

# 시스템 프로그래밍

1분반 32203689 이호영

## [계기]

처음에 어떤 프로그램을 할지 고민하다가 자료구조 시간에 배운 정렬에 관해서 최적화를 진행해보기로 했다. 정렬은 수업 시간에 배운 버블 정렬과 퀵 정렬을 비교하기로 했다. 같은 테스트 케이스에서 정렬을 할 때 정렬 방법에 따라 실행 시간이 다르게 나온다. 정렬 방법은 결국 빠르고 효율적으로 처리하기 위한 알고리즘 최적화에 의해 구분되므로 성능 측정을 하기에 좋다고 생각하여 선정했다.

## [과정]

처음 버블 정렬과 퀵 정렬 코드를 만들 때 테스트 케이스를 10까지 주고 정렬을 시도했지만 큰 차이가 보이지 않았다. 그래서 충분한 성능 차이를 얻기 위해 테스트 케이스를 약1000으로 늘렸고 1000까지 테스트 케이스를 구하기 위해 파이썬으로 따로 테스트 케이스를 만들어주었다. 테스트 케이스를 정렬 코드 안에 집어넣어 주고나서 배열의 크기만큼 for문을 돌리면서 정렬을 진행한다. 퀵 정렬의 메인 함수는 함수 호출 부분을 제외하곤 동일하다

```
1 import random
2 array = []
3
4 for i in range(1000):
5     a = random.randint(1,1000)
6
7     if a in array:
8         i -=1
9     else:
10        array.append(a)
11 print(array)
12 print(len(array))
13
```

PROBLEMS 터미널 OUTPUT 디버그 콘솔 Python Debug Console + - []

c:\Users\user\.vscode\extensions\ms-python.python-2021.12.1559732655\pythonFiles\lib\python\debugpy\launcher '60340' '--' 'c:\Users\user\OneDrive\바탕 화면\2-2\자료\랜덤 배열.py'

[418, 833, 826, 209, 676, 8, 903, 532, 320, 623, 363, 307, 889, 17, 929, 845, 612, 892, 646, 806, 958, 263, 113, 603, 311, 373, 580, 537, 359, 496, 184, 246, 719, 50, 310, 259, 988, 959, 202, 471, 10, 710, 455, 124, 742, 264, 111, 220, 94, 511, 473, 44, 405, 943, 931, 227, 20, 921, 713, 793, 872, 450, 717, 647, 154, 825, 392, 499, 551, 145, 789, 390, 755, 954, 59, 218, 219, 78, 815, 224, 645, 693, 589, 309, 238, 916, 426, 272, 102, 794, 985, 656, 205, 636, 982, 177, 291, 716, 610, 771, 624, 661, 156, 531, 43, 8, 25, 404, 552, 332, 121, 631, 970, 436, 953, 900, 235, 331, 257, 298, 856, 493, 538, 926, 199, 26, 414, 91, 653, 659, 733, 891, 578, 36, 324, 655, 267, 47, 117, 727, 933, 767, 672, 924, 89, 472, 213, 837, 602, 747, 761, 669, 773, 569, 781, 97, 129, 33, 230, 34, 618, 464, 512, 786, 13, 561, 342, 728, 388, 338, 358, 181, 763, 171, 605, 4, 9, 83, 158, 777, 193, 119, 895, 98, 896, 501, 103, 144, 188, 957, 9, 207, 137, 200, 684, 533, 770, 563, 54, 382, 424, 504, 416, 494, 675, 204, 198, 71, 915, 973, 685, 58, 232, 347, 413, 394, 138, 567, 861, 315, 370, 936, 910, 969, 478, 421, 29, 961, 995, 992, 513, 155, 778, 371, 816, 305, 888, 521, 440, 15, 811, 313, 565, 628, 39, 583, 46, 304, 681, 674, 964, 616, 757, 445, 84, 449, 960, 51, 395, 790, 608, 428, 529, 190, 393, 229, 886, 818, 410, 402, 43, 201, 173, 487, 600, 223, 1000, 788, 384, 206, 939, 510, 3, 80, 281, 498, 182, 152, 317, 984, 343, 571, 927, 847, 756, 848, 114, 652, 268, 166, 420, 651, 415, 352, 253, 630, 62, 524, 760, 850, 941, 579, 654, 176, 775, 21, 633, 863, 476, 791, 695, 966, 431, 149, 497, 735, 277, 271, 157, 691, 112, 557, 350, 704, 151, 649, 640, 228, 174, 919, 686, 514, 718, 503, 87, 987, 239, 364, 88, 804, 922, 904, 31, 9, 56, 82, 620, 829, 460, 678, 549, 19, 131, 759, 453, 836, 544, 147, 104, 186, 817, 183, 422, 66, 93, 692, 306, 687, 457, 354, 648, 951, 680, 560, 587, 609, 897, 430, 878, 180, 275, 72, 204, 143, 454, 908, 871, 254, 843, 824, 604, 899, 270, 329, 133, 85, 353, 703, 779, 322, 53, 463, 568, 79, 465, 24, 483, 808, 877, 356, 475, 258, 150, 135, 240, 955, 584, 286, 782, 351, 615, 214, 130, 614, 461, 724, 90, 950, 444, 302, 212, 663, 750, 123, 751, 387, 389, 930, 105, 797, 408, 705, 368, 780, 81, 730, 813, 880, 12, 0, 880, 203, 427, 846, 865, 163, 167, 127, 467, 700, 743, 970, 682, 677, 482, 295, 667, 80, 556, 790, 70, 68, 78, 385, 280, 702, 852, 606, 832, 901, 754, 365, 470, 369, 57, 3, 109, 517, 842, 870, 934, 400, 664, 553, 948, 136, 876, 22, 303, 679, 141, 545, 588, 670, 90, 996, 293, 574, 479, 397, 977, 429, 935, 86, 835, 165, 920, 441, 918, 208, 8, 28, 932, 276, 662, 523, 398, 255, 348, 665, 601, 191, 452, 77, 477, 451, 251, 3, 423, 774, 554, 485, 528, 946, 978, 502, 722, 855, 760, 962, 185, 357, 666, 139, 945, 522, 122, 570, 898, 701, 326, 925, 197, 346, 443, 297, 762, 994, 660, 893, 879, 582, 484, 7, 572, 69, 269, 642, 432, 566, 288, 590, 164, 740, 738, 906, 999, 632, 508, 409, 725, 250, 575, 221, 37, 911, 641, 690, 990, 595, 153, 527, 802, 699, 819, 831, 712, 885, 194, 146, 287, 942, 812, 540, 361, 296, 160, 626, 536, 159, 64, 581, 32, 688, 671, 627, 764, 650, 721, 126]

633

PS C:\Users\user\OneDrive\바탕 화면\2-2\자료>

```

6 void bubble_sort(int arr[], int count) // 매개 변수로 정렬할 배열과 요소의 개수를 받음
7 {
8     int temp;
9
10    for (int i = 0; i < count; i++) // 요소의 개수만큼 반복
11    {
12        for (int j = 0; j < count - 1; j++) // 요소의 개수 - 1만큼 반복
13        {
14            if (arr[j] > arr[j + 1]) // 현재 요소의 값과 다음 요소의 값을 비교하여
15                // 큰 값을
16                temp = arr[j];
17                arr[j] = arr[j + 1];
18                arr[j + 1] = temp; // 다음 요소로 보냄
19        }
20    }
21 }
22
23
24 int main()
25 {
26
27     int numArr[633] = { 418, 833, 826, 209, 676, 8, 903, 532, 320, 623, 363, 307, 889, 17,
551, 145, 789, 390, 755, 954, 59, 218, 219, 78, 815, 224, 645, 693, 589, 309, 238, 916, 4
767, 672, 924, 89, 472, 213, 837, 602, 747, 761, 669, 773, 569, 781, 97, 129, 33, 230, 34
138, 567, 861, 315, 370, 936, 910, 969, 478, 421, 29, 961, 995, 992, 513, 155, 778, 371,
984, 343, 571, 927, 847, 756, 848, 114, 652, 268, 166, 420, 651, 415, 352, 253, 630, 62,
131, 759, 453, 836, 544, 147, 104, 186, 817, 183, 422, 66, 93, 692, 306, 687, 457, 354, 6
461, 724, 99, 950, 444, 302, 212, 663, 750, 123, 751, 387, 389, 930, 105, 797, 408, 705,
303, 679, 141, 545, 588, 670, 90, 996, 293, 574, 479, 397, 977, 429, 935, 86, 835, 165, 9
122, 570, 898, 701, 326, 925, 197, 346, 443, 297, 762, 994, 660, 893, 879, 582, 484, 7, 5
721, 126 }; // 정렬되지 않은 배열
29     int i, loop = 0;
30     struct timeval stime, etime, gap;
31     gettimeofday(&stime, NULL);
32     // 정렬할 배열, 요소 개수, 요소 크기, 비교 함수를 넣어줌
33     bubble_sort(numArr, sizeof(numArr) / sizeof(int)); // 거품 정렬 함수 호출
34
35     for (int i = 0; i < 633; i++)
36     {
37         printf("%d ", numArr[i]); // 1 2 3 4 5 6 7 8 9 10
38     }
39
40     printf("\n");
41
42     gettimeofday(&etime, NULL);
43     gap.tv_sec = etime.tv_sec - stime.tv_sec;
44     gap.tv_usec = etime.tv_usec - stime.tv_usec;
45     if (gap.tv_usec < 0) {
46         gap.tv_sec = (gap.tv_sec - 1);
47         gap.tv_usec = (gap.tv_usec + 1000000);
48     }
49     printf("Elapsed time %ldsec :%ldusec\n", gap.tv_sec, gap.tv_usec);
50
51     return 0;
52 }

```

```

1  #include <stdio.h>
2  #include <stdlib.h> // qsort 함수가 선언된 헤더 파일
3  #include <unistd.h>
4  #include <sys/time.h>
5
6  void quicksort(int *arr, int start, int end){
7      // 분할된 원소가 0개이거나 1개 일때까지 함수 호출
8      if(start >= end){
9          return;
10     }
11
12     int pivot = start; // 피벗은 첫 번째 원소
13     int i = pivot + 1; // i는 피벗 다음 원소
14     int j = end; // j는 마지막 원소
15     int temp;
16
17     while(i <= j){
18         // 피벗 보다 큰 값 만날 때 까지
19         while(i <= end && arr[i] <= arr[pivot]){
20             ++i;
21         }
22         // 피벗 보다 작은 값 만날 때 까지
23         while(j > start && arr[j] >= arr[pivot]){
24             --j;
25         }
26
27         // 위에서 계산된 i와 j가 만나거나 엇갈리면 종료
28         if(i >= j) break;
29
30         // 두 원소 교환
31         temp = arr[i];
32         arr[i] = arr[j];
33         arr[j] = temp;
34     }
35
36     // 피벗 정렬 완료
37     temp = arr[j];
38     arr[j] = arr[pivot];
39     arr[pivot] = temp;
40
41     quicksort(arr, start, j-1); // 피벗 중심으로 왼쪽부분 분할
42     quicksort(arr, j+1, end); // 피벗 중심으로 오른쪽부분 분할
43
44 }

```

## [결과 및 느낀점]

성능 측정 결과, 버블 정렬은 평균적으로 4200~4300usec가 소요됐고 퀵 정렬은 400~410usec 정도가 소요됐다. 이렇게 알고리즘만 조금 바꿨어도 엄청난 성능 차이가 난다는 것을 알 수 있다. 버블 정렬은 처음부터 끝까지 요소를 순회하면서 모든 요소를 비교하고 현재 값과 다음 값을 비교하여 큰 값을 다음으로 보내는 방법을 사용한다. 그렇기 때문에 평균적으로  $O(n^2)$ 이라는 시간복잡도를 갖는다. 퀵 정렬은 기준 키를 기준으로 큰 숫자와 작은 숫자를 구분한다. 기준키를 기준으로 작거나 같은 값을 가지는 데이터는 앞으로, 큰 값을 가지는 데이터는 뒤로 가도록 하여 작은 값을 갖는 데이터와 큰 값을 갖는 데이터로 분리해가며 정렬하는 방법이다. 그렇기 때문에 평균적으로  $O(n\log n)$ 이라는 값을 갖는다. 수업 때 for문의 loop값만을 다르게 돌려서 성능 차이가 크게 와닿지 않았었는데, 직접 배운 알고리즘을 이용해서 성능 테스트를 시도해보니까 같은 테스트 코드여도 큰 성능 차이가 있다는 것을 느껴서 매우 뿌듯했다. 이번 과제를 통해 코드 최적화의 중요성을 알게 됐고 좋은 알고리즘이란 무엇인가에 대해 생각해보게 되었다. 그래서 앞으로 코딩 테스트나 알고리즘 문제를 풀 때 최고의 효율을 낼 수 있는 코드를 구성해야겠다고 생각했다.

```
sys32203689@embedded:~/homework$ vi bsort.c
sys32203689@embedded:~/homework$ gcc -o bsort bsort.c
sys32203689@embedded:~/homework$ ./bsort
3 4 7 8 9 10 13 15 17 19 20 21 22 24 25 26 28 29 32 33 34 36 37 39 43 44 46 47 50 51 53 54 56 58 59 62 64 66 68 69 70 71 72 77 78 79 80 81 82 84 85 86 87 88 89
104 105 109 111 112 113 114 117 119 120 121 122 123 124 126 127 129 130 131 133 135 136 137 138 139 141 143 144 145 146 147 149 150 151 152 153 154 155 156 157
67 171 173 174 176 177 180 181 182 183 184 185 186 188 190 191 193 194 197 198 199 200 201 202 203 204 205 206 207 208 209 212 213 214 218 219 220 221 223 224
9 240 246 250 251 253 254 255 257 258 259 263 264 267 268 269 270 271 272 275 276 277 281 286 287 288 289 291 293 294 295 296 297 298 302 303 304 305 306 307 3
322 324 326 329 331 332 338 342 343 346 347 348 350 351 352 353 354 356 357 358 359 361 363 364 365 368 369 370 371 373 380 382 384 385 387 388 389 390 392 39
405 408 409 410 413 414 415 416 418 420 421 422 423 424 426 427 428 429 430 431 432 436 438 440 441 443 444 445 449 450 451 452 453 454 455 457 460 461 463 464
76 477 478 479 482 483 484 485 487 493 494 496 497 498 499 501 502 503 504 508 510 511 512 513 514 517 521 522 523 524 527 528 529 531 532 533 536 537 538 540
6 557 560 561 563 565 566 567 568 569 570 571 572 573 574 575 578 579 580 581 582 583 584 587 588 589 590 595 600 601 602 603 604 605 606 608 609 610 612 614 6
628 630 631 632 633 636 640 641 642 645 646 647 648 649 650 651 652 653 654 655 656 659 660 661 662 663 664 665 666 667 669 670 671 672 674 675 676 677 678 67
688 690 691 692 693 695 699 700 701 702 703 704 705 710 712 713 716 717 718 719 721 722 724 725 727 728 733 735 738 739 740 742 743 747 750 751 754 755 756 757
69 770 771 773 774 775 777 778 779 780 781 782 786 788 789 790 791 793 794 797 799 802 804 806 808 811 812 813 815 816 817 818 819 824 825 826 828 829 831 832
6 847 848 850 852 855 856 861 863 865 870 871 872 876 877 878 879 880 885 886 888 889 890 891 892 893 895 896 897 898 899 900 901 903 904 906 908 910 911 915 9
926 927 929 930 931 932 933 934 935 936 939 941 942 943 945 946 948 950 951 953 954 955 957 958 959 960 961 962 964 966 969 970 973 977 978 979 982 983 984 98
999 1000
Elapsed time 0sec :4317usec
sys32203689@embedded:~/homework$ ./qsort
3 4 7 8 9 10 13 15 17 19 20 21 22 24 25 26 28 29 32 33 34 36 37 39 43 44 46 47 50 51 53 54 56 58 59 62 64 66 68 69 70 71 72 77 78 79 80 81 82 84 85 86 87 88 89
104 105 109 111 112 113 114 117 119 120 121 122 123 124 126 127 129 130 131 133 135 136 137 138 139 141 143 144 145 146 147 149 150 151 152 153 154 155 156 157
67 171 173 174 176 177 180 181 182 183 184 185 186 188 190 191 193 194 197 198 199 200 201 202 203 204 205 206 207 208 209 212 213 214 218 219 220 221 223 224
9 240 246 250 251 253 254 255 257 258 259 263 264 267 268 269 270 271 272 275 276 277 281 286 287 288 289 291 293 294 295 296 297 298 302 303 304 305 306 307 3
322 324 326 329 331 332 338 342 343 346 347 348 350 351 352 353 354 356 357 358 359 361 363 364 365 368 369 370 371 373 380 382 384 385 387 388 389 390 392 39
405 408 409 410 413 414 415 416 418 420 421 422 423 424 426 427 428 429 430 431 432 436 438 440 441 443 444 445 449 450 451 452 453 454 455 457 460 461 463 464
76 477 478 479 482 483 484 485 487 493 494 496 497 498 499 501 502 503 504 508 510 511 512 513 514 517 521 522 523 524 527 528 529 531 532 533 536 537 538 540
6 557 560 561 563 565 566 567 568 569 570 571 572 573 574 575 578 579 580 581 582 583 584 587 588 589 590 595 600 601 602 603 604 605 606 608 609 610 612 614 6
628 630 631 632 633 636 640 641 642 645 646 647 648 649 650 651 652 653 654 655 656 659 660 661 662 663 664 665 666 667 669 670 671 672 674 675 676 677 678 67
688 690 691 692 693 695 699 700 701 702 703 704 705 710 712 713 716 717 718 719 721 722 724 725 727 728 733 735 738 739 740 742 743 747 750 751 754 755 756 757
69 770 771 773 774 775 777 778 779 780 781 782 786 788 789 790 791 793 794 797 799 802 804 806 808 811 812 813 815 816 817 818 819 824 825 826 828 829 831 832
6 847 848 850 852 855 856 861 863 865 870 871 872 876 877 878 879 880 885 886 888 889 890 891 892 893 895 896 897 898 899 900 901 903 904 906 908 910 911 915 9
926 927 929 930 931 932 933 934 935 936 939 941 942 943 945 946 948 950 951 953 954 955 957 958 959 960 961 962 964 966 969 970 973 977 978 979 982 983 984 98
999 1000
Elapsed time 0sec :405usec
sys32203689@embedded:~/homework$
```