

Input size = 100

```
Microsoft Visual Studio Debu  x  +  v  -  □  x
Please input the heap_size of the array:100
6 35 35 37 40 41 82 84 101 106 118 141 145 153 169 190 253 264 281 288 292 299 308 316 322 323 333 334 350 358 370 376 3
82 391 393 421 436 439 446 447 464 467 478 491 500 529 537 538 538 541 547 548 604 623 626 629 644 648 662 664 667 673 7
03 705 711 716 718 723 724 726 729 741 756 757 771 778 805 811 827 827 840 842 859 868 869 890 894 895 902 912 929 931 9
42 942 944 954 961 962 966 995
Execution time: 12 microseconds

D:\Algorithm\HW4\quickSort\x64\Debug\quickSort.exe (process 9524) exited with code 0.
Press any key to close this window . . .|
```

Time duration: 12 microseconds

Input size = 1,000

```
Microsoft Visual Studio Debu  x  +  v  -  □  x
9 230 232 233 234 235 240 240 245 247 249 249 253 253 255 256 256 257 258 259 260 261 262 262 263 264 264 264 264 264 26
9 270 270 270 271 272 279 279 281 281 281 282 282 285 285 286 286 287 287 288 288 289 290 291 292 292 292 292 295 296 29
6 297 297 299 300 302 303 303 303 303 303 306 308 309 309 310 313 313 313 313 313 314 314 314 315 316 316 317 318 318 31
8 321 322 322 323 323 324 326 328 329 330 333 333 334 334 335 337 337 342 342 343 347 348 350 350 350 350 353 353 355 35
5 355 355 356 357 357 357 358 359 359 360 361 361 362 363 363 365 368 369 369 370 371 372 374 375 376 380 382 383 384 38
6 389 391 391 392 392 393 398 401 402 404 405 409 410 410 411 411 413 413 413 414 416 416 416 418 421 421 422 422 423 42
3 423 423 423 424 425 426 426 428 428 429 430 430 432 433 434 436 437 439 439 439 441 443 446 447 448 450 451 452 454 45
5 457 457 458 459 460 461 462 463 464 464 466 466 467 467 467 472 472 474 474 476 477 477 477 478 480 481 482 483 483 48
3 484 484 484 485 485 487 487 488 488 489 489 491 492 493 496 497 498 500 503 503 503 504 505 506 508 508 510 510 511 51
1 512 512 514 515 515 518 519 519 520 520 523 525 526 527 527 529 529 532 534 535 536 537 537 538 538 538 539 540 540 54
1 541 541 542 543 543 545 546 547 547 548 548 548 549 549 549 549 550 555 555 556 556 556 556 557 558 559 561 561 565 56
5 565 565 570 573 574 576 576 577 578 580 584 584 584 585 585 587 588 588 589 589 589 591 591 593 593 595 596 596 598 60
0 600 600 601 601 601 602 604 604 605 606 607 608 608 609 610 611 616 617 617 617 618 619 619 620 622 623 624 624 625 62
5 625 625 626 626 626 626 627 627 627 629 629 629 629 629 634 634 634 635 636 637 637 637 639 641 643 644 646 646 648 648 64
8 649 650 650 651 651 652 653 654 655 655 657 658 658 659 662 662 662 663 664 667 667 668 668 670 671 673 673 674 674 67
5 676 676 678 678 678 679 681 683 685 685 686 687 688 689 690 690 692 693 694 694 695 695 696 698 699 700 701 701 702 70
3 704 704 704 705 705 705 706 710 711 711 712 712 713 716 717 718 718 721 722 723 723 724 724 725 726 726 728 729 734 73
4 734 734 736 737 740 741 741 745 745 748 748 750 752 753 753 754 756 756 757 757 757 758 758 758 759 759 760 760 762 76
3 763 763 766 767 769 771 771 773 774 775 777 778 778 781 783 783 786 786 787 788 788 789 790 790 796 798 798 798 800 80
1 802 805 807 808 811 812 813 813 814 815 815 815 818 818 823 824 824 825 825 827 827 829 829 829 831 831 832 832 833 83
3 833 835 836 838 840 841 842 843 844 844 847 848 850 850 851 851 853 855 855 855 858 859 861 864 865 866 867 868 869 86
9 869 869 869 870 870 874 875 875 877 878 881 881 882 885 886 887 888 888 890 892 893 893 894 895 896 896 898 900 900 90
0 900 901 902 902 902 903 905 909 909 911 912 912 913 913 913 923 923 924 924 924 926 928 929 930 931 932 932 932 934 93
5 936 937 938 938 938 940 941 941 942 942 943 944 944 944 945 945 945 946 948 948 949 949 951 954 954 954 955 956 958 95
8 958 959 961 961 962 962 962 962 963 964 966 966 969 970 971 971 971 972 972 974 974 975 976 977 977 982 985 985 986 98
9 990 992 993 993 994 995 996 997 998 999
Execution time: 160 microseconds
```

Time duration: 160 microseconds

Input size: 10,000

```
Microsoft Visual Studio Debu... x + -
21 922 922 922 922 922 922 922 922 922 922 923 923 923 923 923 923 923 923 923 923 924 924 924 924 9
24 924 924 924 924 925 925 925 925 925 925 925 925 925 925 925 925 925 925 925 925 926 926 926 926 9
27 927 927 927 928 928 928 928 928 928 928 928 929 929 929 929 929 929 929 929 929 930 930 930 930 9
30 930 930 931 931 931 931 931 931 932 932 932 932 932 932 932 932 932 932 932 932 933 933 933 933 9
34 934 934 934 934 934 935 935 935 935 935 935 935 935 935 935 935 935 935 935 935 936 936 936 936 9
37 937 937 937 937 937 938 938 938 938 938 938 938 938 938 938 938 938 938 938 938 939 939 939 939 9
40 941 941 941 941 941 941 941 941 941 941 941 942 942 942 942 942 942 942 942 942 943 943 943 943 9
43 943 943 943 943 943 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 945 945 945 945 9
46 946 946 946 946 946 946 946 946 946 946 946 946 946 946 946 946 946 946 946 946 947 947 947 947 9
48 948 948 948 948 949 949 949 949 949 949 949 949 949 949 949 949 949 949 949 949 950 950 950 950 9
52 952 953 953 953 953 953 953 953 953 953 953 953 953 953 953 953 953 953 953 953 954 954 954 954 9
55 955 955 955 955 955 955 955 955 955 955 955 955 955 955 955 955 955 955 955 955 956 956 956 956 9
58 958 958 958 958 958 958 958 958 958 958 958 958 958 958 958 958 958 958 958 958 959 959 959 959 9
61 961 962 962 962 962 962 962 962 962 962 962 962 962 962 962 962 962 962 962 962 963 963 963 963 9
65 965 965 965 965 966 966 966 966 966 966 966 966 966 966 966 966 966 966 966 966 966 966 966 966 9
67 967 968 968 968 968 968 968 968 968 968 968 968 968 968 968 968 968 968 968 968 969 969 969 969 9
71 971 971 971 971 971 971 971 971 971 971 971 971 971 971 971 971 971 971 971 971 972 972 972 972 9
74 974 974 974 974 974 974 974 974 974 974 974 974 974 974 974 974 974 974 974 974 975 975 975 975 9
77 977 977 977 977 977 977 977 977 977 977 977 977 977 977 977 977 977 977 977 977 978 978 978 978 9
79 979 979 979 980 980 980 980 980 980 980 980 980 980 980 980 980 980 980 980 980 981 981 981 981 9
82 982 982 982 983 983 983 983 983 983 983 983 983 983 983 983 983 983 983 983 983 984 984 984 984 9
85 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 986 986 986 986 9
87 988 988 988 988 988 988 988 988 988 988 988 988 988 988 988 988 988 988 988 988 989 989 989 989 9
90 990 990 990 990 991 991 991 991 991 991 991 991 991 991 991 991 991 991 991 991 992 992 992 992 9
93 993 993 993 993 993 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 995 995 995 995 9
97 997 997 997 997 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 999 999 999 999 9
Execution time: 2080 microseconds
```

Time duration: 2080 microseconds

Input size: 100,000

```
Microsoft Visual Studio Debu... x + -
992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992
992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992
992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992 992
993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993
993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993
993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993
993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993
993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993 993
994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994
994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994
994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994 994
995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995
995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995
995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995
995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995
996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996
996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996
996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996
997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997
997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997
997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997
997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997 997
998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998
998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998
998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998
999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999
999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999
999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999
Execution time: 26418 microseconds
```

Time duration: 26418 microseconds

將執行時間與 insertion sort、merge sort、heap sort 做比較，可以發現在較少 elements 需排序的時候( $n=100$ )，insertion sort 與 random quicksort 皆擁有最短的排序時間(12ms)，然而將 scale 拉大後( $n>100$ )，heap sort 就擁有最短的執行時間(17874ms)，大於 random quicksort(26418ms)和 mergesort(28514ms)且遠大於 insertion sort 的執行時間(2423709ms)

### 採用時機:

若需要排序的數目較少時，應該選用 insertion sort 或 random quicksort 當作排序的方法，若需排序的數目增加，不應當再採取 insertion sort，而是使用 random quicksort 或 heap sort，其中 heap sort 的排序時間又會優於 random quicksort。

### 複雜度分析:

```
int main()
{
    int arr_size;
    cout << "Please input the heap_size of the array:";
    cin >> arr_size;
    int* input = new int[arr_size];
    for (int i = 0; i < arr_size; i++)
    {
        int num = rand() % 1000;
        input[i] = num;
    }

    auto start_time = chrono::high_resolution_clock::now();
    randomQuickSort(input, 0, arr_size-1);
    auto end_time = chrono::high_resolution_clock::now();
    Printarr(input, arr_size);
    auto duration = chrono::duration_cast<chrono::microseconds>(end_time - start_time);
    cout << "Execution time: " << duration.count() << " microseconds" << endl;

    delete[] input;

    return 0;
}
```

分析 random quicksort 的時間複雜度

```
void randomQuickSort(int* arr, int left, int right) {
    if (left < right) {
        int q = randomPartition(arr, left, right);
        randomQuickSort(arr, left, q - 1);
        randomQuickSort(arr, q + 1, right);
    }
}
```

$$T(n) = O(\text{random partition}) + T(i) + T(n - i)$$

```
int randomPartition(int* arr, int p, int r) {
    int min = p;
    int max = r;
    int i = rand() % (max - min + 1) + min;
    swap(arr[r], arr[i]);
    return Partition(arr, p, r);
}
```

$O(\text{random partition}) = O(\text{swap}) + O(\text{partition})$

```
void swap(int& t1, int& t2)
{
    int temp = t1;
    t1 = t2;
    t2 = temp;
}
```

資料交換而已，所以  $O(\text{swap}) = O(1)$

```
int Partition(int *arr, int p, int r){
    int x = arr[r];
    int i = p - 1;
    for (int j = p; j < r; j++) {
        if (arr[j] <= x) {
            i = i + 1;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[r]);
    return i + 1;
}
```

$O(\text{partition}) = n * O(\text{swap}) = O(n)$

所以  $T(n) = O(n) + T(i) + T(n-i)$ ，假設  $i < \frac{n}{2}$ 。分割的遞迴樹高為  $\log_{\frac{n-i}{i}} n$ ，而相同樹高

的合皆會  $\leq O(n)$ ，也就是 random-Partition 的 complexity，故  $T(n) = O(n \log n)$ ，且使用 random-quicksort 演算法，每個 pivot 皆為 random，故不存在 best case 和 worst case 的分析，不管 input 如何，時間複雜度都會是 average case，也就是  $O(n \log n)$ 。

