page1image45511024

Complexity & Algorithm

Assignment 2: Replacement selectionpage1image45507776

Group 47:

Linh Hoang- 467424

Nam Trinh – 464560

Table of Contents

[1. System design 3](#_Toc27596000)

[1.1 Input file 3](#_Toc27596001)

[1.2 Memory length 3](#_Toc27596002)

[1.3 Min heap 3](#_Toc27596003)

[1.4 Deadspace 3](#_Toc27596004)

[1.5 Class diagram: 4](#_Toc27596005)

[1.6 Flow chart: 5](#_Toc27596006)

[6](#_Toc27596007)

[2. Result reflection: 7](#_Toc27596008)

[Result of IN = 10, M = 5 => Expected runs: 1 7](#_Toc27596009)

[Result of IN = 20, M = 5 => Expected runs: 2 8](#_Toc27596010)

[Result of IN = 1000, M = 15 => Expected runs: 33 9](#_Toc27596011)

# **1. System design**

Based on the assignment description, the replacement selection is an optimal external sorting algorithm. Without using it, an input file with N elements and a heap size H would give N/H runs with average length of H. By implementing dynamic dead space heap algorithm, the number of runs will be expected as N/2H with an average length of 2\*H.

The replacement selection algorithm requires:

+ An input file contains N elements

+ A memory length of M.

+ A min heap with a length of H.

+ A dead space length of D = M - H.

## 1.1 Input file

The input file will be created when the program starts, a number of elements is decided by the inputLength variable.

## 1.2 Memory length

The memory length is a simulated number, controlled by the memoryLength variable.

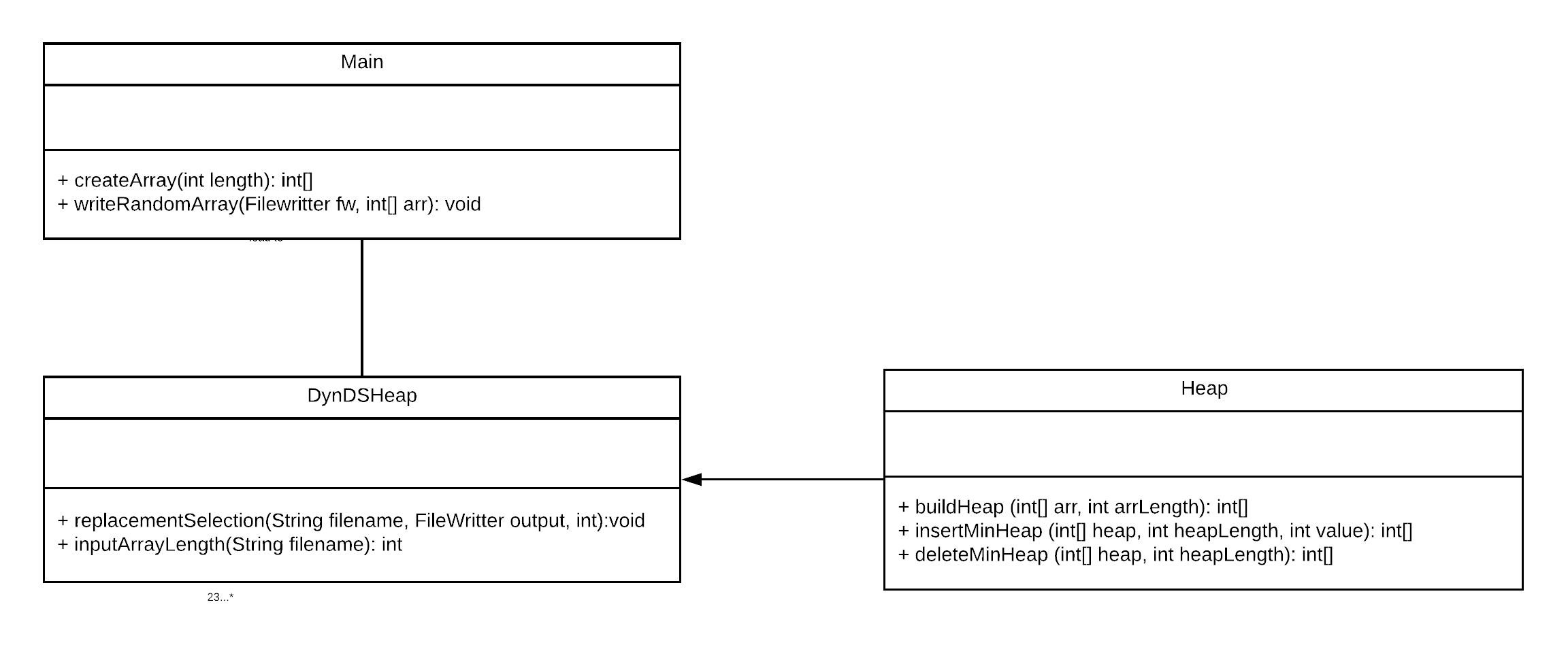
## 1.3 Min heap

Because Java does not implement a heap as a package default data structure, a heap class with function of a heap (heapify, insert, delete) need to be created. We choose to use array to represent a min heap, applied the rules of a heap to make it become a heap.

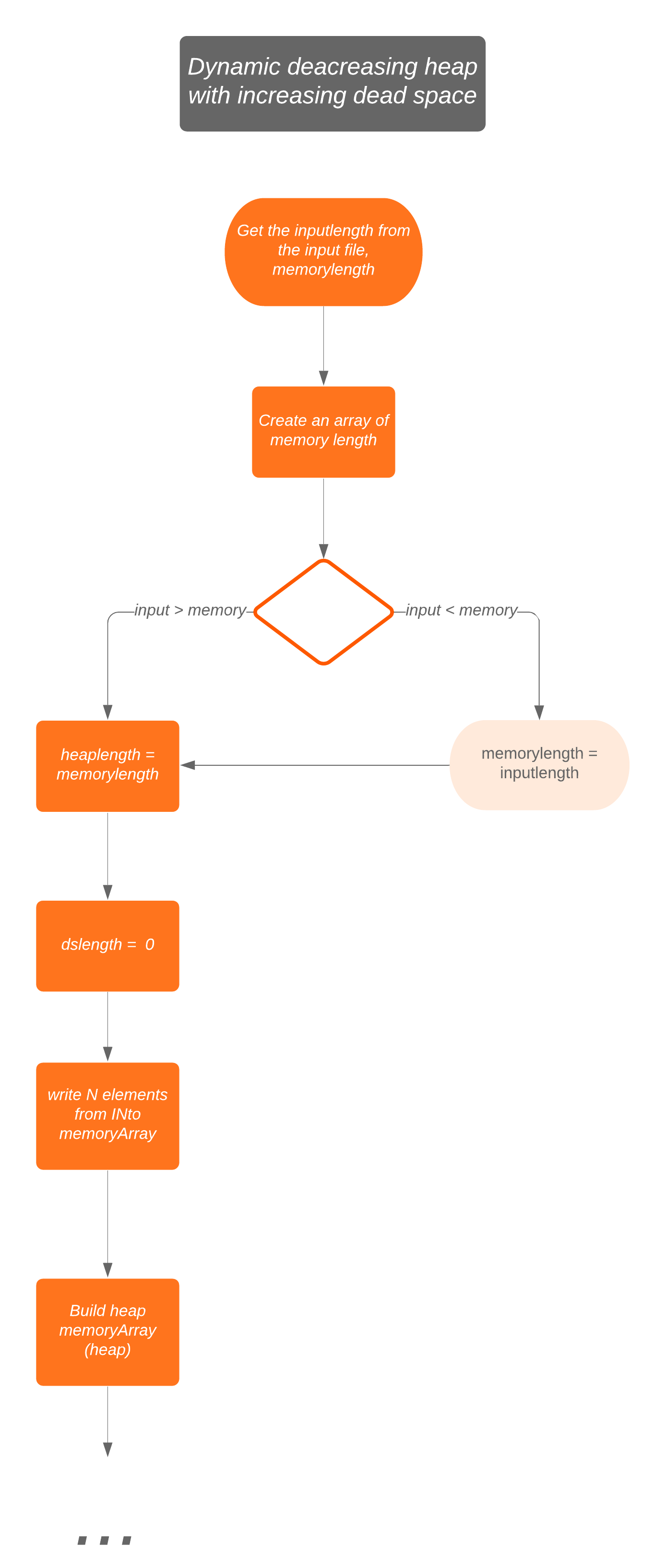
## 1.4 Deadspace

An array contains input that is smaller than the current smallest elements of the min heap. The size of deadspace increase by 1 and the size of min heap will decrease by 1 each time this situation happened.

## 1.5 Class diagram:



## 1.6 Flow chart:



# 

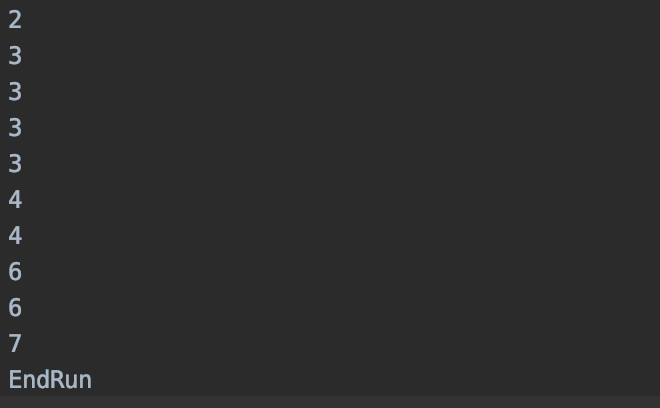
# **2. Result reflection:**

### Result of IN = 10, M = 5 => Expected runs: 1

Input file:



Output file:



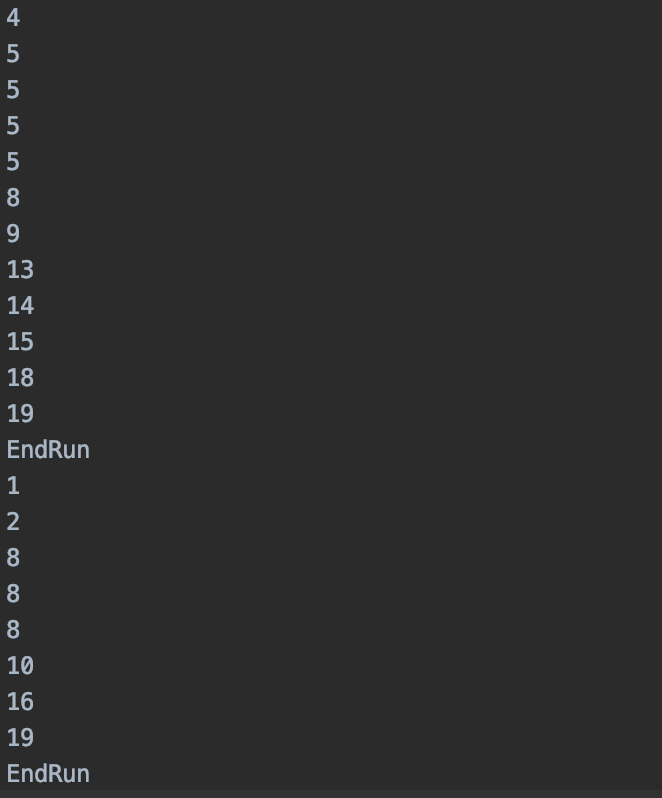
The number of runs is as expected, each run is sorted.

### Result of IN = 20, M = 5 => Expected runs: 2

Input file:



Output file:



The number of runs is as expected, each run is sorted.

### Result of IN = 1000, M = 15 => Expected runs: 33

Input file:

508 219 989 409 266 594 325 368 326 583 249 463 436 339 997 248 257 818 158 34 659 662 852 356 776 64 330 82 403 632 466 628 457 360 673 382 484 239 406 827 454 665 566 60 341 809 797 357 359 302 868 200 30 138 109 709 673 478 93 722 305 123 614 894 401 787 314 17 420 931 187 461 536 863 604 55 534 986 405 939 414 589 95 945 839 314 144 817 208 519 957 166 794 26 85 393 69 433 874 934 104 976 856 97 648 175 824 444 302 888 729 140 31 191 242 949 86 468 66 578 317 422 865 897 765 303 965 144 349 466 847 608 473 579 789 443 365 166 940 198 1 897 521 363 554 40 820 748 390 721 137 787 783 207 866 908 671 263 440 976 994 719 973 439 357 806 43 192 30 461 193 397 599 784 446 434 884 39 293 937 101 134 399 371 636 293 528 959 197 710 982 819 576 367 212 76 712 792 110 372 270 490 887 611 480 987 394 21 563 869 171 609 654 38 547 859 791 366 588 88 186 343 788 239 690 863 961 984 108 209 920 949 761 480 172 533 487 129 41 289 101 48 357 58 184 621 57 118 940 503 864 369 805 408 470 234 404 114 527 267 461 976 92 679 247 697 72 870 220 801 667 826 556 231 739 720 387 770 563 322 695 534 411 920 976 591 990 270 973 913 842 744 21 216 428 216 804 940 774 466 588 576 216 801 363 770 37 370 627 153 480 186 977 665 928 161 43 175 853 575 781 623 437 260 608 476 675 454 359 286 153 696 602 293 612 382 410 518 899 572 299 557 881 440 660 340 413 327 696 435 541 673 555 102 660 269 95 182 518 140 457 80 561 93 425 576 318 401 373 887 240 371 438 477 625 559 462 367 77 222 146 287 205 181 814 888 145 646 143 354 831 721 272 182 852 234 671 609 157 267 291 147 843 242 579 950 732 737 84 279 492 231 546 189 601 843 572 628 534 764 190 570 184 447 882 455 670 876 949 162 367 133 395 992 200 286 870 644 6 119 606 957 5 575 664 291 281 825 299 758 444 812 683 173 588 725 780 514 981 335 668 864 920 311 189 335 610 673 429 194 916 275 126 497 395 825 807 834 28 250 115 435 689 368 563 736 499 721 642 713 850 123 75 728 85 936 778 370 518 344 956 790 717 928 653 324 774 657 990 797 548 993 255 719 748 220 5 774 68 211 342 394 479 833 562 927 239 881 449 879 116 952 594 11 831 935 245 568 753 861 366 800 406 718 763 999 486 438 397 704 642 687 224 378 487 268 208 814 933 923 494 782 185 63 37 917 469 510 407 677 842 603 892 251 217 169 453 677 122 692 581 60 514 706 610 705 442 832 580 202 204 158 299 946 189 597 893 379 509 703 725 283 111 930 256 746 82 709 111 101 451 829 333 951 929 493 577 632 641 156 779 884 138 665 664 240 621 529 294 674 619 130 764 326 96 529 46 720 276 113 947 740 314 687 103 777 325 797 962 873 149 986 739 408 686 423 387 843 596 833 754 315 638 945 206 420 958 739 179 342 189 603 304 617 313 174 999 187 846 989 640 95 74 317 773 106 725 472 91 796 700 677 281 142 199 76 243 469 886 807 566 672 515 145 607 993 787 584 20 296 83 417 31 657 102 777 375 361 373 335 305 808 79 874 738 655 604 692 220 537 759 398 42 51 686 980 570 293 27 169 544 139 81 10 874 607 280 256 263 138 549 674 792 316 786 574 454 607 952 942 521 966 51 353 534 758 564 423 649 289 750 963 65 392 587 124 239 924 48 798 679 754 794 214 127 785 379 209 79 835 514 921 718 744 988 698 487 906 382 899 747 529 813 850 171 327 553 655 794 10 521 728 938 906 326 901 528 268 761 697 702 350 360 908 957 124 955 904 534 593 924 230 166 21 188 888 788 28 884 382 325 434 19 38 223 941 360 320 936 208 276 878 207 674 233 835 603 490 155 393 421 299 972 181 815 159 180 877 661 850 444 573 499 926 853 21 636 385 665 363 338 498 57 219 883 423 676 20 129 478 515 300 181 537 990 247 660 909 895 184 89 133 300 841 881 620 356 156 494 714 384 138 356 781 615 755 285 152 409 144 261 341 842 481 551 754 66 691 431 234 963 813 912 164 231 240 203 543 163 239 877 382 95 138 66 13 688 286 347 433 955 474 365 571 234 769 653 477 94 840 574 779 504 725 746 387 471 622 879 367 745 769 426 90 123 655 85 261 335 828 738 197 512 348 92 763 732 486 911 715 221 619 455 800 351 409 467 867 716 239

Output file:



The number of runs is as expected, each run is sorted. Based on these results, the algorithm has proven its sufficiency.