COS285 Data Structures
Alex Bard & Devin Chamberlain
Assignment 9

Link to GitHub Repository:
https://github.com/AlexBard122/Assignment9
Link to Google Spreadsheet:
https://docs.google.com/spreadsheets/d/1n4qekHYXJmu3GkvgwS4i9mMmWi17qicBIIBrLRJDfzY/edit#gid=0

Task 1:

Table of Average Time (in milliseconds) Taken for Different Partitioning Approaches:

State:	Reports:	First	Last	Median of 3	Median of 10	Mid	Random
AL:	40866	55.6549	82.67844	97.114	274.04454	61.02	157.62
AR:	14601	5.94872	22.72516	16.178	27.26188	5.2328	22.494
CA:	640961	757.73002	895.89784	987.32	33923.97276	474.28	4341.2
CO:	30808	16.7406	17.1231	20.192	104.39906	9.2626	28.29
CT:	34330	18.9172	24.50458	24.768	120.69148	11.2802	32.644
DC:	10901	4.10012	5.20728	5.3224	16.16622	2.5838	5.9058
DE:	6664	2.2324	2.60396	3.0572	7.36406	1.3396	2.6644
FL:	455488	525.94288	528.14802	539.58	17959.71416	291.88	2569.2
ID:	5412	2.53668	2.86746	2.6442	5.453	1.6702	3.2548
IL:	50657	26.87578	34.65888	35.888	252.04882	16.044	72.346
IN:	25176	12.52604	18.17634	15.544	68.9616	7.4638	23.27
KS:	9422	3.55328	5.98398	4.5764	12.56102	1.8908	5.0698
KY:	6922	2.33924	4.30994	3.2412	7.7068	1.4588	3.1228
LA:	62083	43.97026	51.82208	47.776	370.91766	22.638	88.716
MA:	17423	8.25774	7.95612	8.3888	35.29198	3.9294	11.556
MD:	51299	33.15716	34.91596	36.864	253.85216	18.952	76.53
ME:	350	0.09838	0.0874	0.114	0.13554	0.0586	0.0878
MI:	48921	31.20916	31.02962	37.516	237.83672	17.08	74.95
MN:	69987	51.20154	57.29282	60.792	467.79566	28.04	127.08
MS:	6173	2.63654	2.64	3.7076	6.34938	1.4902	3.526
MT:	20642	12.03084	15.18006	16.172	49.74394	7.0146	18.396
NC:	127237	121.71208	115.95386	118.46	1505.23912	59.404	326.32
ND:	2037	0.85272	0.69082	1.0388	1.3071	0.3768	0.7994
NE:	3257	1.28128	1.20426	1.3472	2.5488	0.7054	1.3134
NH:	1591	0.95996	0.477	0.5816	0.91244	0.2544	0.521

NJ:	60102	44.1792	41.01276	46.438	361.0472	23.036	95.186
NM:	3477	1.12784	1.09638	1.815	2.79738	0.6846	1.2966
NV:	7910	3.50136	2.87934	4.139	27.7271	1.7774	3.7122
NY:	140411	123.46108	117.28762	140.32	1813.1664	72.398	354.76
OH:	38899	22.50668	21.32248	26.7	161.84122	11.816	47.458
OR:	57757	44.54874	41.40362	46.768	321.6277	23.5	88.122
RI:	3727	1.82732	1.36354	1.7168	2.96388	0.624	1.441
SD:	17	0.00788	0.00582	0.0074	0.00866	0.0042	0.0072
TN:	73196	54.12044	50.61708	55.266	521.81456	29.02	128.78
TX:	183079	167.85856	167.2033	171.48	3184.00024	87.294	563.04
VA:	147414	118.0291	121.9306	175.92	1924.26684	74.452	414.82
VT:	139	0.03884	0.03316	0.0452	0.0514	0.0266	0.051
WA:	30129	17.29346	20.45936	22.538	98.98438	9.0106	36.234
WV:	8103	3.51026	3.25646	4.7688	10.11562	1.9036	4.789
WY:	2432	0.82032	0.91386	1.3052	1.71896	0.6746	0.9712

The table above shows the average time in milliseconds for each partitioning approach to sort the reports in a given state (average time is calculated across 5 separate tests). Out of all the given approaches, partitioning based on the middle value yields significantly shorter processing times.

Task 2:

Table of Average Time (in milliseconds) Taken for Old and New Mid Partitioning Approaches:

State:	Reports:	New Mid	Old Mid
AL:	40866	119.0816	61.02
AR:	14601	20.54124	5.2328
CA:	640961	787.02118	474.28
CO:	30808	14.2457	9.2626
CT:	34330	16.75178	11.2802
DC:	10901	4.58046	2.5838
DE:	6664	2.62708	1.3396
FL:	455488	470.16974	291.88
ID:	5412	2.58156	1.6702
IL:	50657	39.44566	16.044

IN:	25176	13.47214	7.4638
KS:	9422	3.95718	1.8908
KY:	6922	2.43448	1.4588
LA:	62083	44.73016	22.638
MA:	17423	6.56798	3.9294
MD:	51299	32.25926	18.952
ME:	350	0.07952	0.0586
MI:	48921	29.84392	17.08
MN:	69987	52.17352	28.04
MS:	6173	2.34526	1.4902
MT:	20642	9.86948	7.0146
NC:	127237	104.1095	59.404
ND:	2037	0.6775	0.3768
NE:	3257	1.263	0.7054
NH:	1591	0.44744	0.2544
NJ:	60102	43.4315	23.036
NM:	3477	1.34634	0.6846
NV:	7910	3.5424	1.7774
NY:	140411	123.01474	72.398
OH:	38899	34.48196	11.816
OR:	57757	36.82756	23.5
RI:	3727	1.3506	0.624
SD:	17	0.00792	0.0042
TN:	73196	49.20098	29.02
TX:	183079	164.47972	87.294
VA:	147414	132.72322	74.452
VT:	139	0.02824	0.0266
WA:	30129	16.36918	9.0106
WV:	8103	3.16476	1.9036
WY:	2432	0.8252	0.6746

In order to reduce overhead with recursive calls, we modified our method to use insertion sort when the input was smaller than 8 reports. Although we expected this to reduce the sorting time, this change actually resulted in the method taking roughly twice as much time to sort the same number of reports as the old method.