始爾濱Z業大學数据库实验报告

题	目 _数据库查询算法的实现_
专	业 信息安全
学	号 <u>1180300829</u>
学	生 <u>余 涛</u>

指导教师 史建焘

1 实验目的

掌握多路归并排序算法,并用高级语言实现

2 实验环境

JAVA 语言

- 3 实验内容及要求
- 3.1 随机生成具有 1000000 条记录的二进制文件, 每条记录的长度为 16 字节

创建一个 Data 类储存数据:

```
public class Data {
    private int A; // 4字节int
    private String B; // 12位String
   public Data(int a, String b) {
        A = a;
        B = b;
创建数据代码如下:
// 产生数据
public void create_data(String filename) {
    try {
        FileOutputStream os = new FileOutputStream(filename);
        Random rm = new Random();
        for (int i = 0; i < data num; i++) {</pre>
           int A = rm.nextInt(Integer.MAX_VALUE); // 生成4字节int
             System.out.println(i + " " + A);
           os.write(sumHex(A));
           String B = getRandomStr( bytes: 12); // 生成12位String
             System.out.println(i + "...." + B);
           os.write(B.getBytes());
        }
        os.close();
    } catch (IOException e) {
        e.printStackTrace();
```

3.2 分成小文件:

1MB 内存 = 1024*1024 B, 而待排序文件大小=16B * 1000000, 若分为 16 个子集合,则每个子集合大小为 1000*1000B < 1MB。本机的内存页大小为 4KB, 1MB

内存共有256个内存页,16<256,因此分为16个子集合满足两趟扫描算法的要求。

第一趟分成小文件如下:

```
1. // 划分小文件
      public void split_file(String filename) {
        File myfile = new File(filename);
3.
        if (myfile.isFile() && myfile.exists()) {
4.
5.
              FileInputStream fi = new FileInputStream(myfile);
6.
             for (int i = 0; i < block_number; i++) { // 16 个子文件
7.
8.
                List<Data> templist = new ArrayList<>();
9.
                for (int j = 0; j < part_file_size / 16; j++) { // 每个文件的记录
   总数
10.
                   Data tempdata = get_data(fi);
                   if (tempdata != null) {
11.
                     templist.add(tempdata);
12.
                   }
13.
14.
                }
                Collections.sort(templist, new Comparator<Data>() { // 对子
15.
   文件内按照 A 进行排序
16.
                   @Override
17.
                   public int compare(Data o1, Data o2) {
                     return o1.getA() - o2.getA();
18.
19.
                   }
20.
                });
                FileOutputStream os = new FileOutputStream(prefix + i);
21.
22.
                for (Data tempdata: templist) { // 将 templist 中的所有记录读入
    一个小文件
23.
                   os.write(sumHex(tempdata.getA()));
24. //
                    System.out.println(tempdata.getA());
25.
                   os.write(tempdata.getB().getBytes());
26. //
                    System.out.println(tempdata.getB());
27.
                }
28.
                os.close();
                  System.out.println("-----
29. //
30.
              }
31.
              fi.close();
32.
           } catch (IOException e) {
              e.printStackTrace();
33.
34.
           }
35.
        }
```

3.3 归并排序:

按照 ppt 的算法生成代码即可,代码内容如下:

```
public void merge_sort() {
2.
        try {
3.
          int num = 1;
4. //
            int all = 1:
5.
          Data[][] M = new Data[block_number][256];
          Data[] M_compare = new Data[block_number]; // 比较缓冲区
6.
7.
          Data[] M output = new Data[block number]; // 输出缓冲区
          int[] Mi_next = new int[block_number]; // 记录 Mi 中元素移动的指针,
8.
   即 Mi 中移到第几个元素了
9.
          int[] Si_next = new int[block_number]; // 记录 Si 中读到第几块了
10.
          int[] flag = new int[block number]; // 记录某一块是否读完
          Data finished = new Data(-1, "finished"); // 记录已经读完的 Data
11.
          FileOutputStream os = new FileOutputStream("src/merge_sort_resul
12.
   t/result"); // 输出文件
13.
14.
          int i;
15.
          for (i = 0; i < block_number; i++) {
16.
             FileInputStream fi = new FileInputStream(prefix + i);
             for (int j = 0; j < page_size / 16; j++) { // 读 Si 的第一块存到 Mi
17.
18.
               M[i][j] = readData(fi);
19.
             }
             M_compare[i] = M[i][0]; // 第一个元素
20.
21.
             fi.close();
22.
          }
23.
          int P_output = 0; // 输出指针起始位置
24.
          while (getmin(M_compare) != -1) { // 如果 M_compare 有最小值
25.
             i = getmin(M_compare); // M_compare 最小值的位置
26.
             Data min_i = M_compare[i]; // M_compare 最小值
             M_output[P_output] = min_i; // 将第 i 个元素存入 M_output 的
27.
   P output 位置
28.
             P_output++; // P_output 指向下一个位置
29.
             Mi_next[i] = Mi_next[i] + 1; // 记录 Mi 中的位置的指针指向下一个
30.
             if (min_i.getA() == -1) { // 为 finished 跳出循环
               break;
31.
32.
             }
```

```
33.
              if (P_output == M_output.length) { // 如果 P_output 指向结束位置,
   输出 M output 的内容
34.
                for (Data data : M_output) {
                   System.out.println(num + ":" + data.getA());
35.
36. //
                    System.out.println(data.getA());
37.
                   os.write(CreateData.intToBytes(data.getA()));
38.
                   System.out.println(num + ":" + data.getB());
39. //
                    System.out.println(data.getA());
40.
                   os.write(data.getB().getBytes());
41.
                   num++;
42.
                }
43.
                P_output = 0; // P_output 置于输出缓冲区起始位置
44.
              }
              if (Mi_next[i] < M[i].length) { // 如果 Mi 有下一个元素,将 Mi 的下一
45.
   个元素放入 M_compare 的第 i 位置
46.
                M_compare[i] = M[i][Mi_next[i]];
47.
48.
                FileInputStream fi = new FileInputStream(prefix + i);
                fi.read(new byte[page_size * (Si_next[i] + 1)]); // 先读入已经
49.
   读过的块, 跳过
50.
                Si_next[i] = Si_next[i] + 1;
                if (flag[i] == 0) { // 如果还有下一块
51.
52.
                   for (int j = 0; j < page_size / 16; j++) { // 重新读入新的一
   块中所有内容到 M
53.
                     Data tempdata = readData(fi);
54.
                     if (tempdata == null) { // 没有下一块
55.
                        flag[i] = 1;
                        M[i][j] = finished;
56.
57.
                        break;
58.
59.
                     M[i][j] = tempdata;
60.
                   }
                   Mi_next[i] = 0;
61.
62.
                   M_{compare[i]} = M[i][0];
63.
                } else {
                   M_compare[i] = finished;
64.
65.
                }
66.
                fi.close();
              }
67.
68.
           } // M_compare 中所有值均为 Finished,没有最小值,算法结束
69.
        } catch (IOException e) {
70.
           e.printStackTrace();
71.
        }
72.
      }
```

3.4 实验结果

(1) 分成的 16 个小文件:

1000000 个数据总大小为 1000000 * 16 B, 分成 16 个小文件,则每个小文件大小为 1000000B,为 977KB,小文件大小满足要求。

文件大小:	977 KB 1,000,000 字节			
part_0		2021/5/14 18:08	文件	977 KB
part_1		2021/5/14 18:08	文件	977 KB
part_2		2021/5/14 18:08	文件	977 KB
part_3		2021/5/14 18:08	文件	977 KB
part_4		2021/5/14 18:08	文件	977 KB
part_5		2021/5/14 18:08	文件	977 KB
part_6		2021/5/14 18:08	文件	977 KB
part_7		2021/5/14 18:08	文件	977 KB
part_8		2021/5/14 18:08	文件	977 KB
part_9		2021/5/14 18:08	文件	977 KB
part_10		2021/5/14 18:08	文件	977 KB
part_11		2021/5/14 18:08	文件	977 KB
part_12		2021/5/14 18:08	文件	977 KB
part_13		2021/5/14 18:08	文件	977 KB
part_14		2021/5/14 18:08	文件	977 KB
part_15		2021/5/14 18:08	文件	977 KB

用 winhex 随便打开一个小文件查看:

按照 A 属性在小文件内部完成了排序:

00117536	OE 70 36 E7 57 6D 4B 37	7A 52 31 54 44	49 69 5A	OE 70 84 66 63 38 69 77 39 36 68 34 61 38 69 7A p6cWmK7zH	R1TDI
00117568	OE 70 D3 70 38 61 34 4A	55 70 66 36 37	39 37 44	OE 71 17 75 30 43 35 5A 38 63 77 39 30 76 34 67 póp8a4JUr	pf679
00117600	OE 71 3D 00 34 41 38 6E	36 34 32 54 77	30 51 38	OE 71 3D FD 31 44 61 6D 38 48 33 76 66 35 52 74 q= 4A8n64	42Tw0
00117632	0E 71 5A 1E 32 43 35 4E	38 59 35 30 38	31 38 38	OE 71 5D 9A 41 6A 74 4D 36 4B 30 4A 78 35 33 36 qZ 2C5N8	Y5081
00117664	0E 71 F3 0D 51 37 57 71	68 31 34 49 36	34 30 30	0E 72 51 F5 35 42 77 76 42 6B 69 30 31 33 36 39 q6 Q7Wqh:	14164
00117696	0E 73 2C DA 4D 64 62 31	32 4B 37 42 57	36 5A 33	OE 73 64 76 33 43 35 31 34 6C 36 4E 33 36 31 36 s, UMdb12F	K7BW6
00117728	OE 74 04 9B 6D 30 32 34	33 33 37 33 38	38 72 73	OE 74 3C 00 6E 55 30 30 30 36 4B 66 30 39 65 38 t >m02433	
00117760	OE 75 81 70 31 34 30 4E	37 35 68 55 61	31 73 6C	OE 76 7F 3E 35 48 36 32 32 78 38 43 31 4D 71 33 u p140N75	5hUm1
00117792	0E 78 03 23 35 39 48 79	57 36 7A 30 35	76 36 48	OE 79 79 1A 38 56 37 31 35 44 4B 31 53 38 39 41 x #59HyW6	6z05v
00117824	OE 7A 04 41 62 39 74 30	55 51 62 30 34	36 76 32	OE 7A 17 AE 59 39 31 38 4F 64 37 32 31 34 31 67 z Ab9t0UQ	Qb046
00117856	0E 7B 05 64 30 78 70 37	47 36 63 35 61	72 39 50	OE 7B 5C 91 66 68 34 76 38 46 36 72 37 4C 32 39 { d0xp7G6	6c5mr
00117888	0E 7C 29 58 39 30 36 33	33 61 35 32 30	30 77 44	OE 7D 62 1D 38 7A 76 36 50 33 37 68 77 34 47 39) X906338	a5200
00117920	OE 7D 73 9E 34 64 45 37	31 34 4D 34 55	31 35 39	OE 7D 80 84 52 6F 36 48 61 31 4B 52 34 73 37 58 }sž4dE714	4M4U1
00117952	0E 7E 11 F9 46 65 46 63	36 38 39 58 36	74 30 39	OE 7E 73 EC 39 50 73 45 46 42 33 34 34 41 34 6E ~ ùFeFc68	89X6t
00117984	OE 7F C3 C8 38 37 78 39	38 30 34 46 39	75 36 37	OE 7F C5 7F 4D 4D 31 72 74 67 62 45 39 39 37 61 ÃÈ87x980	04F9u
00118016	OE 80 34 C8 77 41 32 48	39 76 54 30 33	34 37 33	0E 80 51 48 37 57 48 77 77 73 33 34 38 69 54 4D €4ÈwA2H9v	vT034
00118048	OE 80 B4 B2 39 72 33 56	31 42 56 6C 53	31 50 67	0E 82 3C BF 49 37 34 32 32 31 39 76 36 56 58 37 €'29r3V1E	BV1S1
00118080	OE 82 85 29 36 6F 4F 41	47 7A 33 4D 30	39 6A 36	OE 82 CC AD 31 6C 36 46 5A 57 41 39 76 61 50 38 ,) 600AG	z3M09
00118112	0E 83 25 F1 4E 41 32 4F	56 66 67 38 73	4B 6F 47	OE 83 45 2D 36 68 66 32 4F 30 32 39 36 30 35 6A f%nNA2OV	fg8sK
00118144	OE 83 69 DB 58 38 30 76	34 6F 31 35 76	38 47 35	OE 83 E6 OF 63 30 79 74 46 35 37 34 77 4F 4D 34 fiûx80v4c	o15v8
00118176	OE 84 45 BD 34 31 57 39	4C 37 32 31 35	34 4B 48	OE 85 94 B1 37 35 31 72 58 57 37 48 34 31 38 36 "E%41W9L"	72154
00118208	OE 86 85 96 6A 39 32 77	52 56 36 35 64	33 30 34	OE 86 BO 8A 50 36 33 36 6C 43 64 31 30 68 6E 48 Tj92wRV	V65d3
00118240	OE 87 5A AE 30 67 31 6C	34 33 31 33 32	69 4C 35	OE 87 B5 B4 77 33 30 48 62 37 37 4E 33 47 39 34 #Z@Oq1143	3132i
00118272	OE 88 B5 DC 30 73 39 32	57 31 45 41 70	70 36 72	OE 88 E3 43 38 74 35 30 76 34 30 59 41 51 32 31 ^μÜOs92W3	1EApp
00118304	OE 88 FE F1 33 78 39 6A	32 46 61 6D 34	33 35 62	0E 89 0D 9F 32 39 53 72 4D 57 4D 34 30 54 39 51 ^pñ3x9j2i	Fam43
00118336	OE 89 3B 99 37 33 38 39	74 36 32 6B 34	59 47 56	OE 8A E9 15 45 50 35 32 62 32 63 49 36 35 37 67 %;™7389t6	62k4Y
00118368	OE 8B 67 87 75 36 35 58	47 39 36 6D 47	68 32 70	0E 8C 5E 52 34 37 39 44 34 39 71 35 65 33 7A 31 <q‡u65xg< td=""><td>96mGh</td></q‡u65xg<>	96mGh
00118400	0E 8C 87 CD 6B 67 34 37	63 61 47 35 76	64 64 35	OE 8D BA C2 6F 73 41 4B 30 33 75 5A 37 65 6E 31 @#fkg47ca	aG5vd
00118432	OE 8E 96 4A 73 39 78 54	33 47 67 49 31	. 50 32 6F	OE 8E CF 79 35 35 37 4B 45 42 6B 4B 37 37 32 5A Ž-Js9xT30	GgI1P
00118464	0E 8F 22 63 35 69 33 67	52 6D 6C 42 5F	67 6C 68	OE 8F 54 49 30 36 45 33 30 34 47 47 38 35 59 75 "c5i3qRr	mlBZg
00118496	0E 8F BD 9F 71 33 33 68	4E 74 4C 71 72	30 67 36	0E 8F CD 10 38 38 52 50 32 36 32 42 70 6D 78 48 % Yq33hNt	tLqr0
00118528	OE 8F DA DD 39 36 33 37	30 31 30 76 39	35 43 34	OE 8F E9 31 32 6C 67 6A 36 31 30 43 66 52 33 30 ÚÝ963703	10v95
00118560	0E 90 5C 7C 32 36 34 31	38 45 33 31 36	31 33 33	OE 90 9C E8 6F 30 34 34 6D 73 31 30 61 33 38 4F \ \ \ \ 26418F	E3161
00118592	0E 90 BE E0 52 39 72 33	36 7A 55 64 4A	30 43 57	0E 91 3F 9D 4F 30 39 33 32 33 34 37 7A 73 5A 46 % AR9r36	zUdJ0
00118624	0E 91 74 7D 32 6E 6D 34	38 35 54 31 77	75 5A 76	OE 91 A5 DE 31 39 36 72 34 5A 4E 38 63 70 43 71 't}2nm485	5T1zu
00118656	0E 91 AC F4 32 57 76 37	31 6D 74 48 53	34 76 59	0E 91 E5 8B 47 5A 38 30 35 35 6F 34 37 63 35 45 '¬ô2Wv71r	mtHS4
00118688	0E 91 FE A1 63 4A 52 4D	7A 63 67 74 37	32 38 30	OE 92 34 FA 57 33 67 54 39 34 32 58 4B 49 77 66 'p;cJRMzd	cgt72
00118720	OE 93 11 OE 32 6D 73 36	4A 61 32 36 67	4B 57 41	OE 94 71 98 33 4C 31 6D 67 6A 47 37 30 36 31 54 " 2ms6Ja	a26gK
00118752	0E 94 7F 6C 34 30 36 35	61 35 4F 36 45	30 34 38	OE 95 CA F6 36 73 32 32 35 44 4F 79 33 32 64 35 " 14065a5	506E0
00118784	0E 97 11 FC 33 48 30 59	47 33 5A 34 54	57 53 72	OE 97 30 D2 30 36 52 73 30 56 35 30 32 32 33 37 - ü3HOYG3	3Z4TW
00118816	0E 97 B8 BD 34 4C 32 77	32 31 38 49 34	31 46 31	0E 98 39 13 38 38 4A 33 46 33 37 44 33 39 55 38 - 44L2w2	18141
00118848	0E 98 7B 0C 31 4F 38 66	59 44 31 32 37	30 32 6B	0E 98 80 4E 46 34 6C 49 37 31 36 70 50 35 31 37 "{ 108fYI	D1270
00118880	OE 99 8A B5 65 34 32 35	49 37 32 46 35	33 36 36	0E 9A 8E B0 77 30 31 52 6B 35 31 32 4E 55 4A 36 ™Šµe425I	72F53
00118912	OE 9B 32 2D 6D 79 38 38	45 37 35 62 31	71 39 73	0E 9B 38 2C 77 56 78 4D 49 39 32 30 45 35 73 34 >2-my88E	75b1q
00118944	OE 9B 79 49 38 38 75 36	55 37 33 4A 31	4B 37 67	OE 9D 0E 6F 68 59 6E 67 31 57 32 6A 30 4A 32 57 >yI88u6U	73J1K
00118976	0E 9D 1B C8 47 72 49 37	66 49 73 31 37	72 37 38	OE 9E 75 19 36 59 6F 37 78 30 66 36 31 50 32 30 ÈGrI7f	Is17r
00119008	OE 9E E7 52 51 65 77 61	66 70 66 58 42	33 39 61	0E 9F A0 CA 35 34 47 6B 72 65 31 4A 42 70 77 35 žçRQewafp	
00119040				OE AO 65 C2 35 36 69 34 79 62 4F 5A 38 67 46 74 >MMyOv	0GW6p
00119072	or λ∩ 72 21 38 31 51 3∩	20 57 20 55 69	EB 3E 33	OF NO C2 N7 50 65 65 75 6F 62 6F 47 26 27 76 71 ■ -191000	MULTIPP

(2) 最后的输出文件:

源文件大小如下:

□ data 2021/5/14 18:06 文件 15,625 KB

输出文件大小如下:

□ result 2021/5/14 18:19 文件 15,625 KB

文件大小: 15.3 MB 16,000,000 字节

刚好 1000000 个数据,文件大小合适,然后查看内部 A 属性的顺序:

result				
Offset	0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	^
03710304	1D 2C A8 48 30 32 35 41	31 55 38 38 38 58 32 37	1D 2C B2 8F 4F 31 39 36 55 76 6C 31 71 63 34 76 .	"H025A1U888X
03710336	1D 2C CB 19 62 4F 36 37	32 34 32 73 37 35 44 30	1D 2C D1 B9 34 51 33 44 63 50 36 78 4A 72 33 33 ,	Ë b067242s75
03710368	1D 2C F3 C9 77 38 65 37	57 68 38 39 59 4C 35 31	1D 2C FE 5C 48 39 37 4D 39 34 6E 38 4E 78 35 4D .	óÉw8e7Wh89YL
03710400	1D 2D 06 2C 39 32 35 38	7A 4F 56 39 36 32 65 33	1D 2D 0D 1F 48 31 78 72 31 36 32 37 30 36 39 4E -	,9258z0V962
03710432	1D 2D 18 50 35 4D 35 36	53 61 32 62 45 35 77 64	1D 2D 1F 4A 61 36 36 79 30 39 74 30 33 52 42 54 -	P5M56Sa2bE5
03710464	1D 2D 29 8A 39 59 48 72	30 4B 6A 39 44 35 38 38	1D 2D 30 DB 59 72 36 4C 38 54 36 37 30 33 38 64 -)Š9YHr0Ki9D5
03710496	1D 2D 46 2E 6F 59 34 37	72 7A 78 4A 31 37 6B 48	1D 2D 53 3F 44 39 62 48 59 67 33 53 38 6F 51 31 -	F.oY47rzxJ17
03710528	1D 2D 56 FE 67 32 42 61	47 35 32 36 55 74 58 34	1D 2D 59 3C 70 5A 31 43 79 47 31 39 6B 57 32 31 -	-Vbg2BaG526Ut
03710560	1D 2D 5C B3 52 66 33 34	30 4F 39 71 44 47 4E 7A	1D 2D 5F 63 37 76 30 66 38 4D 6A 56 50 54 38 34 -	\3Rf34009qDG
03710592	1D 2D 62 76 6C 34 56 7A	56 67 61 48 74 48 39 69	1D 2D 65 03 4C 7A 39 32 39 6A 36 46 6A 47 4D 54 -	-bv14VzVgaHtH
03710624	1D 2D 67 92 39 4A 37 6F	47 37 6E 31 41 45 37 36	1D 2D 6F CD 70 65 39 57 32 72 31 35 6F 39 37 33 -	g'9J7oG7n1AE
03710656	1D 2D 70 74 72 38 37 33	39 32 35 56 65 38 34 32	1D 2D 72 06 30 4E 63 6A 6F 30 37 79 4A 30 65 47 -	ptr873925Ve8
03710688	1D 2D 75 03 30 6D 45 52	35 38 41 30 33 35 57 77	1D 2D 76 12 4A 30 79 38 56 57 32 65 57 62 73 31 -	u 0mER58A035
03710720	1D 2D 7B 66 36 71 37 59	6E 34 5A 41 55 34 43 75	1D 2D 7C 55 62 35 37 38 35 39 32 61 77 36 38 53 -	-{f6q7Yn4ZAU4
03710752	1D 2D 8D 1B 37 41 33 4F	35 37 6C 35 39 31 38 30	1D 2D 8D AB 55 37 31 54 34 53 6D 56 44 74 37 65 -	- 7A30571591
03710784	1D 2D 95 37 78 38 61 4B	71 36 37 47 51 35 4C 39	1D 2D 97 C3 76 6C 59 31 34 37 74 57 6E 39 59 33 -	•7x8aKq67GQ5
03710816	1D 2D 99 47 70 34 32 33	34 73 39 45 31 38 4B 69	1D 2D 9E 92 36 50 35 56 55 79 6D 77 64 64 35 62 -	-™Gp4234s9E18
03710848	1D 2D A3 1E 36 39 38 46	36 5A 4C 39 35 38 36 41	1D 2D AA CE 30 57 37 38 63 6B 6F 39 32 31 36 79 -	£ 698F6ZL958
03710880	1D 2D AC 6A 34 31 77 6C	32 34 56 7A 39 47 4E 36	1D 2D C8 8E 6F 70 30 41 58 33 5A 37 63 31 67 71 -	-j41w124Vz9G
03710912	1D 2D CC 5B 53 34 6D 74	38 34 34 36 31 4F 63 33	1D 2D DA 57 4C 37 38 37 38 4D 37 51 36 31 73 52 -	·Ì[S4mt844610
03710944	1D 2D E0 58 33 36 4C 63	48 56 37 48 38 6C 37 37	1D 2D E4 C5 42 4E 30 6E 38 32 32 34 57 4C 32 74 -	àX36LcHV7H81
03710976	1D 2D E9 71 31 36 38 46	78 31 38 30 37 39 4B 55	1D 2D EB 42 31 39 55 41 34 32 33 36 79 48 38 46	-éq168Fx18079
03711008	1D 2D F1 46 65 56 4D 36	36 6D 6F 38 37 73 67 38	1D 2D F1 5E 54 6D 49 38 35 33 36 39 37 71 6D 53 -	ñFeVM66mo87s
03711040	1D 2D F6 E1 39 48 37 58	32 5A 48 34 76 39 30 49	1D 2D F8 5C 42 72 43 63 55 30 6F 51 74 53 32 4D -	-öá9H7X2ZH4v9
03711072	1D 2D F9 8F 31 66 42 34	39 55 37 31 4F 69 42 5A	1D 2D FF 13 6B 46 36 44 30 72 78 76 30 57 33 31 -	ù 1fB49U710i
03711104	1D 2E 13 A8 31 31 59 39	4C 51 31 34 34 6C 30 38	1D 2E 15 21 38 39 38 76 65 31 35 71 54 38 48 64 .	"11Y9LQ1441
03711136	1D 2E 17 55 32 33 37 33	73 36 48 51 6C 7A 71 39		U2373s6HQ1z
03711168	1D 2E 1B 8E 6B 30 37 62	32 4C 47 49 58 30 33 73		Žk07b2LGIX0
03711200	1D 2E 20 24 38 33 58 46	67 39 39 39 67 4D 70 67		\$83XFg999gM
03711232	1D 2E 2D 71 39 72 39 39	37 34 61 55 30 34 39 6E		-q9r9974aU04
03711264	1D 2E 3E E8 79 37 31 4B	32 67 4A 4D 46 39 63 53		>èy71K2gJMF9
03711296	1D 2E 4B 4F 4D 31 44 38	35 68 66 33 34 32 33 56		KOM1D85hf342
03711328	1D 2E 53 51 30 35 45 38	4F 34 50 39 79 37 6E 38		SQ05E804P9y7
03711360	1D 2E 62 CB 59 7A 37 31	41 47 30 35 32 35 49 32		bËYz71AG0525
03711392	1D 2E 6D 4D 31 31 34 36	33 30 31 64 65 37 54 32		mM1146301de7
03711424	1D 2E 88 3E 42 49 39 75	6E 51 65 4F 66 61 61 54		^>BI9unQeOfa
03711456	1D 2E 8D 4E 31 55 4C 41	59 30 36 78 38 62 45 36		N1ULAY06x8b
03711488	1D 2E A7 5D 70 7A 37 68	46 6E 74 66 32 77 55 32		§]pz7hFntf2w
03711520	1D 2E BE 5A 4D 36 72 6A	50 50 31 32 30 47 37 30		%ZM6rjPP120G
03711552	1D 2E C7 0D 5A 79 73 6D	34 32 37 31 6E 39 4D 38		Ç Zysm4271n9
03711584	1D 2E D8 86 78 42 5A 6D	39 79 32 6A 4B 59 32 47		؆xBZm9y2jKY
03711616	1D 2E E0 47 73 4C 73 78	6D 42 37 59 32 75 36 36		àGsLsxmB7Y2u
03711648	1D 2E EA AC 32 65 36 6B	34 76 61 31 39 36 39 34		ê-2e6k4va196
03711680	1D 2F 02 52 57 31 6A 68	32 53 30 48 44 39 31 76		RW1jh2S0HD9
03711712	1D 2F 07 F5 35 37 30 70	32 6B 30 33 36 38 36 32		ő570p2k0368
03711744	1D 2F 0D 25 76 49 33 34	37 42 39 32 39 49 64 4E		%vI347B929I
03711776	1D 2F 15 DF 32 41 54 38	37 35 5A 33 31 33 6A 38		B2AT875Z313
03711808	1D 2F 1C 50 46 30 36 32	39 30 4F 6D 50 36 34 30	1D 2F 24 79 37 61 4C 36 30 6E 37 74 38 31 6A 38 /	PF062900mP6
03711870	111 76 70 PE 37 66 57 77	P. 48 44 PB 44 41 41 41	10 76 76 40 AA AA 57 38 30 35 58 65 71 60 35 60 7	TI INDEASON'T

可见输出文件完成了排序。

(3) 运行时间:

第一趟运行时间: **11.172**秒 第二趟运行时间: **10.823**秒

第一趟:使用 java 自带的 comparator 对子文件按照 A 属性排序,速度较快。

第二趟:由于每次读入文件的时候是一次性读入16个字节作为一个新的数据,相对于一次性读一个字节,效率较高。