Program Structures & Algorithms

Spring 2022

Assignment 4

Name: Kanishk Bimalkumar Bhatia

NUID: 001580259

- Task:

1. A cutoff (defaults to, say, 1000) which you will update according to the first argument in the command line when running. It's your job to experiment and come up with a good value for this cutoff. If there are fewer elements to sort than the cutoff, then you should use the system sort instead.
2. Recursion depth or the number of available threads. Using this determination, you might decide on an ideal number (t) of separate threads (stick to powers of 2) and arrange for that number of partitions to be parallelized (by preventing recursion after the depth of lg t is reached).
3. An appropriate combination of these.

- Evidence:

  To find out the optimal value for cutoff at which we need to switch from sort parallelly to using system sort. Looking at the observations below, we can see how the number of threads, or the height of parallelism would affect the performance of this algorithm.

  Experiment 1:

  For this, consider an array of fixed size, and height and vary the cut off value between 1% and 49% of the array size. Repeat this for different levels of heights and sizes of arrays.

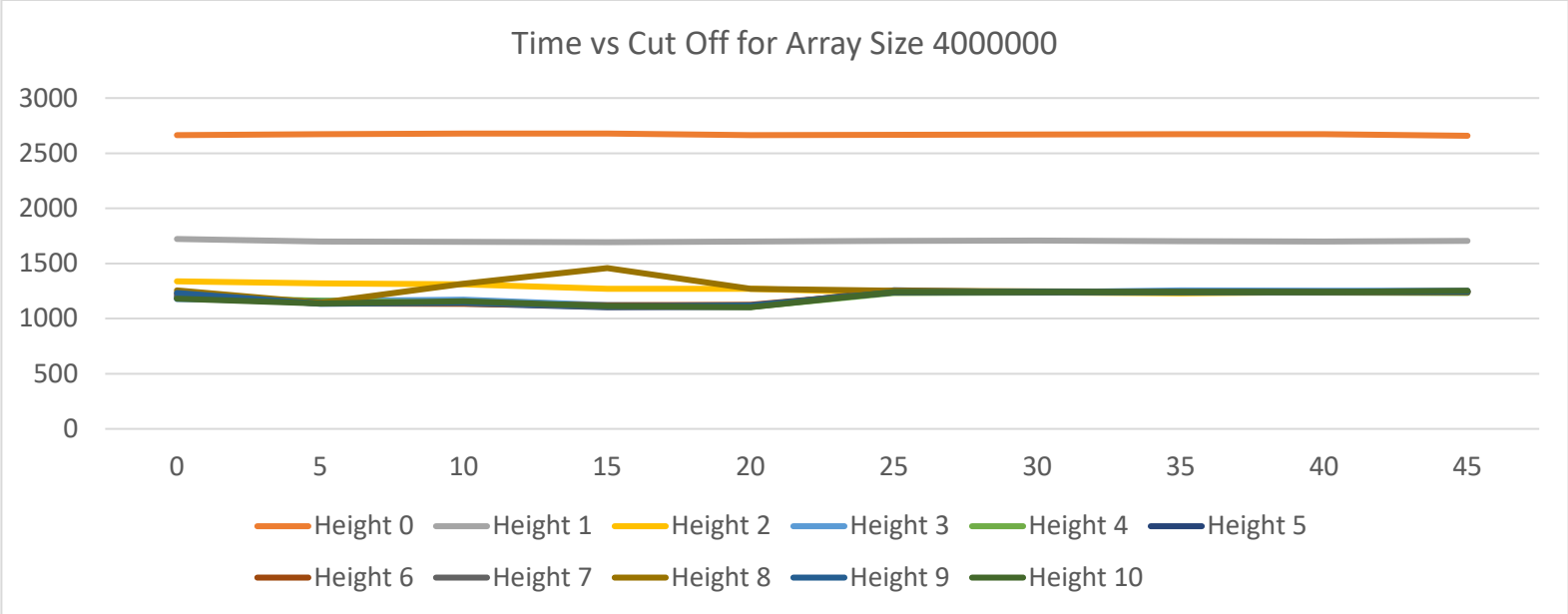
Experiment 2:

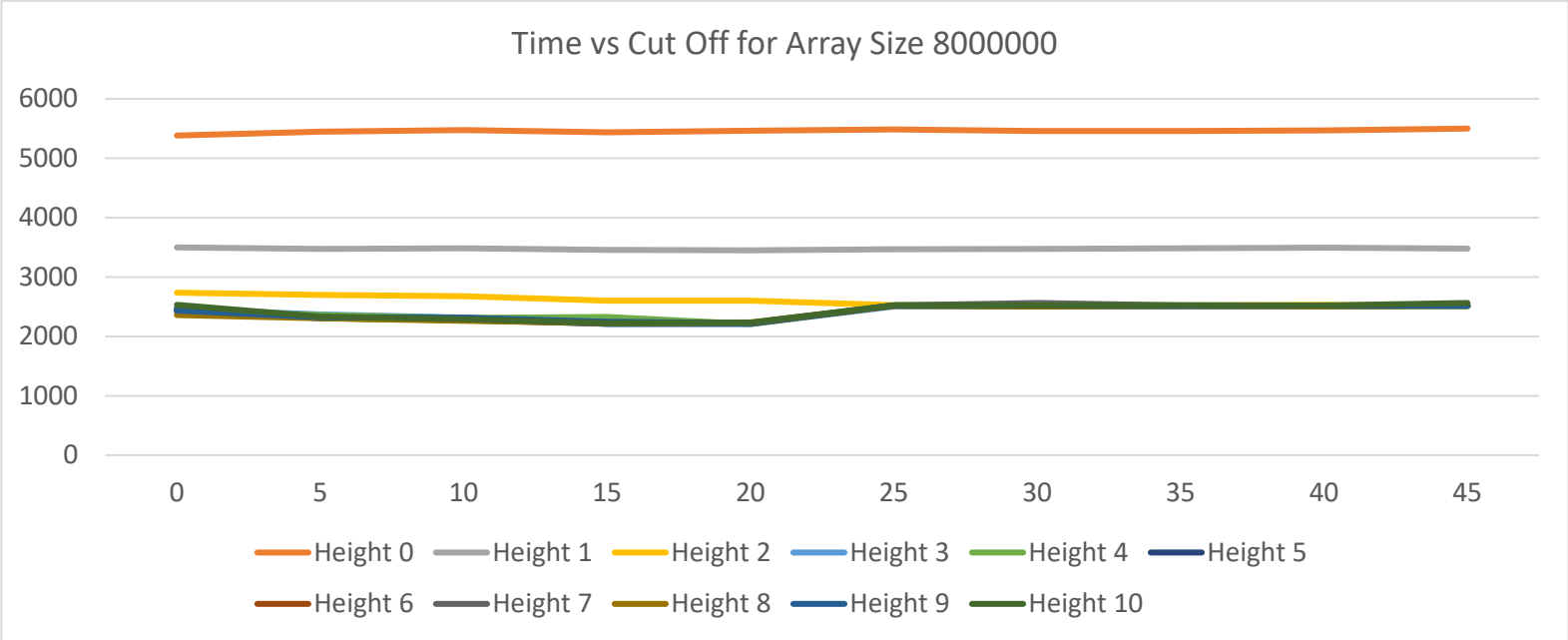
From observation 1, we find that the optimal cut off value is around 20% (check conclusion for further explanation), hence we fix the cut off at 20%. Fix the array size and vary the level of parallelism to judge its effect on the performance. Repeat this for different sizes of array to check for the consistency.

- Result:

From Experiment 1:

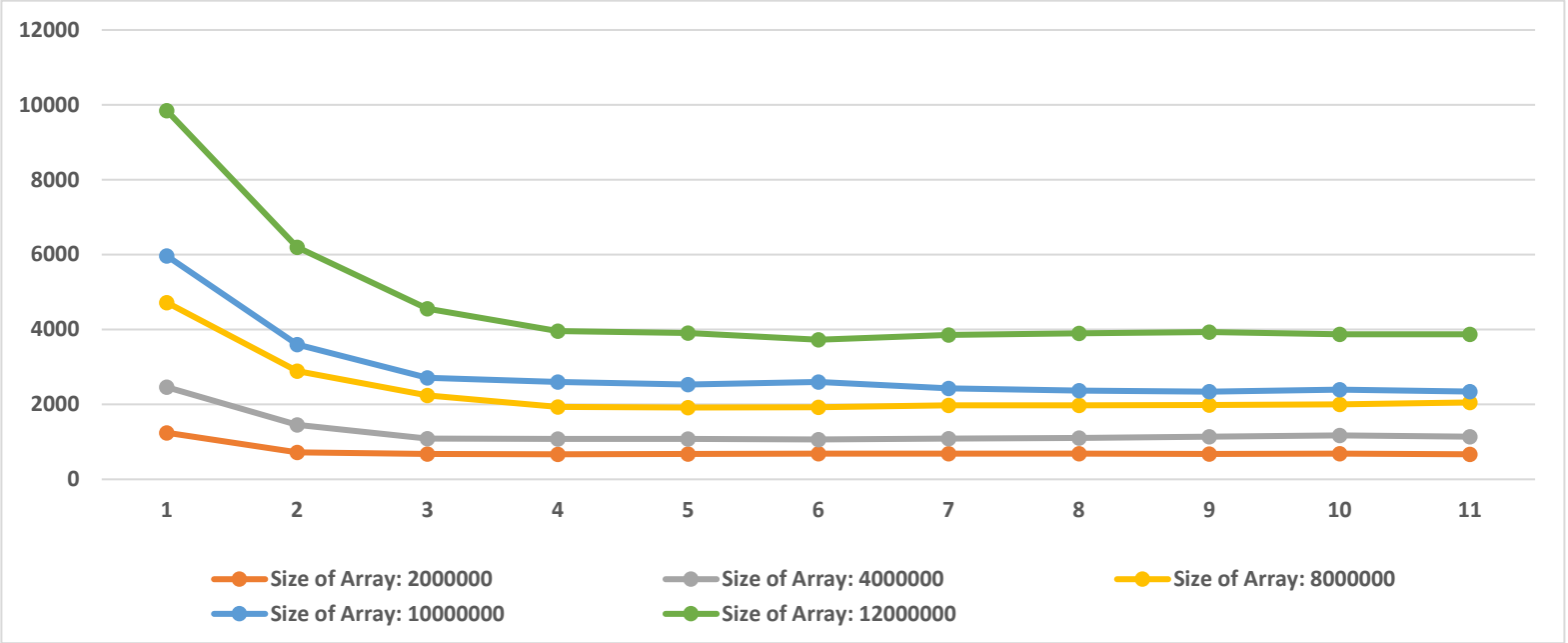
It is observed from the graphs that for all sizes of arrays and for levels of parallelism > 2, the optimal value for cut off is 20% of the array size.



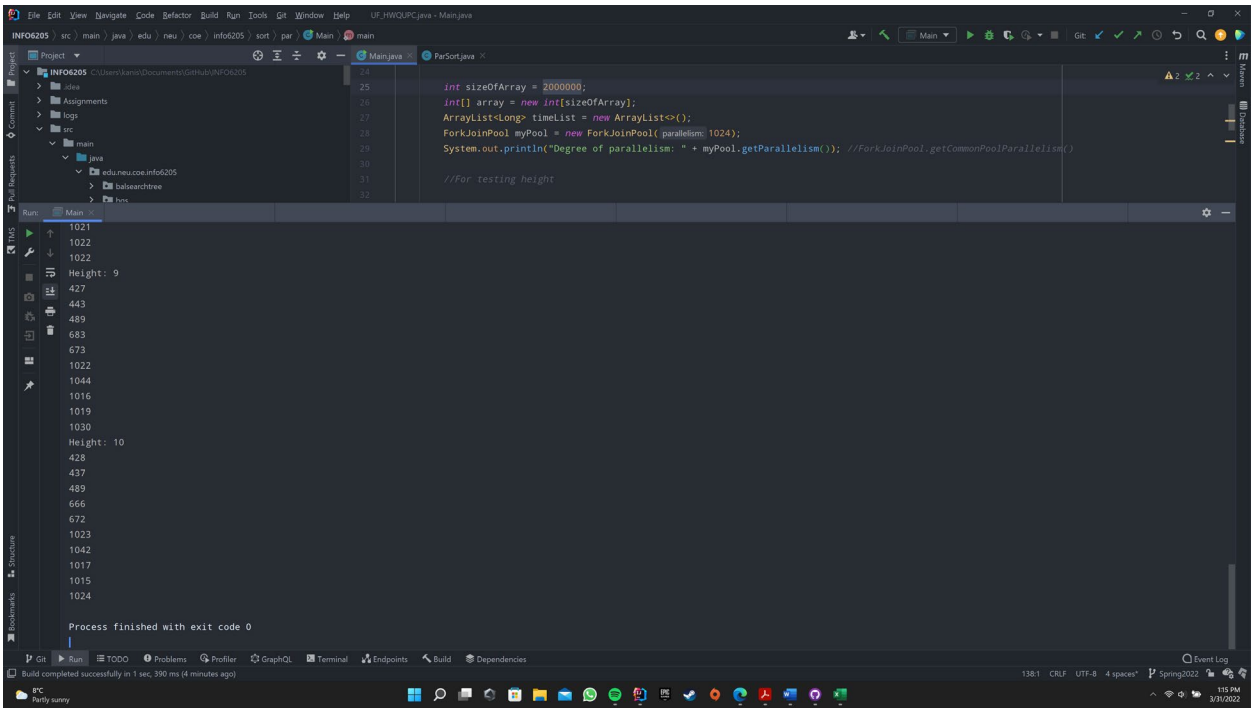


It is observed that at height = 0 (Only system sort, no parallel sort), the performance is worst. By having more levels of height sorted parallelly, the performance is increased. At height = 1 and height = 2, we see a significant increase. From height = 3, the performance improvement is not that significant. The advantage of sorting parallelly is balanced by the disadvantage of having to mange the partitions and threads.

From Experiment 2:



As we can see from the graph above, that with no parallel sorting for any size of array, it will give us the worst performance. As we increase the level of parallelism, the performance increases up to a height of 3. Any increase in threads will not lead to many gains in the performance.



Data for Observation 2:

Array Size: 2000000

Cutoff	Height 0	Height 1	Height 2	Height 3	Height 4	Height 5	Height 6	Height 7	Height 8	Height 9	Height 10
1	1325	855	625	595	603	595	657	606	700	855	861
6	1271	839	627	566	563	572	584	566	591	597	605
11	1261	820	650	567	571	564	566	563	572	573	576
16	1280	827	723	545	564	545	555	545	564	555	557
21	1314	814	621	569	567	545	553	542	626	548	564
26	1327	824	619	608	606	655	604	608	604	610	622
31	1271	831	606	608	607	619	603	611	608	609	612
36	1290	825	609	608	625	600	605	604	608	614	618
41	1342	824	615	606	615	599	602	604	603	633	615
46	1278	822	609	651	604	600	614	606	607	626	621

Array Size: 4000000

Cutoff	Height 0	Height 1	Height 2	Height 3	Height 4	Height 5	Height 6	Height 7	Height 8	Height 9	Height 10
1	2663	1722	1338	1177	1188	1207	1226	1186	1255	1233	1180
6	2671	1698	1319	1161	1164	1150	1137	1143	1143	1133	1136
11	2678	1696	1311	1171	1140	1136	1135	1142	1316	1146	1157
16	2677	1693	1269	1122	1111	1103	1121	1107	1458	1112	1114
21	2665	1699	1269	1114	1105	1107	1124	1108	1270	1119	1104
26	2668	1704	1236	1238	1232	1257	1239	1245	1253	1238	1242
31	2670	1707	1239	1239	1236	1238	1245	1241	1246	1244	1242
36	2671	1701	1229	1256	1237	1244	1244	1241	1244	1242	1240
41	2673	1698	1240	1254	1236	1243	1240	1244	1241	1240	1241
46	2658	1705	1246	1252	1234	1239	1246	1250	1239	1244	1252

Array Size: 8000000

Cutoff	Height 0	Height 1	Height 2	Height 3	Height 4	Height 5	Height 6	Height 7	Height 8	Height 9	Height 10
1	5383	3499	2736	2459	2430	2465	2420	2442	2361	2435	2532
6	5450	3475	2701	2368	2362	2305	2317	2310	2308	2324	2332
–11	5473	3485	2679	2314	2313	2280	2284	2274	2264	2319	2291
16	5437	3456	2604	2209	2329	2213	2222	2216	2226	2248	2223
21	5464	3450	2602	2208	2209	2214	2220	2221	2236	2224	2232
26	5483	3470	2525	2507	2517	2519	2518	2523	2518	2519	2525
31	5458	3473	2508	2506	2511	2518	2515	2563	2507	2520	2520
36	5461	3484	2519	2511	2517	2516	2514	2520	2514	2518	2525
41	5471	3494	2531	2508	2511	2518	2518	2511	2512	2521	2522
46	5500	3480	2509	2508	2514	2522	2521	2513	2515	2517	2563