山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

学号：202200130048	姓名： 陈静雯	班级： 6
实验题目：外排序		
实验学时：10	实验日期：4.17	

实验目的：
应用竞赛树结构模拟实现外排序。

软件开发工具：
Vscode

1. 实验内容
（1）设计并实现最小输者树结构 ADT，ADT 中应包括初始化、返回赢者，重构等基本操作。
（2）应用最小输者树设计实现外排序，外部排序中的生成最初归并串以及 K 路归并都应用竞赛树结构实现；
（3）随机创建一个较长的文件作为外排序的初始数据；设置归并路数以及缓冲区的大小；获得外排序的访问磁盘的次数并进行分析。可采用小文件来模拟磁盘块。

2. 数据结构与算法描述 （整体思路描述，所需要的数据结构与算法）
竞赛树（最小输者树）：
（1） 私有成员：numofplayer 比赛成员个数；int *tree 内部节点，数组描述，存储输者下标；thewin 每次赢的成员下标；T* player 参加比赛的成员数组；lowext 竞赛树最底层的外部节点个数，2*(n-s)，有的外部节点赢者直接晋级到上上层；offset 成员下标与对应内部节点相互转换的计算下标偏移量，2*s-1
（2） Winner（），loser（）函数，比较两者的 player 值，小的赢，返回下标
（3） Play（int p,int l,int r）函数，p 为树中的下标，l 与 r 比，更新 p 位置的值，如果 p 为奇数，说明可以往上更新父节点
（4） Replay（）重构函数，根据要重构的下标，先分两种情况，一种是下标对应的晋级者是在输者树的最底层，另一种是下标对应的会晋两级，就是在输者树的倒数第二层；如果是前者，treeindex=(offset+theplayer)/2;l=2*treeindex-offset; r=l+1，后者还要分左节点和右节点是不是都是>lowext 的成员，也有可能是一个<=lowext，一个>lowext，分情况处理。得到 l、r 和 treeindex 之后，考虑要修改的节点是不是最终赢者，如果是，往上重新与输者比更新父节点即可，如果不是用 l 与 r 进行重赛，更新输者树
外排序：
（1）顺串结构体：num 顺串号,key 关键字;文件结构体：path 路径,nvisit 访问次数,sumall 数字总个数,cachesize 缓冲区大小,numofdisk 生成文件个数，k 要求归并路数
（2）init（）用随机数生成一个原文件
（3）make_seq（），生成顺串，构建一个 cachesize 大小的顺串输者树，然后第一次先将树填满，每次得到一个树中的最小值，输出到对应顺串的文件中，然后从原文件取出一个新的数，如果这个数比最小值小，它的顺串号为最小值的顺串号+1，如果比最小值大，顺串号相同

（4）merge_k（）k 路合并，对 k 个文件，有对应的 k 个缓冲区，和第 k+1 个输出缓冲区，缓冲区用队列进行存储，每次先填满对应的输入缓冲区，每个缓冲区的第一个数构成大小为 k 的输者树，得到最终赢者，放入输出缓冲区，输出缓冲区满再都输出到文件，从对应顺串号的缓冲区 pop 一个队首放入输者树对应的位置，进行重构，如果输入缓冲区空了，就从对应文件读满缓冲区。最后把输出缓冲区剩余的数全部输出到文件。

3. 测试结果（测试输入，测试输出）

```
size of cache:
3
initial number of disk:
20
number of merge:
2
the sum of seq:4
visit disk:72
PS D:\code repository\code>
```

Disk: 590 148 730 481 394 593 723 302 727 514 496 151 714 219 477 331 299 38 41 138
Disk1: 148 481 590 593 723 730
Disk2: 302 394 496 514 714 727
Disk3: 151 219 299 331 477
Disk4: 38 41 138
Disk5: 148 302 394 481 496 514 590 593 714 723 727 730
Disk6: 38 41 138 151 219 299 331 477
Disk7: 38 41 138 148 151 219 299 302 331 394 477 481 496 514 590 593 714 723 727 730

```
size of cache:
100
initial number of disk:
1000
number of merge:
4
the sum of seq:6
visit disk:2040
```

Disk: 180 140 768 559 104 342 755 669 745 532 12 343 619 596 91 369 516 722 694 621 242 866 210 554 490 21 73 765 779 397 948 689 226 836 121 883 640 242 902 577 353 934 455 157 558 591 571 117 165 475 750 917 230 215 123 598 554 1000 24 568 840 226 719 654 192 591 241 233 43 474 406 56 490 652 91 726 686 39 707 317 512 194 249 690 451 748 932 376 983 500 116 66 134 856 893 3 228 444 737 48 115 598 779 529 380 816 961 355 821 685 689 817 743 288 392 301 463 585 934 353 648 169 366 259 855 512 901 937 783 941 169 768 870 62 172 964 776 288 896 507 762 161 349 474 105 132 170 197 972 734 369 769 487 623 653 844 338 698 512 39 442 824 424 922 6 129 913 641 28 205 754 806 989 451 465 527 214 144 831 58 665 786 884 174 35 213 810 675 975 110 437 365 315 878 646 269 970 721 668 385 90 135 644 877 430 123 682 457 750 938 185 492 981 455 999 434 892 797 229 307 534 259 261 356 652 855 4 33 353 108 811 331 976 454 936 198 997 168 522 823 225 172 990 732 506 571 875 956 490 965 789 217 535 547 744 459 995 10 458 527 805 219 822 927 72 506 94 484 818 395 681 497 751 932 870 778 575

855 80 460 398 46 698 279 714 724 749 434 258 569 687 830 349 30 126 165 825 370 766
836 237 36 263 359 505 115 402 716 740 378 125 211 710 639 885 364 255 199 852 586
416 258 932 646 761 709 482 582 514 95 926 288 510 371 243 674 29 280 722 404 233
334 323 784 891 314 606 968 791 963 966 297 45 846 992 143 33 471 132 562 21 537 809
103 646 104 773 177 700 333 221 553 38 265 1 836 470 606 68 52 168 130 179 266 553
997 852 882 605 189 881 495 57 984 437 169 635 523 878 220 191 304 315 340 51 836
520 29 341 858 808 199 987 930 987 194 180 719 128 654 597 88 483 87 398 228 91 839
614 571 990 932 308 802 214 635 152 367 488 109 465 79 385 762 761 485 756 635 766
915 851 108 417 45 562 121 956 100 306 856 8 241 716 752 157 403 611 704 206 994 616
132 635 85 616 947 984 729 801 916 215 66 149 658 618 991 686 926 171 329 946 428
615 485 1 432 208 419 533 507 755 509 488 696 242 659 573 660 602 22 768 446 263 853
639 579 734 45 525 604 619 755 829 692 525 535 500 501 999 665 490 931 560 287 634
947 509 555 119 500 5 78 249 407 14 689 497 372 750 142 162 878 405 346 865 711 505
278 742 334 897 158 482 560 366 594 451 340 726 563 679 996 107 424 246 833 636 265
618 935 635 339 83 937 535 646 918 941 770 491 126 315 637 723 374 906 595 812 518
180 548 676 628 603 963 779 92 764 898 468 227 91 803 580 417 359 314 399 949 839
754 535 765 909 21 382 214 106 889 678 281 242 463 390 671 937 381 448 838 319 630
225 36 896 656 929 699 495 647 852 638 716 872 888 107 659 912 663 377 63 870 47 828
150 23 473 999 311 617 855 633 394 265 665 548 263 737 34 105 705 732 758 456 348
220 918 543 349 310 832 496 537 844 665 192 377 380 892 365 732 374 8 576 269 836
956 523 102 877 198 221 734 675 69 131 975 878 579 569 147 511 655 226 752 866 858
741 865 518 121 776 600 342 106 633 161 669 698 746 952 590 310 501 333 874 777 953
694 374 477 627 694 411 870 248 629 903 611 554 409 21 134 729 11 889 398 412 436
462 531 561 810 835 914 345 566 563 841 463 832 220 574 26 13 738 483 497 480 909
492 815 110 537 876 816 526 133 21 930 241 974 62 31 606 556 370 490 42 804 495 232
646 840 180 721 601 821 238 470 543 819 189 118 618 82 99 348 560 643 228 611 493
100 172 781 659 377 462 151 215 31 401 314 550 178 496 838 336 50 849 919 292 481
277 228 892 200 450 662 298 667 166 750 194 266 173 690 754 674 557 114 784 739 264
93 891 90 73 145 108 418 299 619 822 495 686 252 668 340 387 653 778 205 562 820 423
495 953 431 370 503 376 872 290 256 524 210 706 185 585 283 117 750 307 835 694 909
462 4 763 476 332 938 506 170 31 700 501 411 64 931 379 136 528 259 591 742 588 561
965 981 590 836 641 700 794 926 626 219 774 940 304 722 214 228 914 760 542 607 595
238 329 143 349 829 506 819 318 725 906 833 563 752 465 491 970 526 256 205 268 906
142 621 963 484 465 420 360 553 951 237 953 728 78 43 9 750 393 200 416
Disk1: 3 12 21 24 39 43 48 56 66 73 91 91 104 115 116 117 121 123 134 140 157 165
169 180 192 194 210 215 226 226 228 230 233 241 242 242 249 259 288 288 301 317 342
343 349 353 353 355 366 369 376 380 392 397 406 444 451 455 463 474 474 475 487 490
490 500 507 512 512 512 516 529 532 554 554 558 559 568 571 577 585 591 591 596 598
598 619 621 623 640 641 648 652 653 654 665 669 675 685 686 689 689 690 694 698 707
719 721 722 726 734 737 743 745 748 750 750 754 755 762 765 768 768 769 776 779 779
783 786 797 806 810 811 816 817 821 824 831 836 840 844 855 855 856 866 870 877 878
883 884 892 893 896 901 902 913 917 922 932 934 934 936 937 938 941 948 956 961 964
965 970 972 975 976 981 983 989 990 995 997 999 1000
Disk2: 4 6 10 28 33 35 39 46 58 62 72 80 90 94 105 108 110 123 126 129 132 135 144
161 165 168 169 170 172 172 174 185 197 198 205 211 213 214 217 219 225 229 237 255
258 258 259 261 263 269 279 288 307 315 331 338 349 353 356 359 364 365 369 370 371

378 385 395 398 402 404 416 424 430 434 434 437 442 451 454 455 457 458 459 460 465 471 482 484 490 492 497 505 506 506 510 514 522 527 527 534 535 537 547 553 553 562 569 571 575 582 586 605 606 606 635 639 644 646 646 646 652 668 674 681 682 687 698 700 709 710 714 716 722 724 732 740 744 749 751 761 766 773 778 784 789 791 805 808 809 818 822 823 825 830 836 836 836 839 846 852 852 855 858 870 875 878 881 882 885 891 915 926 927 930 932 932 932 956 963 966 968 984 987 987 990 992 997

Disk3: 1 8 21 29 29 30 33 36 38 45 45 51 52 57 66 68 79 85 87 88 91 95 100 103 104 108 109 115 121 125 128 130 132 132 143 149 152 157 168 169 171 177 179 180 189 191 194 199 199 206 208 214 215 220 221 228 233 241 242 243 263 265 266 280 287 297 304 306 308 314 315 323 329 333 334 340 341 367 372 385 398 403 407 417 419 428 432 437 446 465 470 483 485 485 488 488 490 495 497 500 500 501 505 507 509 509 520 523 525 525 533 535 555 560 560 562 563 571 573 579 594 597 602 604 611 614 615 616 616 618 618 619 634 635 635 635 635 636 637 639 646 654 658 659 660 665 676 679 686 689 692 696 704 711 716 719 723 726 729 734 742 750 752 754 755 755 756 761 762 764 765 766 768 770 779 801 802 803 812 829 833 838 839 851 852 853 856 865 872 878 888 889 896 897 898 906 909 912 916 918 926 929 931 935 937 937 941 946 947 947 949 963 984 991 994 996 999 999

Disk4: 1 5 14 21 22 23 34 36 45 47 63 78 83 91 92 105 106 107 107 119 126 142 150 158 162 180 192 198 214 220 221 225 227 242 246 249 263 265 265 269 278 281 310 311 314 315 319 334 339 340 342 346 348 349 359 365 366 374 374 377 377 380 381 382 390 394 399 405 411 417 424 448 451 456 463 468 473 477 482 491 495 496 501 511 518 518 523 531 535 535 537 543 548 548 554 561 563 566 569 574 576 579 580 590 595 600 603 611 617 627 628 629 630 633 633 638 647 655 656 659 663 665 665 669 671 675 678 694 694 698 699 705 716 721 729 732 732 734 737 738 741 746 752 758 776 777 804 810 815 816 819 821 828 832 832 835 836 840 841 844 855 858 865 866 870 870 874 876 877 878 889 892 892 903 909 914 918 919 930 952 953 956 974 975

Disk5: 8 11 13 21 21 26 31 31 42 50 62 69 73 82 90 93 99 100 102 106 108 110 114 118 121 131 133 134 145 147 151 161 166 172 173 178 180 185 189 194 200 205 210 215 220 226 228 228 232 238 241 248 252 256 264 266 277 283 290 292 298 299 307 310 314 332 333 336 340 345 348 370 370 374 376 377 379 387 398 401 409 411 412 418 423 431 436 450 462 462 462 463 470 476 480 481 483 490 492 493 495 495 495 496 497 501 503 506 506 524 526 526 528 537 542 543 550 553 556 557 560 561 562 563 585 588 590 591 595 601 606 607 611 618 619 621 626 641 643 646 653 659 662 667 668 674 686 690 694 700 700 706 722 725 728 739 742 750 750 750 752 754 760 763 774 778 781 784 794 819 820 822 829 833 835 836 838 849 872 891 906 906 909 914 926 931 938 940 951 953 953 963 965 970 981

Disk6: 4 9 31 43 64 78 117 136 142 143 170 200 205 214 219 228 237 238 256 259 268 304 318 329 349 360 393 416 420 465 465 484 491

Disk7: 1 1 3 4 5 6 8 10 12 14 21 21 21 22 23 24 28 29 29 30 33 33 34 35 36 36 38 39 39 43 45 45 45 46 47 48 51 52 56 57 58 62 63 66 66 68 72 73 78 79 80 83 85 87 88 90 91 91 91 91 92 94 95 100 103 104 104 105 105 106 107 107 108 108 109 110 115 115 116 117 119 121 121 123 123 125 126 126 128 129 130 132 132 132 134 135 140 142 143 144 149 150 152 157 157 158 161 162 165 165 168 168 169 169 169 170 171 172 172 174 177 179 180 180 180 185 189 191 192 192 194 194 197 198 198 199 199 205 206 208 210 211 213 214 214 214 215 215 217 219 220 220 221 221 225 225 226 226 227 228 228 229 230 233 233 237 241 241 242 242 242 242 243 246 249 249 255 258 258 259 259 261 263 263

263 265 265 265 266 269 269 278 279 280 281 287 288 288 288 297 301 304 306 307 308
310 311 314 314 315 315 315 317 319 323 329 331 333 334 334 338 339 340 340 341 342
342 343 346 348 349 349 349 353 353 353 355 356 359 359 364 365 365 366 366 367 369
369 370 371 372 374 374 376 377 377 378 380 380 381 382 385 385 390 392 394 395 397
398 398 399 402 403 404 405 406 407 411 416 417 417 419 424 424 428 430 432 434 434
437 437 442 444 446 448 451 451 451 454 455 455 456 457 458 459 460 463 463 465 465
468 470 471 473 474 474 475 477 482 482 483 484 485 485 487 488 488 490 490 490 490
491 492 495 495 496 497 497 500 500 500 501 501 505 505 506 506 507 507 509 509 510
511 512 512 512 514 516 518 518 520 522 523 523 525 525 527 527 529 531 532 533 534
535 535 535 535 537 537 543 547 548 548 553 553 554 554 554 555 558 559 560 560 561
562 562 563 563 566 568 569 569 571 571 571 573 574 575 576 577 579 579 580 582 585
586 590 591 591 594 595 596 597 598 598 600 602 603 604 605 606 606 611 611 614 615
616 616 617 618 618 619 619 621 623 627 628 629 630 633 633 634 635 635 635 635 635
636 637 638 639 639 640 641 644 646 646 646 646 647 648 652 652 653 654 654 655 656
658 659 659 660 663 665 665 665 665 668 669 669 671 674 675 675 676 678 679 681 682
685 686 686 687 689 689 689 690 692 694 694 694 696 698 698 698 699 700 704 705 707
709 710 711 714 716 716 716 719 719 721 721 722 722 723 724 726 726 729 729 732 732
732 734 734 734 737 737 738 740 741 742 743 744 745 746 748 749 750 750 750 751 752
752 754 754 755 755 755 756 758 761 761 762 762 764 765 765 766 766 768 768 768 769
770 773 776 776 777 778 779 779 779 783 784 786 789 791 797 801 802 803 804 805 806
808 809 810 810 811 812 815 816 816 817 818 819 821 821 822 823 824 825 828 829 830
831 832 832 833 835 836 836 836 836 836 838 839 839 840 840 841 844 844 846 851 852
852 852 853 855 855 855 855 856 856 858 858 865 865 866 866 870 870 870 870 872 874
875 876 877 877 878 878 878 878 881 882 883 884 885 888 889 889 891 892 892 892 893
896 896 897 898 901 902 903 906 909 909 912 913 914 915 916 917 918 918 919 922 926
926 927 929 930 930 931 932 932 932 932 934 934 935 936 937 937 937 938 941 941 946
947 947 948 949 952 953 956 956 956 961 963 963 964 965 966 968 970 972 974 975 975
976 981 983 984 984 987 987 989 990 990 991 992 994 995 996 997 997 999 999 999 1000
Disk8: 1 1 3 4 4 5 6 8 8 9 10 11 12 13 14 21 21 21 21 21 22 23 24 26 28 29 29 30 31
31 31 33 33 34 35 36 36 38 39 39 42 43 43 45 45 45 46 47 48 50 51 52 56 57 58 62 62
63 64 66 66 68 69 72 73 73 78 78 79 80 82 83 85 87 88 90 90 91 91 91 91 92 93 94 95
99 100 100 102 103 104 104 105 105 106 106 107 107 108 108 108 109 110 110 114 115
115 116 117 117 118 119 121 121 121 123 123 125 126 126 128 129 130 131 132 132 132
133 134 134 135 136 140 142 142 143 143 144 145 147 149 150 151 152 157 157 158 161
161 162 165 165 166 168 168 169 169 169 170 170 171 172 172 172 173 174 177 178 179
180 180 180 180 185 185 189 189 191 192 192 194 194 194 197 198 198 199 199 200 200
205 205 205 206 208 210 210 211 213 214 214 214 214 215 215 215 217 219 219 220 220
220 221 221 225 225 226 226 226 227 228 228 228 228 228 229 230 232 233 233 237 237
238 238 241 241 241 242 242 242 242 243 246 248 249 249 252 255 256 256 258 258 259
259 259 261 263 263 263 264 265 265 265 266 266 268 269 269 277 278 279 280 281 283
287 288 288 288 290 292 297 298 299 301 304 304 306 307 307 308 310 310 311 314 314
314 315 315 315 317 318 319 323 329 329 331 332 333 333 334 334 336 338 339 340 340
340 341 342 342 343 345 346 348 348 349 349 349 349 353 353 353 355 356 359 359 360
364 365 365 366 366 367 369 369 370 370 370 371 372 374 374 374 376 376 377 377 377
378 379 380 380 381 382 385 385 387 390 392 393 394 395 397 398 398 398 399 401 402
403 404 405 406 407 409 411 411 412 416 416 417 417 418 419 420 423 424 424 428 430

431 432 434 434 436 437 437 442 444 446 448 450 451 451 451 454 455 455 456 457 458
459 460 462 462 462 463 463 463 465 465 465 465 468 470 470 471 473 474 474 475 476
477 480 481 482 482 483 483 484 484 485 485 487 488 488 490 490 490 490 490 491 491
491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491
491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491
491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491 491
491 491 491 491 4 9 31 43 64 78 117 136 142 143 170 200 205 214 219 228 237 238 256
259 268 304 318 329 349 360 393 416 420 465 465 484 491 492 492 493 495 495 495 495
495 496 496 497 497 497 500 500 500 501 501 501 503 505 505 506 506 506 506 507 507
509 509 510 511 512 512 512 514 516 518 518 520 522 523 523 524 525 525 526 526 527
527 528 529 531 532 533 534 535 535 535 535 537 537 537 542 543 543 547 548 548 550
553 553 553 554 554 554 555 556 557 558 559 560 560 560 561 561 562 562 562 563 563
563 566 568 569 569 571 571 571 573 574 575 576 577 579 579 580 582 585 585 586 588
590 590 591 591 591 594 595 595 596 597 598 598 600 601 602 603 604 605 606 606 606
607 611 611 611 614 615 616 616 617 618 618 618 619 619 619 621 621 623 626 627 628
629 630 633 633 634 635 635 635 635 635 636 637 638 639 639 640 641 641 643 644 646
646 646 646 646 647 648 652 652 653 653 654 654 655 656 658 659 659 659 660 662 663
665 665 665 665 667 668 668 669 669 671 674 674 675 675 676 678 679 681 682 685 686
686 686 687 689 689 689 690 690 692 694 694 694 694 696 698 698 698 699 700 700 700
704 705 706 707 709 710 711 714 716 716 716 719 719 721 721 722 722 722 723 724 725
726 726 728 729 729 732 732 732 734 734 734 737 737 738 739 740 741 742 742 743 744
745 746 748 749 750 750 750 750 750 750 751 752 752 752 754 754 754 755 755 755 756
758 760 761 761 762 762 763 764 765 765 766 766 768 768 768 769 770 773 774 776 776
777 778 778 779 779 779 781 783 784 784 786 789 791 794 797 801 802 803 804 805 806
808 809 810 810 811 812 815 816 816 817 818 819 819 820 821 821 822 822 823 824 825
828 829 829 830 831 832 832 833 833 835 835 836 836 836 836 836 836 838 838 839 839
840 840 841 844 844 846 849 851 852 852 852 853 855 855 855 855 856 856 858 858 865
865 866 866 870 870 870 870 872 872 874 875 876 877 877 878 878 878 878 881 882 883
884 885 888 889 889 891 891 892 892 892 893 896 896 897 898 901 902 903 906 906 906
909 909 909 912 913 914 914 915 916 917 918 918 919 922 926 926 926 927 929 930 930
931 931 932 932 932 932 934 934 935 936 937 937 937 938 938 940 941 941 946 947 947
948 949 951 952 953 953 953 956 956 956 961 963 963 963 964 965 965 966 968 970 970
972 974 975 975 976 981 981 983 984 984 987 987 989 990 990 991 992 994 995 996 997
997 999 999 999 1000

4. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）

可以改进的就是最初构建顺串时没有利用输入输出缓冲区，所以读取文件次数较多，可以优化一下。

5. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）

```
#include<iostream>
#include<string>
#include<fstream>
#include<cmath>
#include<cstdlib>
```

```
#include<ctime>
#include<queue>
using namespace std;

template<class T>
class losertree{   //最小输者树，小的赢，数组下标都从 1 开始
public:
    losertree(T* theplayer, int n);
    int getwin(){
        return tree[0];       //返回赢者，即最小的数的下标
    }
    int winner(int x, int y);
    int loser(int x, int y);
    void play(int p,int l,int r);
    void replay(int theplayer,T thevalue);//theplayer 数组下标,thevalue 要修改的新值
    ~losertree(){
        delete [] tree;
        delete [] thewin;
    }
private:
    int numofplayer; //比赛成员个数
    int* tree;       //内部节点，数组描述，存储输者下标
    int* thewin;   //每次赢的成员下标
    T* player;       //参加比赛的成员数组
    int lowext;      //竞赛树最底层的外部节点个数, 2*(n-s), 有的外部节点赢者直接晋级到上上层
    int offset;      //成员下标与对应内部节点相互转换的计算下标偏移量, 2*s-1
};

template<class T>
losertree<T>::losertree(T* theplayer,int n){ //theplayer 为比赛成员，n 为成员个数
    if(n<2) return;    //节点数必大于等于 2
    numofplayer=n;
    player=theplayer;
    tree=new int [n+1];//从 1 到 n
    thewin=new int [n+1];
    int s;
    for(s=1;2*s<=n-1;s*=2);//s=2^log(n-1)-1,最底层最左节点 index，这样计算 s 更快
    lowext=2*(n-s); //lowext 和 offset 的计算公式，不会推、硬记
    offset=2*s-1;
    for(int i=2;i<=lowext;i+=2) play((i+offset)/2,i-1,i);//从初始数组开始比，得到输者树最底层元素，i-1 和 i 所以每次 i+2，到 lowext 即遍历最底层对应的外部节点
    int t=0;
    if(n%2==1){//奇数节点,则有一个节点单独和前面外部节点的赢者比，因为奇数节点的最后一个不产生最底层的赢者，所以赢者树总数为 n-1
```

```
        play(n/2,thewin[n-1],lowext+1);//前面一组外部节点的赢者放在赢者数组的最后一个，与下一个单独的外部节点下标为 lowext+1，比较之后赢者放在内部节点的倒数第二层，n 为奇数，n/2=(n-1)/2，即 n-1 的赢者下标为 n/2
        t=lowext+3;//把 lowext 下一个节点单独处理后，只剩下 0 个或偶数个剩下的外部节点
    }
    else{
        t=lowext+2;//t 为要处理的剩下的外部节点的下一组的右成员下标
    }
    for(int i=t;i<=n;i+=2) play((i-lowext+n-1)/2,i-1,i);//剩下的外部节点，它们的赢者连晋两级，这个下标也不会推 TwT
    tree[0]=thewin[1];
}

template<class T>
int losertree<T>::winner(int x,int y){ //小的赢，返回赢者在成员数组中的下标
    if(player[x]<=player[y]) return x;
    else return y;
}

template<class T>
int losertree<T>::loser(int x,int y){
    if(player[x]<=player[y]) return y;
    else return x;
}

template<class T>
void losertree<T>::play(int p,int l,int r){
    tree[p]=loser(l,r);
    thewin[p]=winner(l,r);
    while(p%2==1 && p>1){  //树下标为奇数，可以往上得出上层晋级者
        tree[p/2]=loser(thewin[p-1],thewin[p]);
        thewin[p/2]=winner(thewin[p-1],thewin[p]);
        p=p/2;
    }
}

template<class T>
void losertree<T>::replay(int theplayer, T thevalue){//某个成员值变了，重构树
    int n=numofplayer;
    if(theplayer<=0||theplayer>n) return; //下标合法
    player[theplayer]=thevalue;
    int treeindex,l,r;//treeindex 为树中对应要修改的节点下标
    if(theplayer<=lowext){//修改的节点在竞赛树底层的外部节点
        treeindex=(offset+theplayer)/2;
        l=2*treeindex-offset;//l=theplayer，或者 theplayer 左边，看 player 奇偶，通
```

```
过 treeindex 计算省略了讨论
        r=l+1;
    }
    else{
        treeindex=(theplayer-lowext+n-1)/2;//记
        if(2*treeindex==n-1){ //n 为奇数，且是外部节点和内部底层最后一个赢者比
            l=thewin[2*treeindex];
            r=theplayer;
        }
        else{
            l=2*treeindex-n+1+lowext;  //两个都是>lowext 的成员
            r=l+1;
        }
    }
    if(theplayer==tree[0]){//重构的点是全赢者
        for(;treeindex>=1;treeindex/=2){
            int oldloser=tree[treeindex]; //保存一下之前的输者，与新值比较
            tree[treeindex]=loser(oldloser,theplayer);
            thewin[treeindex]=winner(oldloser,theplayer);
            theplayer=thewin[treeindex];  //theplayer 保存每次赢者，往上比较
        }
    }
    else{
        tree[treeindex]=loser(l,r);
        thewin[treeindex]=winner(l,r);
        if(treeindex==n-1&&n%2==1){//treeindex 是最后一层的最后一场比赛，且 n 是奇
数
            treeindex/=2;
            tree[treeindex]=loser(thewin[n-1],lowext+1);  //外部节点和内部底层最
后一个赢者比
            thewin[treeindex]=winner(thewin[n-1],lowext+1);
        }
        treeindex/=2;
        for(;treeindex>=1;treeindex/=2){  //竞赛树内部节点比较
            tree[treeindex]=loser(thewin[treeindex*2],thewin[treeindex*2+1]);
            thewin[treeindex]=winner(thewin[treeindex*2],thewin[treeindex*2+1]);
        }
    }
    tree[0]=thewin[1];
}

struct seq{  //顺串数结构体
    int num,key;  //顺串号，关键字
    bool operator<=(seq &p){   //num 优先，比较顺串大小
        if(num<p.num) return true;
        else if(num==p.num){
```

```
            if(key<=p.key) return true;
        }
        return false;
    }
};

struct filedisk{ //大磁盘[文件]
    string path; //路径
    int nvisit,sumall,cachesize,numofdisk;//访问次数，数字总个数，缓冲区大小，生成文件个数
    int k;//要求归并数
};

void init(filedisk &fi){  //初始化外排序数据，即生成一个较长文件作为外排序初始数据
    cout<<"size of cache:"<<endl; //设置缓冲区大小
    cin>>fi.cachesize;
    cout<<"initial number of disk:"<<endl;  //初始数据个数
    cin>>fi.sumall;
    cout<<"number of merge:"<<endl;
    cin>>fi.k;
    fi.path="disk.txt";   //保存到 disk.txt 文件中
    fi.nvisit=0;
    ofstream fileout(fi.path);
    if(!fileout.is_open()){   //确保文件正确打开
        cout<<"error open file"<<endl;
        exit(0);
    }
    srand((unsigned)time(NULL));  //生成随机数种子
    for(int i=0;i<fi.sumall;i++){ //生成随机数
        int x=rand()%1000+1;
        fileout<<x<<' ';          //输出到文件
    }
    fileout.close();
}

void make_seq(filedisk &fi, seq *theseq){  //生成最初归并串
    ifstream filein(fi.path);  //从初始大文件输入
    if(!filein.is_open()){
        cout<<"erroe open file"<<endl;
        exit(0);
    }
    for(int i=1;i<=fi.cachesize;i++){//把前 size 个数生成初始串
        filein>>theseq[i].key;
        theseq[i].num=1;  //设初始顺串号为 1
        fi.nvisit++;  //每从文件输入一个数，文件访问次数++
```

```
    }
    losertree<seq> seqtree(theseq,fi.cachesize); //构建初始顺串输者树
    int nn=0;
    for(int i=1;i<=fi.sumall;i++){ //从大磁盘中不断取数
        if(!(filein>>nn)) nn=INT_MIN;//文件所有数都输入完了先把 nn 变为最小值
        else fi.nvisit++;             //输出一个数就要访问一次磁盘
        seq thewinner=theseq[seqtree.getwin()]; //找到最小值顺串下标
        seq newseq;
        newseq.key=nn;                //文件中下一个数代替最小顺串的位置
        if(nn>=thewinner.key) newseq.num=thewinner.num;  //如果新值大于之前的最
小值，则新值对应的顺串号相同
        else{
            if(nn==INT_MIN){
                newseq.key=INT_MAX;
                newseq.num=INT_MAX; //把最大值放入输者树，使其最后一个赢，标记过
程结束
            }
            else  newseq.num=thewinner.num+1; //否则新值顺串号+1
        }
        seqtree.replay(seqtree.getwin(),newseq); //新树重构
        string sdisk="disk"+to_string(thewinner.num)+".txt";
        fi.numofdisk=max(fi.numofdisk,thewinner.num);//更新磁盘块数
        ofstream fileout(sdisk,ios::app);//追加模式，在末尾添加
        fileout<<thewinner.key<<' ';     //输出最小值到对应顺串磁盘块里
        fileout.close();
        fi.nvisit++;                     //每次输出访问次数++
    }
    cout<<"the sum of seq:"<<fi.numofdisk<<endl;
}

void merge_k(filedisk &fi,int k,int nnn,int diskn){//k 路归并, 文件起始号 diskn，
要输出的归并文件起始号
    string toname="disk"+to_string(nnn)+".txt";
    ofstream fileout(toname);
    queue<int> cache[k+2];
    if(k==1){
        string sfile="disk"+to_string(diskn)+".txt";
        ifstream filein(sfile);
        int temp;
        while(filein>>temp){
            for(int i=0;i<fi.cachesize;i++){
                fileout<<temp<<' ';
                if(!(filein>>temp)) break;
            }
            fi.nvisit+=2;
        }
```

```
        fileout.close();
        filein.close();
        return ;
    }
    int theplayer[k+5];
    int sp[k+5];                //存储每个小文件当前数字读取位置
    for(int i=1;i<=k;i++){      //初始化 theplayer 为各个顺串最小值
        string sfile="disk"+to_string(diskn+i-1)+".txt";
        ifstream filein(sfile);
        for(int j=0;j<fi.cachesize;j++){  //把 k 个缓冲区读满数据
            int t;
            filein>>t;
            cache[i].push(t);
        }
        fi.nvisit++;
        theplayer[i]=cache[i].front();
        cache[i].pop();
        sp[i]=filein.tellg();              //记录读取位置
        filein.close();
    }
    losertree<int> disktree(theplayer,k); //新的磁盘数最小输者树
    while(1){  //直到赢者为正无穷合并结束
        int temp=disktree.getwin();  //temp 为赢者下标，即对应顺串号
        int thewinner=theplayer[temp];
        if(thewinner==INT_MAX) break;
        if(cache[k+1].size()>=fi.cachesize){//输出缓冲区满，把缓冲区数据输出到文
件
            int t;
            while(!cache[k+1].empty()){
                t=cache[k+1].front();
                cache[k+1].pop();
                fileout<<t<<' ';
            }
            fi.nvisit++;
        }
        cache[k+1].push(thewinner);      //赢者输出到最终文件缓冲区
        int nn;
        if(cache[temp].empty()){         //对应文件的输入缓冲区空了，再次读取数据
            string name1="disk"+to_string(diskn+temp-1)+".txt";
            ifstream filein(name1);
            filein.seekg(sp[temp]+1);//移动指针位置到 sp 之前存的位置的下一个，跳
过空格
            for(int j=0;j<fi.cachesize;j++){
                int t;
                if(filein>>t){
                    cache[temp].push(t);
```

```
                }
                else{
                    cache[temp].push(INT_MAX); //下一位置为空，则该文件全部读取完
毕，设最大值
                    break;
                }
            }
            fi.nvisit++;
            sp[temp]=filein.tellg();
            filein.close();
        }
        nn=cache[temp].front();  //新的代替赢者的即同一磁盘块缓冲区中的次小值
        cache[temp].pop();
        disktree.replay(temp,nn);
    }
    int t;
    if(!cache[k+1].empty()) fi.nvisit++;
    while(!cache[k+1].empty()){
        t=cache[k+1].front();
        cache[k+1].pop();
        fileout<<t<<' ';
    }
    fileout.close();
}

void merge_seq(filedisk &fi){   //合并顺串
    int nnn=fi.numofdisk+1;     //已有文件数
    int next=1;                 //要归并的起始位置
    while(next+fi.k<=nnn){      //k 路归并
        merge_k(fi,fi.k,nnn,next);
        nnn++;
        next+=fi.k;
    }
    if(nnn-next>1) merge_k(fi,nnn-next,nnn,next); //不足 k 路且不是一路，归并
    cout<<"visit disk:"<<fi.nvisit<<endl;
}

int main(){
    filedisk thefile;
    init(thefile);
    seq theplayer[thefile.sumall+5];
    make_seq(thefile,theplayer);
    merge_seq(thefile);
}
```