算法测试报告

通过对多种算法及数据结构进行高压测试，我们能够选择更适合的算法来实现我们的模块，并能够发现算法中的潜在问题。本报告详细介绍了测试环境、测试数据、测试方法及各算法的性能比较。

测试环境

- **编程语言**：Java
- **硬件配置**：Windows11+Intel 12700H
- **测试工具**：自定义 Timer 类记录时间，List用于存储数据

测试数据集

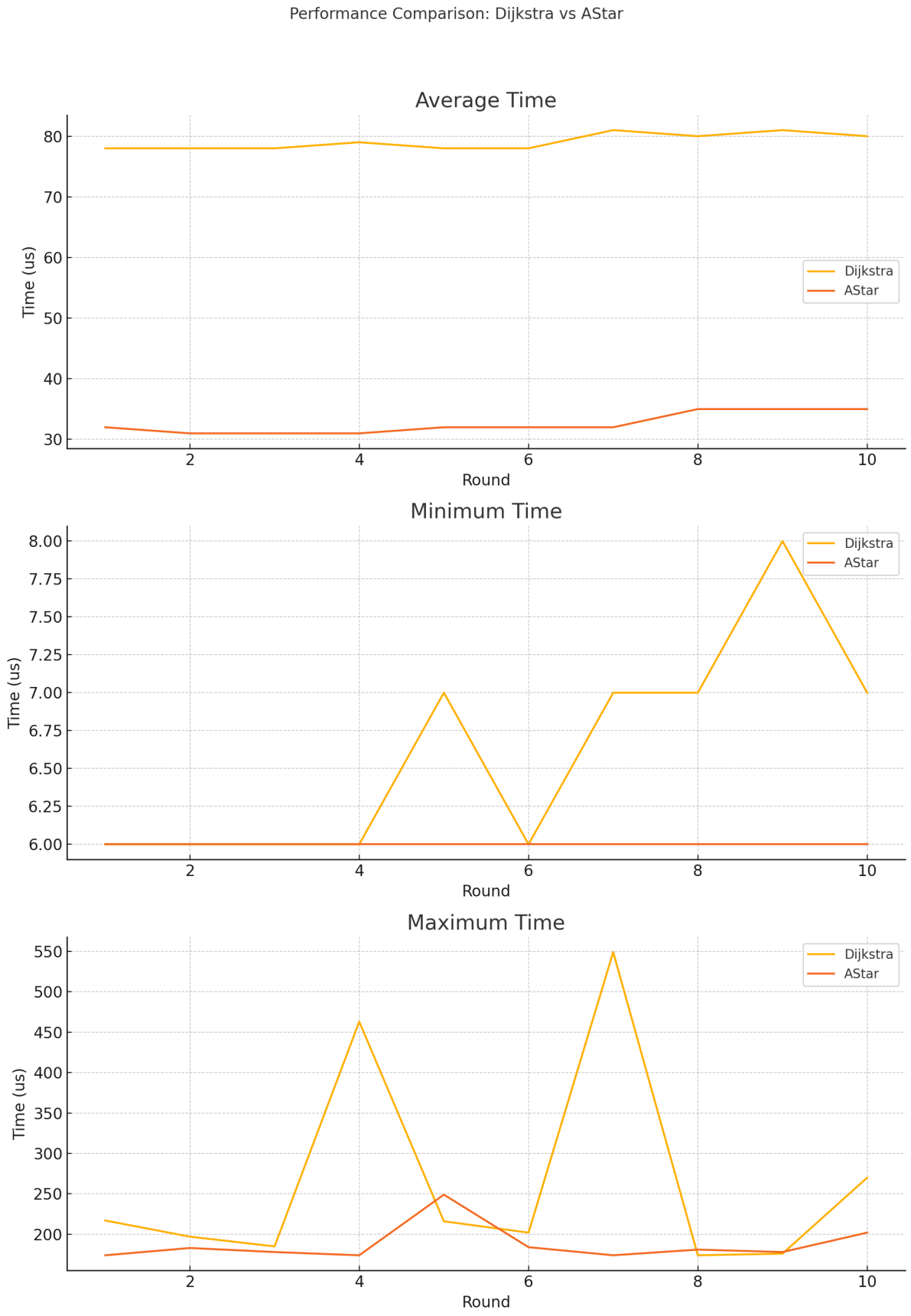
测试数据集是随机生成的，包含1000个不同的List。

测试方法

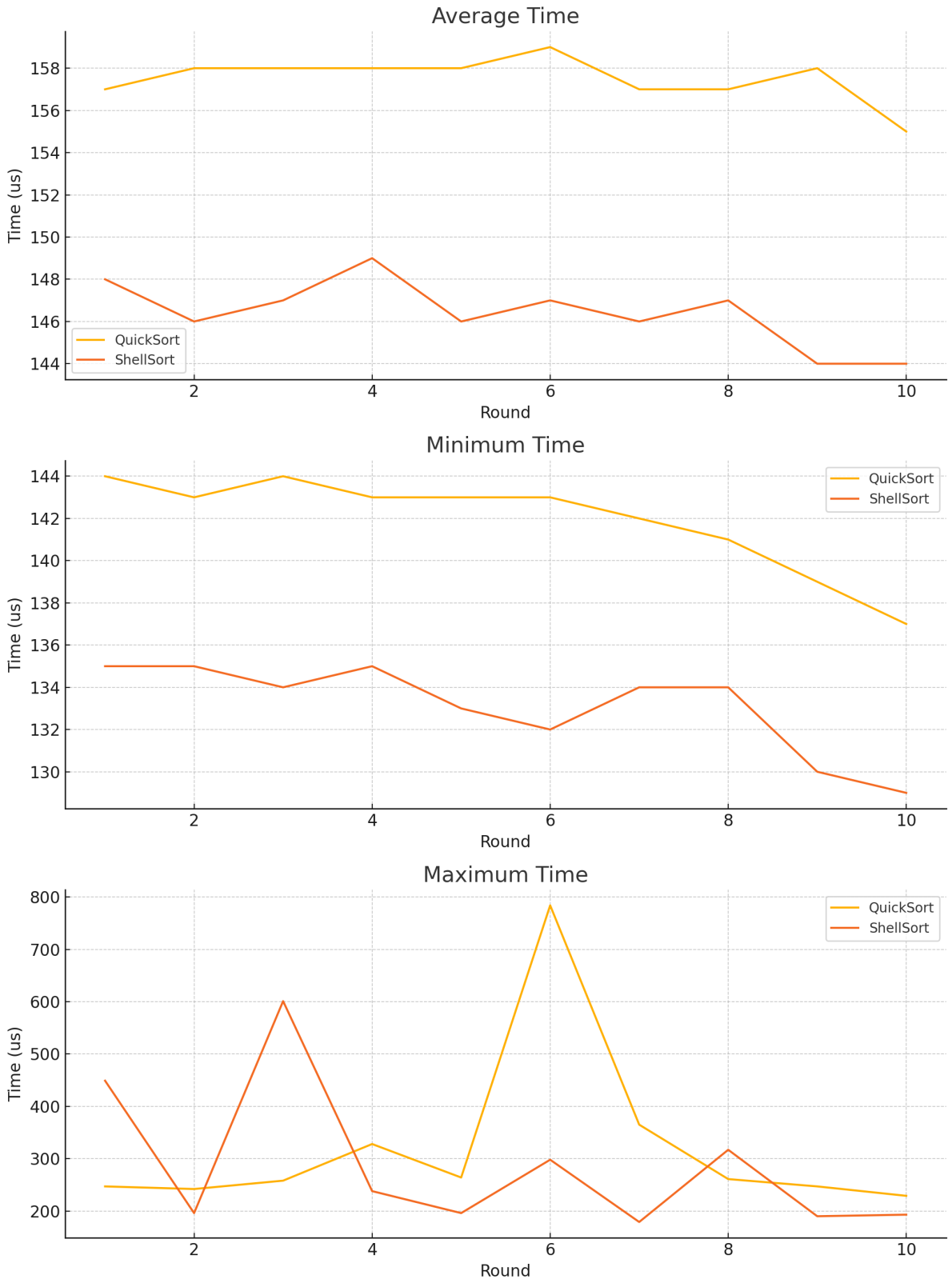
每轮测试包括以下步骤：

1. 生成或加载测试数据集。
2. 对算法进行多轮（10轮）测试，每轮记录总时间、最短时间和最长时间。
3. 对比多种算法的平均时间、最短时间和最长时间。

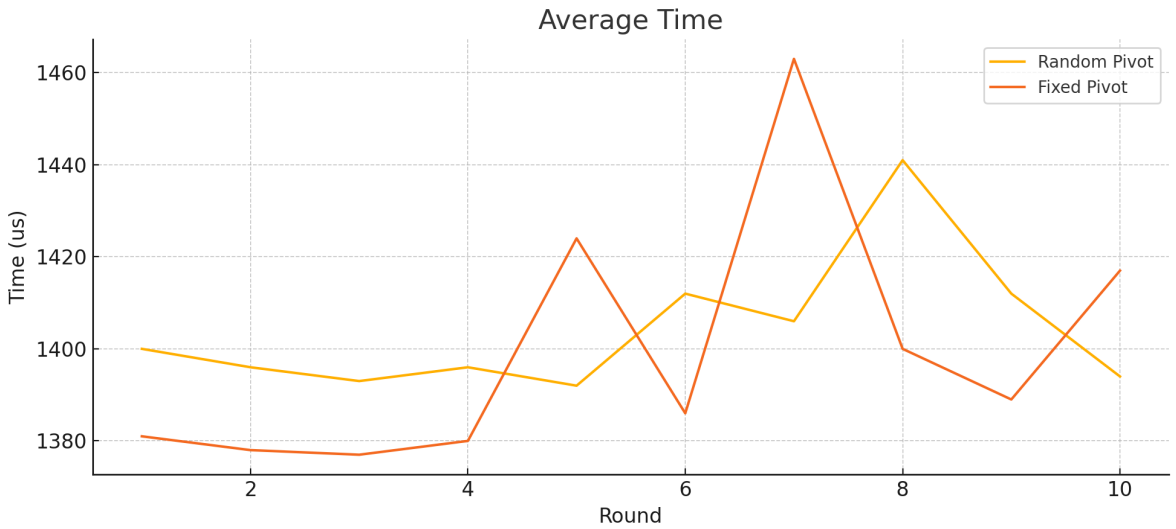
图表对比

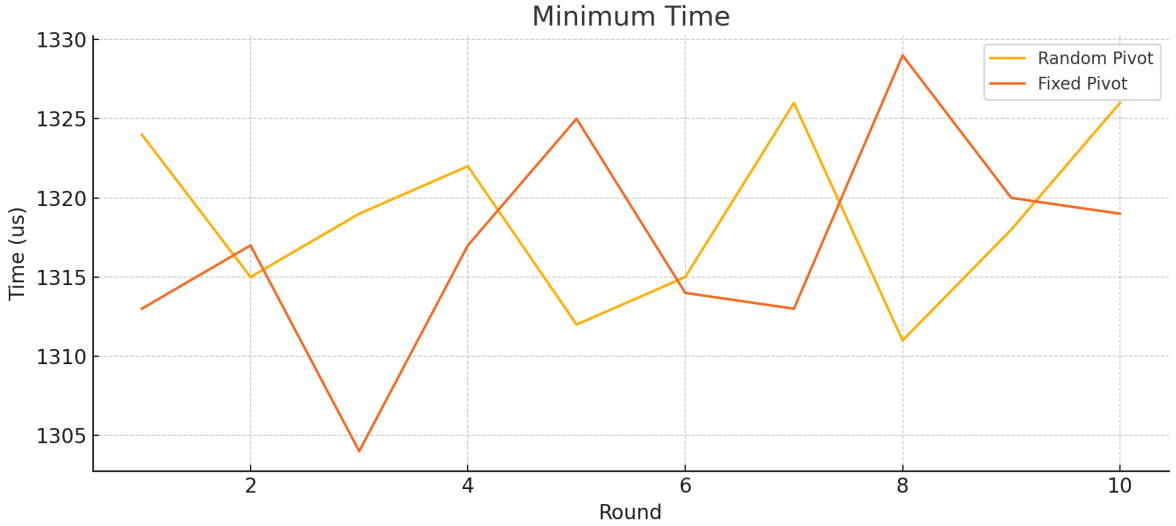

Performance Comparison: Dijkstra vs AStar
Average Time
Time (us)
Round
Dijkstra
AStar
Minimum Time
Time (us)
Round
Dijkstra
AStar
Maximum Time
Time (us)
Round
Dijkstra
AStar

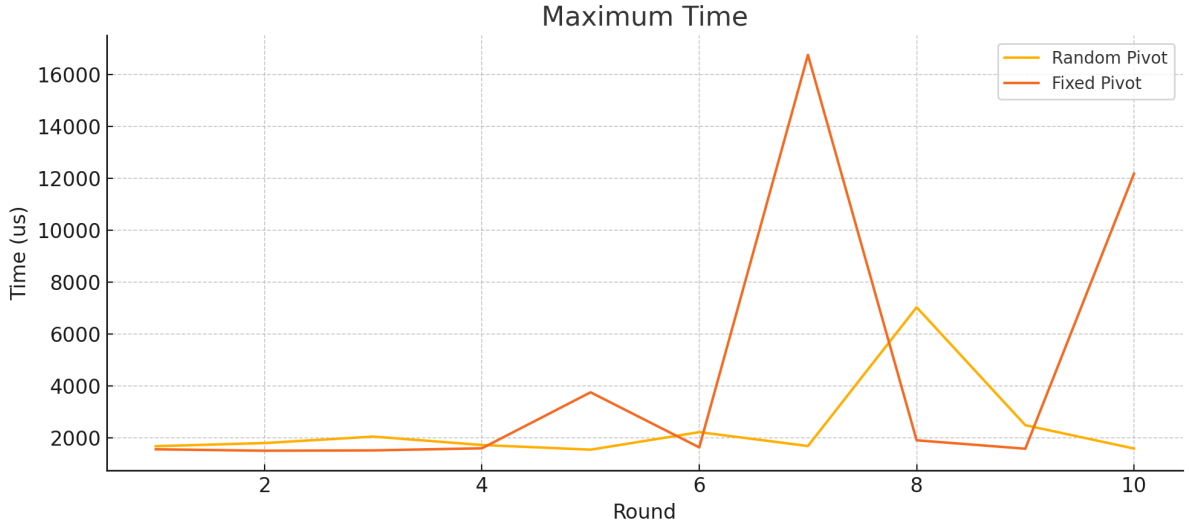
Performance Comparison: QuickSort vs ShellSort



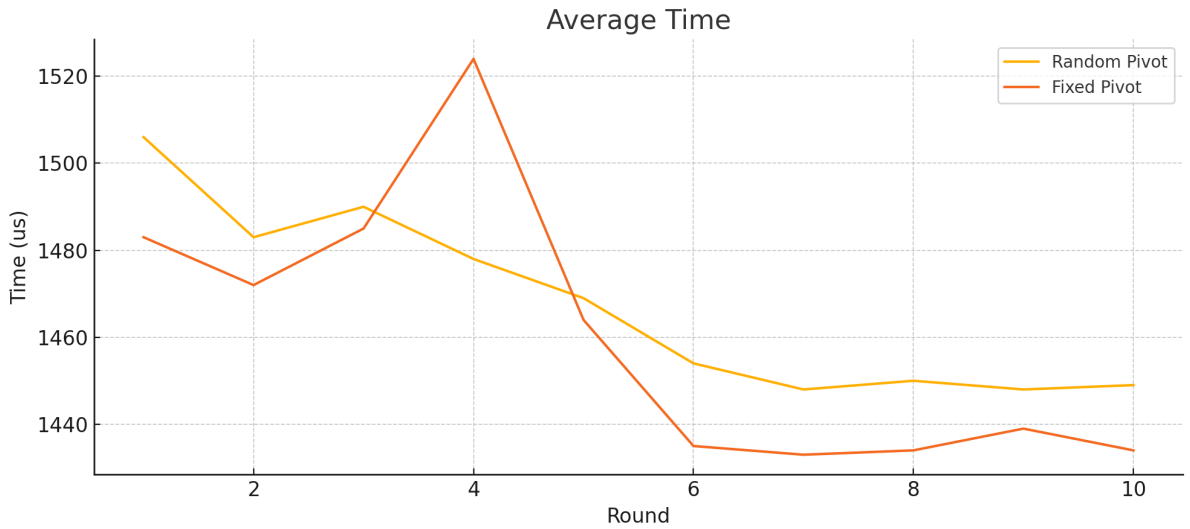
Performance Comparison: QuickSort (Random Pivot vs Fixed Pivot) on Ordered Array

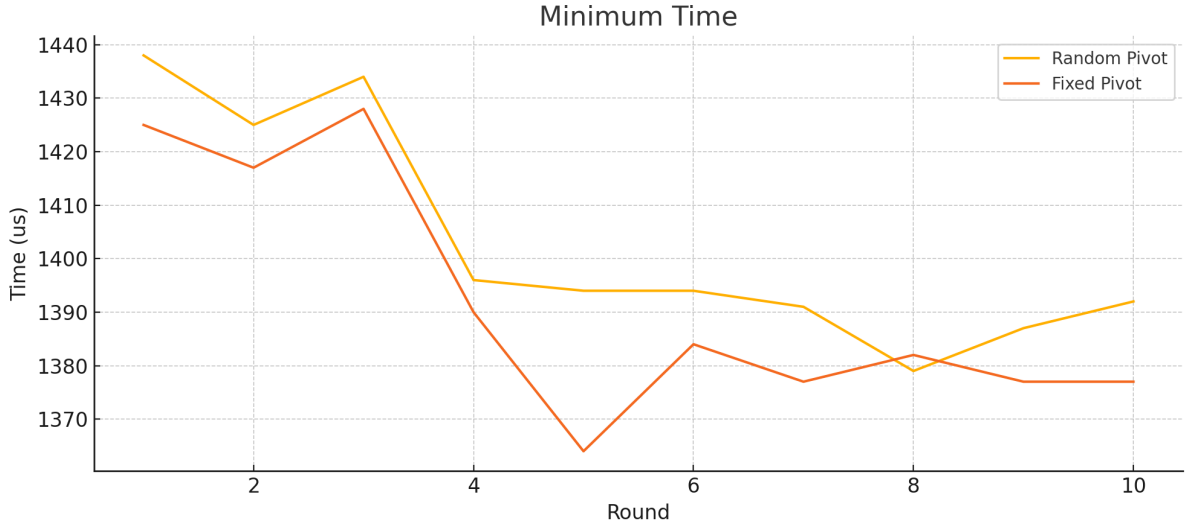


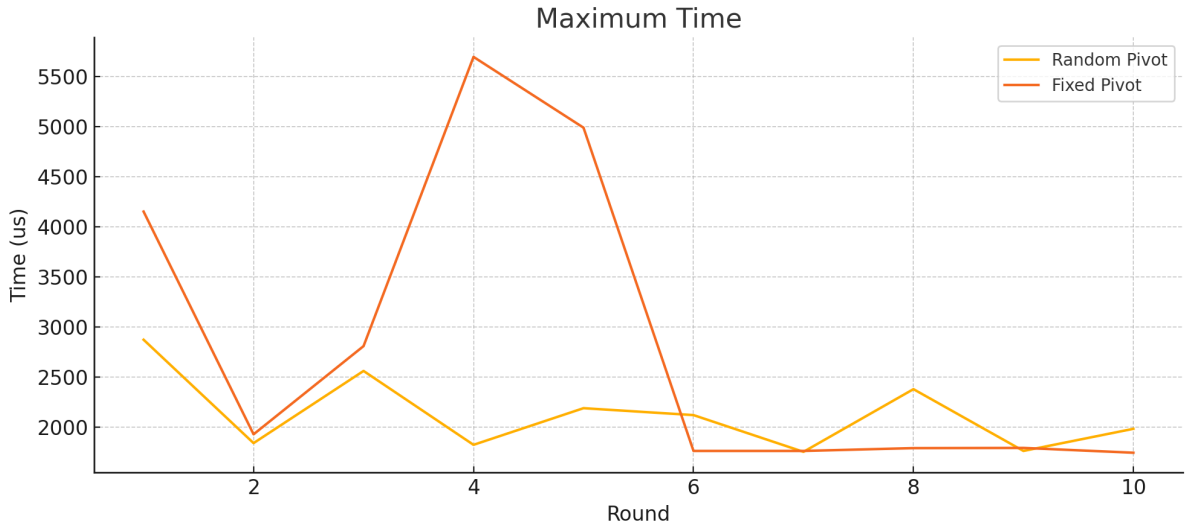




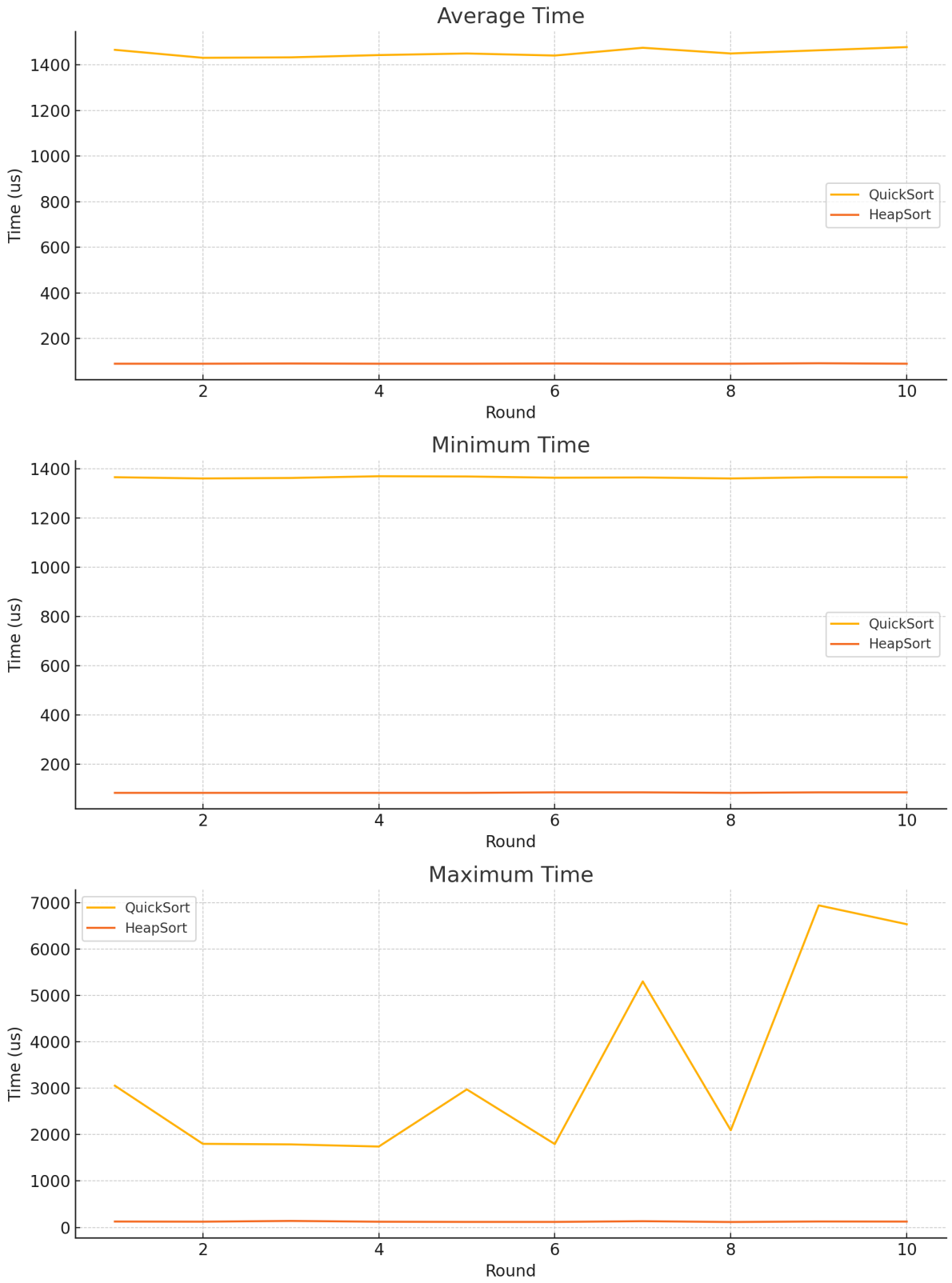
Performance Comparison: QuickSort (Random Pivot vs Fixed Pivot) on Unordered Array



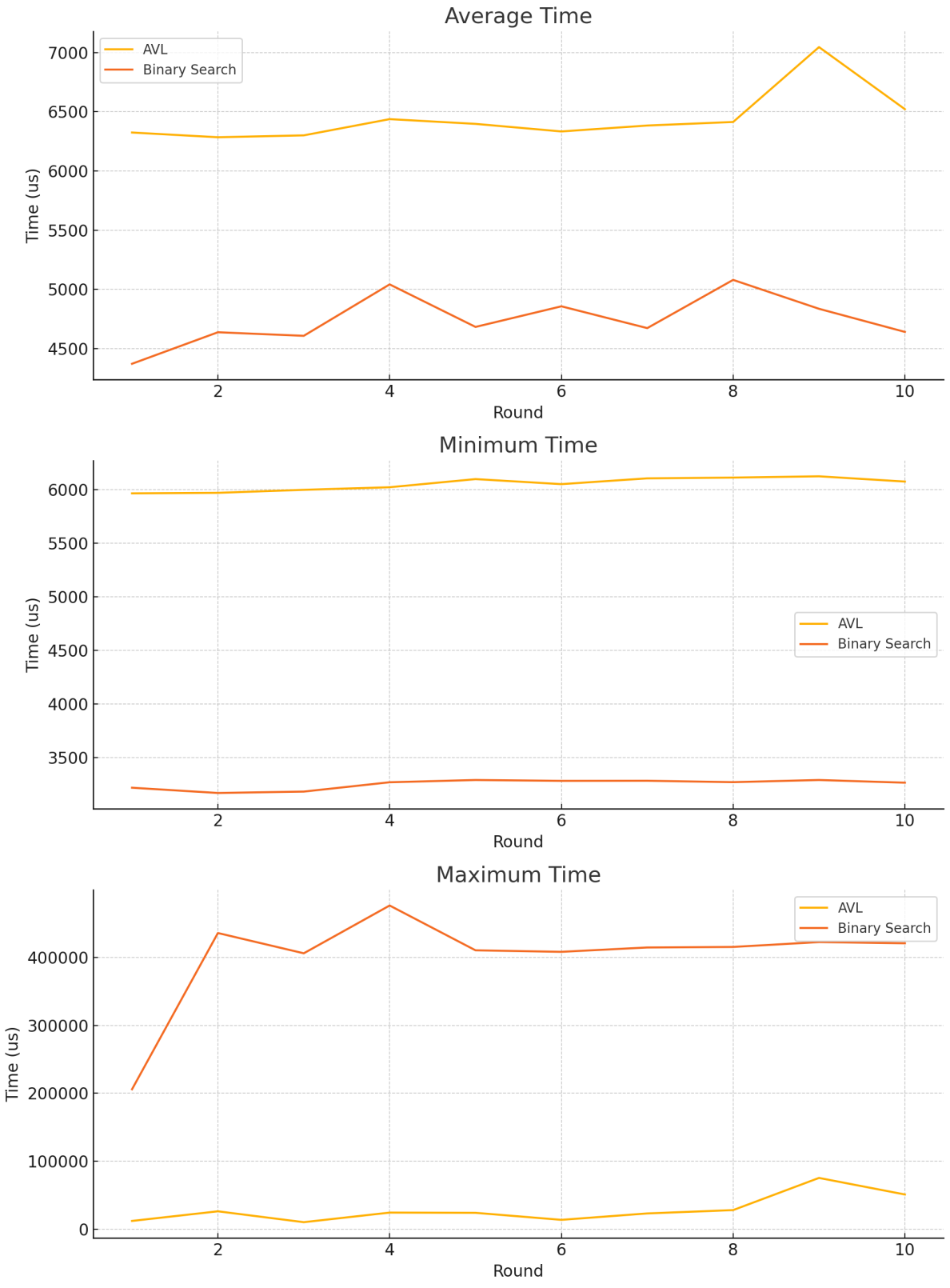


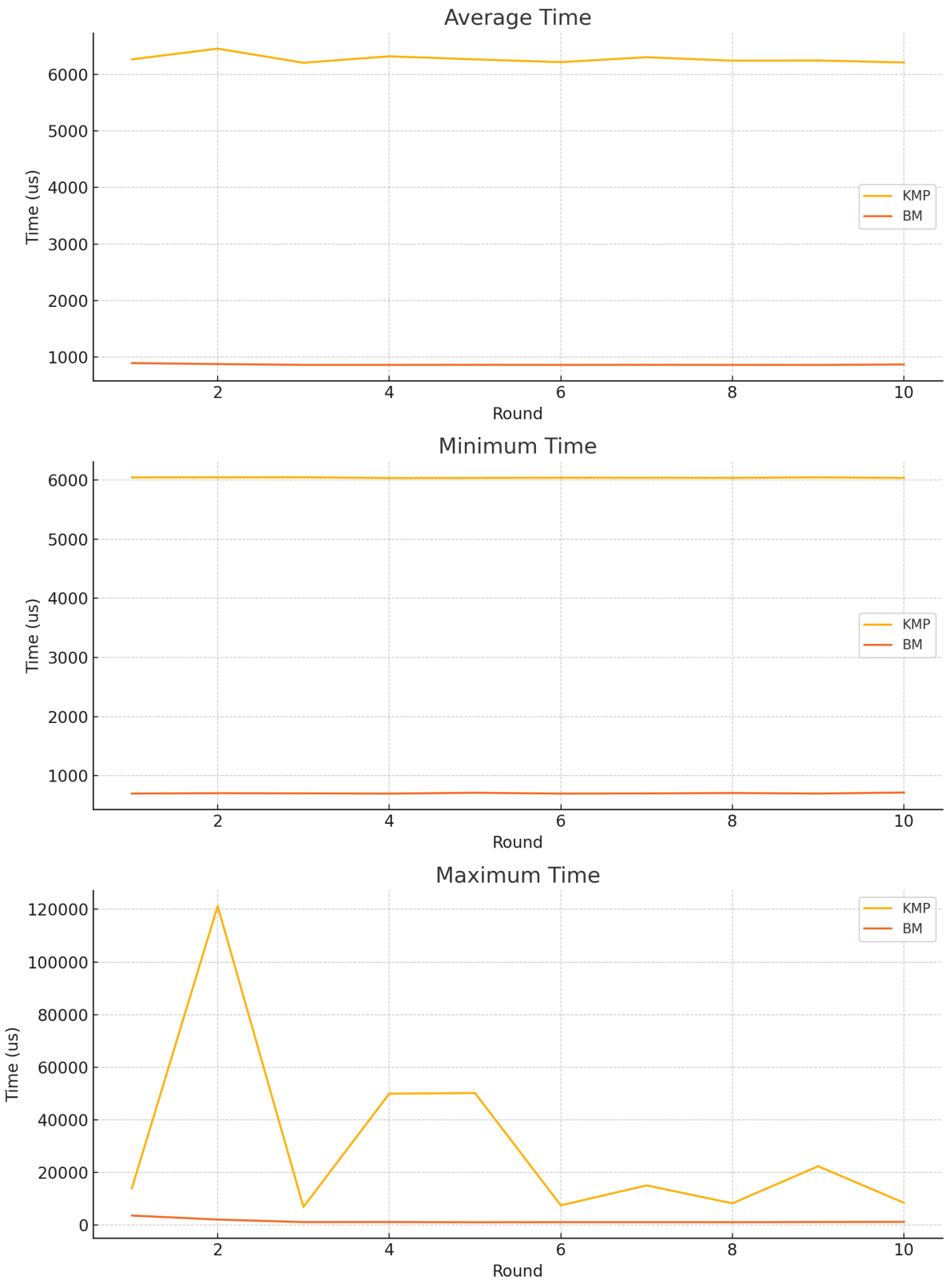


Performance Comparison: QuickSort vs HeapSort (Top 10 elements)



Performance Comparison: AVL vs Binary Search (Insertion + Deletion + Search)





测试结果

1. AStar 和 Dijkstra 性能比较

Dijkstra 性能记录（us）

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	78	6	217

Round	AveTime (us)	MinTime (us)	MaxTime (us)
2	78	6	197
3	78	6	185
4	79	6	463
5	78	7	216
6	78	6	202
7	81	7	549
8	80	7	174
9	81	8	176
10	80	7	270

AStar 性能记录（us）

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	32	6	174
2	31	6	183
3	31	6	178
4	31	6	174
5	32	6	249
6	32	6	184
7	32	6	174
8	35	6	181
9	35	6	178
10	35	6	202

总结：AStar 算法在所有测试轮次中的平均时间、最短时间和最长时间均优于 Dijkstra 算法。

2. ShellSort 和 QuickSort 性能比较

QuickSort 性能记录（us）

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	157	144	247
2	158	143	242
3	158	144	258
4	158	143	328

Round	AveTime (us)	MinTime (us)	MaxTime (us)
5	158	143	264
6	159	143	784
7	157	142	365
8	157	141	261
9	158	139	247
10	155	137	229

ShellSort 性能记录（us）

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	148	135	449
2	146	135	196
3	147	134	601
4	149	135	238
5	146	133	196
6	147	132	298
7	146	134	179
8	147	134	317
9	144	130	190
10	144	129	193

总结：ShellSort 在所有测试轮次中的平均时间和最短时间均优于 QuickSort，但 QuickSort 的最大时间较为稳定。

3. QuickSort 是否带随机 pivot 性能比较

数组元素有序:

随机 pivot

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	1400	1324	1675
2	1396	1315	1796
3	1393	1319	2045
4	1396	1322	1721
5	1392	1312	1541

Round	AveTime (us)	MinTime (us)	MaxTime (us)
6	1412	1315	2217
7	1406	1326	1682
8	1441	1311	7031
9	1412	1318	2485
10	1394	1326	1584

固定 pivot ((l + r) / 2)

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	1381	1313	1555
2	1378	1317	1500
3	1377	1304	1513
4	1380	1317	1594
5	1424	1325	3752
6	1386	1314	1629
7	1463	1313	16757
8	1400	1329	1902
9	1389	1320	1578
10	1417	1319	12182

总结：在有序数据上，随机 pivot 和固定 pivot 表现相近，但固定 pivot 有时会出现极端高的最大时间。

数组元素无序：

随机 pivot

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	1506	1438	2872
2	1483	1425	1839
3	1490	1434	2560
4	1478	1396	1822
5	1469	1394	2188
6	1454	1394	2119
7	1448	1391	1752
8	1450	1379	2378

Round	AveTime (us)	MinTime (us)	MaxTime (us)
9	1448	1387	1761
10	1449	1392	1982

固定 pivot ((l + r) / 2)

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	1483	1425	4152
2	1472	1417	1927
3	1485	1428	2808
4	1524	1390	5697
5	1464	1364	4990
6	1435	1384	1761
7	1433	1377	1761
8	1434	1382	1789
9	1439	1377	1791
10	1434	1377	1743

总结：在无序数据上，随机 pivot 和固定 pivot 表现相近，但随机 pivot 更加稳定。

4. QuickSort 和 HeapSort 性能比较（取前10个元素）

QuickSort 性能记录（us）

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	1466	1365	3053
2	1431	1360	1800
3	1433	1362	1788
4	1443	1369	1742
5	1450	1368	2973
6	1441	1363	1795
7	1475	1364	5300
8	1450	1360	2093
9	1464	1365	6938
10	1478	1365	6532

HeapSort 性能记录（us）

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	89	84	126
2	89	84	123
3	90	84	139
4	89	84	122
5	89	84	118
6	90	86	118
7	89	86	133
8	89	84	116
9	91	86	127
10	89	86	125

总结：HeapSort 在处理前10个元素时，性能明显优于 QuickSort。

5. AVL和 Binary Search 性能比较（插入+删除+查找）

插入基本有序，随机查找：

AVL

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	6324	5964	11940
2	6284	5969	26113
3	6300	5997	10087
4	6437	6021	24212
5	6397	6097	23884
6	6333	6050	13532
7	6383	6104	22984
8	6413	6111	27900
9	7045	6123	75281
10	6521	6074	50890

Binary Search

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	4372	3219	205848

Round	AveTime (us)	MinTime (us)	MaxTime (us)
2	4638	3170	435955
3	4608	3183	406089
4	5042	3270	476464
5	4683	3291	410481
6	4857	3283	408258
7	4673	3284	414660
8	5080	3271	415461
9	4836	3291	422624
10	4641	3266	420870

总结：在插入基本有序的数据时，AVL 树的表现优于 Binary Search，但 Binary Search 在某些测试轮次中出现了极高的最大时间。

6. KMP 和 BM 性能比较

KMP

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	6268	6044	14033
2	6457	6045	121267
3	6207	6046	6873
4	6321	6034	49934
5	6267	6036	50169
6	6218	6039	7498
7	6305	6038	15079
8	6244	6037	8260
9	6247	6045	22368
10	6212	6036	8459

BM

Round	AveTime (us)	MinTime (us)	MaxTime (us)
1	891	699	3617
2	874	705	2098
3	860	702	1127
4	860	698	1145

Round	AveTime (us)	MinTime (us)	MaxTime (us)
5	861	712	1063
6	860	698	1098
7	861	701	1108
8	860	708	1087
9	859	699	1150
10	867	716	1224

总结：BM 算法在所有测试轮次中的平均时间、最短时间和最长时间均优于 KMP 算法。

结论

通过本次测试，我们得出以下结论：

1. **AStar 优于 Dijkstra**：AStar 算法在路径查找上的性能明显优于 Dijkstra 算法。
2. **ShellSort 和 QuickSort**：在处理一般排序任务时，ShellSort 的性能优于 QuickSort，特别是其最短时间表现。
3. **QuickSort Pivot 选择**：在有序数据上，随机 pivot 和固定 pivot 的表现相近，但固定 pivot 有时会出现极端高的最大时间；在无序数据上，随机 pivot 更加稳定。
4. **HeapSort 优于 QuickSort**：在处理前10个元素的任务时，HeapSort 的性能明显优于 QuickSort。
5. **AVL和 Binary Search**：在插入基本有序的数据时，AVL树的表现优于 Binary Search，但 Binary Search 在某些测试轮次中出现了极高的最大时间。
6. **BM 优于 KMP**：在字符串匹配上，BM 算法在所有测试轮次中的性能均优于 KMP 算法。