

알고리즘

과제번호	04주차
날 짜	2018.10.04
학 번	201302395
이 름	류경빈

문제 1(2.2-2) 선택 정렬

◆ 선택 정렬 의사 코드

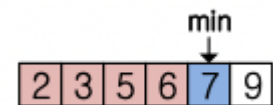
```

1  int min; // 최소 값 저장
2  int temp; // 교체 값 저장
3  selectionSort(Array[], n) {
4      for i in length(array)-1: // array 길이 -1만큼 실행
5          min = i
6          for j in (i+1 to length(array)): // i+1=j 값 과 비교
7              if array[j] < array[min] // 최소 값 과 비교
8                  min = j
9          temp = array[i]
10         array[i] = array[min]
11         array[min] = temp

```

◆ n개를 바꾸지 않고 n-1개만 바꾸면 되는 이유

SelectionSort를 진행하게 되면 n-1번째 min 값은 여태까지 앞서 정렬 된 값들보다 무조건 큰 값을 가지게 된다. 그럴 경우 n-1번째 값과 n번째 값을 비교해서 sort 해줬을 경우는 이미 n보다 앞에 있는 배열들이 모두 정렬 되었으며, n번째 값은 다른 배열의 값들보다 무조건 크기 때문이다.



◆ 최선의 경우와 최악의 경우의 수행시간 분석

선택 정렬의 경우 최선의 경우 각 순서마다 정렬되지 않은 범위에서 가장 작은 원소를 찾아 맨 앞자리 값의 자리와 변경한다. 가장 작은 값을 찾아 맨 왼쪽 원소와 변경해 오름차순으로 정렬한다. 의사코드에서 볼 수 있듯이 첫 번째 for 루프는 n-1번 반복되고, 두 번째 for 루프에서는 가장 작은 수를 찾기 위한 비교 상수 시간 작업이기 때문에

$$T(n) = (n-1) + (n-2) + \dots + 2 + 1 = n(n-1)/2 = O(n^2)$$

시간복잡도 : $O(n^2)$

문제 2(2.3-7) 버블 정렬

◆ 배열에서 두 원소의 합이 x가 되는 경우가 있는지를 알아내는 $\Theta(n \lg n)$ 시간 알고리즘 작성

```

1  array A, int x
2  A ← MergeSort(A)
3  for i ← 1 to length(A) do
4      if A[i] >= 0 and BinarySearch(A[i]-x) then
5          return true
6  end for
7      return false

```

먼저 배열 A를 Merge Sort를 진행 해 오름차순으로 배열을 정렬한다. = $O(n \lg n)$

배열 A에서 Binary Search를 통해 $A[i]-x$ 값을 찾는다. = $O(n \lg n)$

$x = A[i] + A[k]$ 일 때, $A[k] = -(A[i]-x)$ 이다.

◆ 버블 정렬 의사 코드

```

1 array A
2 BubbleSort(A[]){
3     for last ← downto 2{
4         for l ← 1 to last -1
5             if(A[l]>A[l+1] then A[l] <-> A[l+1] // 교환
6         }
7     }

```

◆ $\Theta(n \lg n)$ 시간복잡도를 가지는 이유 설명

첫 번째 for 루프는 $n-1$ 번 반복이 되고 안쪽 두 번째 for 루프는 $n-1, n-2, \dots, 2, 1$ 번 반복이 진행된다. 교환 진행은 상수시간 작업이다.

$$T(n) = (n-1) + (n-2) + \dots + 2 + 1 = n(n-1)/2 = O(n^2)$$

시간복잡도 : $O(n^2)$

문제 3 선택 정렬 구현

```
public void doSelectionSort(){
    int size = array.size();
    int min;
    int temp;

    for (int i = 0; i < size-1; i++){
        min = i;
        for (int j=i+1; j < size; j++){
            if(array.get(min) > array.get(j)){
                min = j;
            }
        }
        temp = array.get(min);
        array.set(min, array.get(i));
        array.set(i, temp);
    }
}
```

doSelectionSort() 함수 구현

```

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 32 3
3 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 6
3 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 9
3 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 11
114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 13
140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 16
2 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184
185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 20
207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228
230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 25
2 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274
275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 29
297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318
320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 3
342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364
365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386
3788 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409
410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 4
433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454
455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476
478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499
500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 5
523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544
545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566
5568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589
590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 6
613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634
635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 65
658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679
680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 70
703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724

```

doSelectionSort() 함수를 통해
data04_Sort_Sel.txt 파일 생성

문제 4 버블 정렬 구현

```
public void doBubbleSort(){
    int temp = 0;
    for (int i = array.size()-1; i>=0; i--){
        for (int j=0; j<i; j++){
            if (array.get(j) > array.get(j+1)){
                temp = array.get(j);
                array.set(j, array.get(j+1));
                array.set(j+1, temp);
            }
        }
    }
}
```

doBurbbleSort() 함수 구현

```

0      data04_Sort_Sub.txt
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,3
3,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,6
3,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,9
3,94,95,96,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,127,128,129,130,131,132,133,134,135,136,137,138,139,
140,141,142,143,144,145,146,147,148,149,150,151,152,153,154,155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,181,182,183,184,185,186,187,188,189,190,191,192,193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,
208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,227,228,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245,246,247,248,249,250,251,252,253,254,255,256,257,258,259,260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,301,302,303,304,305,306,307,308,309,310,311,312,313,314,315,316,317,318,319,320,321,322,323,324,325,326,327,328,329,330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345,346,347,348,349,350,351,352,353,354,355,356,357,358,359,360,361,362,363,364,365,366,367,368,369,370,371,372,373,374,375,376,377,378,379,380,381,382,383,384,385,386,387,388,389,390,391,392,393,394,395,396,397,398,399,400,401,402,403,404,405,406,407,408,409,410,411,412,413,414,415,416,417,418,419,420,421,422,423,424,425,426,427,428,429,430,431,432,433,434,335,436,437,438,439,440,441,442,443,444,445,446,447,448,449,450,451,452,453,454,455,456,457,458,459,460,461,462,463,464,465,466,467,468,469,470,471,472,473,474,475,476,477,478,479,480,481,482,483,484,485,486,487,488,489,490,491,492,493,494,495,496,497,498,499,500,501,502,503,504,505,506,507,508,509,510,511,512,513,514,515,516,517,518,519,520,521,522,523,524,525,526,527,528,529,530,531,532,533,534,535,536,537,538,539,540,541,542,543,544,545,546,547,548,549,550,551,552,553,554,555,556,557,558,559,560,561,562,563,564,565,566,567,568,569,570,571,572,573,574,575,576,577,578,579,580,581,582,583,584,585,586,587,588,589,590,591,592,593,594,595,596,597,598,599,600,601,602,603,604,605,606,607,608,609,610,611,612,613,614,615,616,617,618,619,620,621,622,623,624,625,626,627,628,629,630,631,632,633,634,635,636,637,638,639,640,641,642,643,644,645,646,647,648,649,650,651,652,653,654,655,656,657,658,659,660,661,662,663,664,665,666,667,668,669,670,671,672,673,674,675,676,677,678,679,680,681,682,683,684,685,686,687,688,689,690,691,692,693,694,695,696,697,698,699,700,701,702,703,704,705,706,707,708,709,710,711,712,713,714,715,716,717,718,719,720,721,722,723,724,725,726,727,728,729,730,731,732,733,734,735,736,737,738,739,740,741,742,743,744,745,746,747,748,749,750,751,752,753,754,755,756,757,758,759,760,761,762,763,764,765,766,767,768,769,770,771,772,773,774,775,776,777,778,779,780,781,782,783,784,785,786,787,788,789,790,791,792,793,794,795,796,797,798,799,800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,819,820,821,822,823,824,825,826,827,828,829,830,831,832,833,834,835,836,837,838,839,840,841,842,843,844,845,846,847,848,849,850,851,852,853,854,855,856,857,858,859,860,861,862,863,864,865,866,867,868,869,870,871,872,873,874,875,876,877,878,879,880,881,882,883,884,885,886,887,888,889,890,891,892,893,894,895,896,897,898,899,900,901,902,903,904,905,906,907,908,909,910,911,912,913,914,915,916,917,918,919,920,921,922,923,924,925,926,927,928,929,930,931,932,933,934,935,936,937,938,939,940,941,942,943,944,945,946,947,948,949,950,951,952,953,954,955,956,957,958,959,960,961,962,963,964,965,966,967,968,969,970,971,972,973,974,975,976,977,978,979,980,981,982,983,984,985,986,987,988,989,990,991,992,993,994,995,996,997,998,999,1000,1001,1002,1003,1004,1005,1006,1007,1008,1009,1010,1011,1012,1013,1014,1015,1016,1017,1018,1019,1020,1021,1022,1023,1024,1025,1026,1027,1028,1029,1030,1031,1032,1033,1034,1035,1036,103
```

doBubbleSort() 함수를 통해
data04 Sort Bub.txt 파일 생성