smallest element in linearSelect is 16336
linearSelect time: 30213

Process finished with exit code 0

作图分析：



2. 乱序性

运行截图：

```
D:\clionlabs\hm7\cmake-build-debug\hm7.exe
1625 1300 2687 452 3904 1124 205 3939 3197 1329 1704 1259 4103 523 1
8 11 17 23 42 44 45 56 62 69 77 85 88 96 115 124 139 154 157 166 167
数据规模为： 500
The 250th smallest element in quickSelect(乱序) is 2205
quickSelect time: 47
The 250th smallest element in quickSelect(有序) is 2205
quickSelect time: 32
The 250th smallest element in linearSelect(乱序) is 2205
linearSelect time: 218
The 250th smallest element in linearSelect(有序) is 2205
linearSelect time: 161

Process finished with exit code 0
```

```
6723 1072 9324 1455 8429 622 5275 4425 7025 3495 2267 7581 141
3 12 25 35 39 49 50 62 64 72 98 135 141 142 146 149 161 165 177
数据规模为： 1000
The 500th smallest element in quickSelect(乱序) is 4560
quickSelect time: 70
The 500th smallest element in quickSelect(有序) is 4560
quickSelect time: 44
The 500th smallest element in linearSelect(乱序) is 4560
linearSelect time: 388
The 500th smallest element in linearSelect(有序) is 4560
linearSelect time: 310

Process finished with exit code 0
```

```
16818 5096 13097 1163 12640 3425 17232 7838 4341 11809 17977 1
17 27 32 36 40 44 51 58 68 90 95 103 113 115 118 145 157 168 1
数据规模为： 2000
The 1000th smallest element in quickSelect(乱序) is 8543
quickSelect time: 163
The 1000th smallest element in quickSelect(有序) is 8543
quickSelect time: 77
The 1000th smallest element in linearSelect(乱序) is 8543
linearSelect time: 595
The 1000th smallest element in linearSelect(有序) is 8543
linearSelect time: 588

Process finished with exit code 0
```

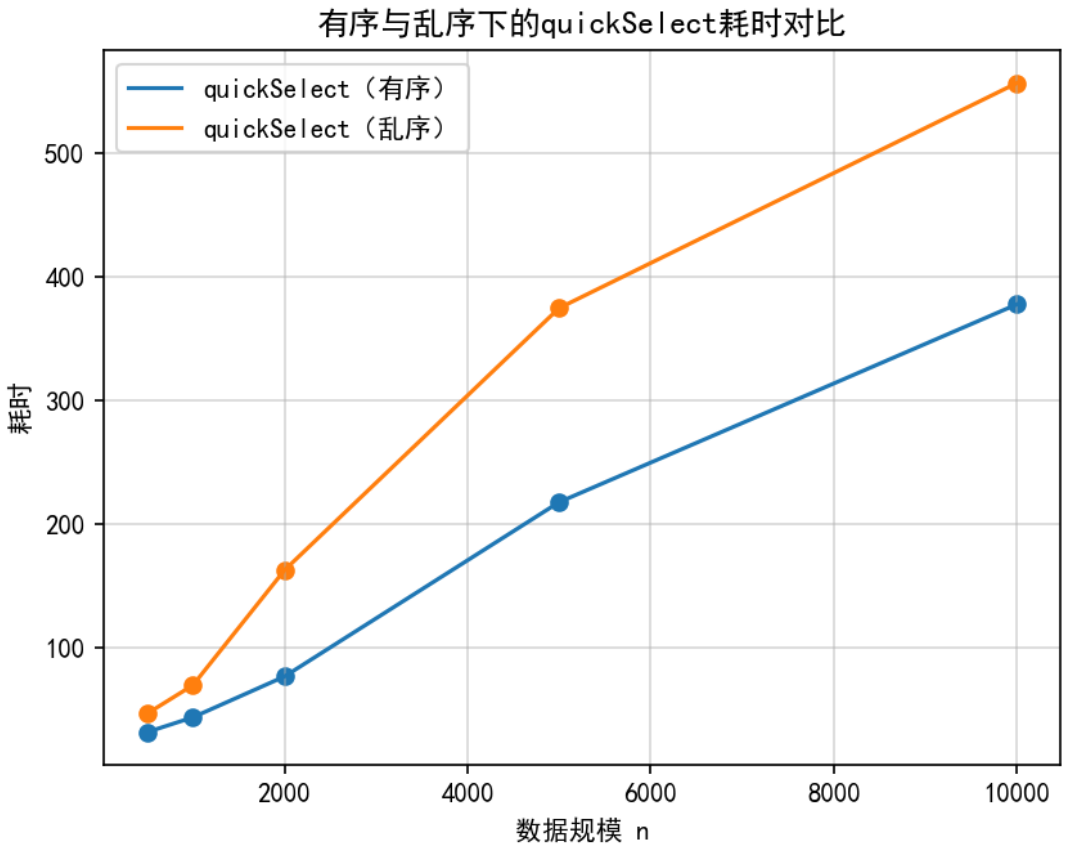
```
16877 1959 6972 28319 21694 8570 7720 7915 6637 19447 15315 1
8 16 45 47 58 75 77 79 82 86 103 104 121 146 148 167 171 172
数据规模为： 5000
The 2500th smallest element in quickSelect(乱序) is 16646
quickSelect time: 375
The 2500th smallest element in quickSelect(有序) is 16646
quickSelect time: 218
The 2500th smallest element in linearSelect(乱序) is 16646
linearSelect time: 2557
The 2500th smallest element in linearSelect(有序) is 16646
linearSelect time: 2319

Process finished with exit code 0
```

```
17037 4343 30349 27773 2821 4372 29158 28318 29273 13104 8068 2180
4 7 8 9 10 11 13 16 25 26 36 39 40 41 43 78 85 100 106 111 115 116
数据规模为： 10000
The 5000th smallest element in quickSelect(乱序) is 16206
quickSelect time: 557
The 5000th smallest element in quickSelect(有序) is 16206
quickSelect time: 378
The 5000th smallest element in linearSelect(乱序) is 16206
linearSelect time: 5402
The 5000th smallest element in linearSelect(有序) is 16206
linearSelect time: 2524

Process finished with exit code 0
```

作图分析：



由于**选取主元素一直是随机的**，因此有序与无序并不会影响太大，否则应该出现quickSelect（O（n^2））明显差于linearSelect，最差的情况出现的概率极小（quickSelect有序情况下选取最大或最小元素为主元素）

- 最坏情况下的性能理论上甚至比 Linear Select 还要差：选取主元素为arr[0]，且有序：（会发现quickSelect性能很差）

```
数据规模为： 1000
The 500th smallest element in quickSelect is 4247
quickSelect time: 5023
The 500th smallest element in linearSelect is 4247
linearSelect time: 377

Process finished with exit code 0
```

```
数据规模为： 10000
The 5000th smallest element in quickSelect is 16532
quickSelect time: 470521
The 5000th smallest element in linearSelect is 16532
linearSelect time: 2804

Process finished with exit code 0
```

3. |Q|的选取

```
20273 6414 6255 19413 948 29137 637 10973 8259 15217 25333 3
数据规模为： 10000
The 5000th smallest element in linearSelect(乱序) is 16281
|Q| = 5
linearSelect time: 3048

Process finished with exit code 0
```

```
D:\clionlabs\hm7\cmake-build-debug\hm7.exe
20384 11411 23813 18366 23963 31574 8157 9749 12597 2791 2030
数据规模为： 10000
The 5000th smallest element in linearSelect(乱序) is 16346
|Q| = 10
linearSelect time: 2090

Process finished with exit code 0
```

```
D:\clionlabs\hm7\cmake-build-debug\hm7.exe
20528 25589 23404 28576 19053 7742 14032 12021 18211 27188 1
数据规模为： 10000
The 5000th smallest element in linearSelect(乱序) is 16362
|Q| = 15
linearSelect time: 1834

Process finished with exit code 0
```

```
D:\clionlabs\hm7\cmake-build-debug\hm7.exe
20612 10136 29121 31630 25088 11533 29420 1448 21529 31178 9
数据规模为： 10000
The 5000th smallest element in linearSelect(乱序) is 16258
|Q| = 20
linearSelect time: 1872

Process finished with exit code 0
```

```
20691 5954 31877 19325 10493 7471 5815 29498 24591 14696 6401 5
数据规模为： 10000
The 5000th smallest element in linearSelect(乱序) is 16138
|Q| = 25
linearSelect time: 1872

Process finished with exit code 0
```

```
D:\clionlabs\hm7\cmake-build-debug\hm7.exe
20834 20131 31468 29535 5583 16407 11691 31770 30205 6325 32661 74
数据规模为： 10000
The 5000th smallest element in linearSelect(乱序) is 16313
|Q| = 30
linearSelect time: 1860

Process finished with exit code 0
```

```
20772 12520 19729 31084 6214 7335 18090 21852 11397 24835 2703 6662 25
数据规模为： 10000
The 5000th smallest element in linearSelect(乱序) is 16580
|Q| = 40
linearSelect time: 1939

Process finished with exit code 0
```

在所选范围内，n=10000时，|Q|选15最好。|Q|的选择要依赖于n。

结果分析：

1. quickSelect在最坏情况下，时间复杂度为 O(n^2)，但是在平均情况下，时间复杂度为 O(n)。这个算法的一个重要优化是随机化主元素的选择，避免出现最坏情况，即数组已经是有序的或者近似有序的情况。在实践中，可以通过多次随机选择主元素来减少出现最坏情况的概率。
2. linearSelect在最坏情况下，时间复杂度为 O(n)，但是在平均情况下，时间复杂度为 O(n)。和 Quick Select 算法类似，这个算法也可以通过多次随机选择子数组中的元素来减少出现最坏情况的概率。需要注意的是，在选择子数组的大小时，不能选择太小，否则计算中位数时可能会出现数组越界的情况，也不能选择太大，否则递归层数会太多，降低算法效率。通常使用大小为 5 左右的子数组效果较好.

从实验结果可以看出，当数据集是有序的时候，quickSelect 的性能要好于 Linear Select，但是当数据集是乱序的时候，quickSelect 的性能会出现波动，最坏情况下的性能甚至比 Linear Select 还要差。

在实际应用中，为了避免出现最坏情况，可以使用随机化的方式来提高 quickSelect 的性能，即在选择主元素时随机选取，而不是固定选择第一个元素或者最后一个元素。