19CSE302 - Design Analysis Of Algorithms
Lab Evaluation 1

Praneeth V - CB.EN.U4CSE22244
Sai Krishna - CB.EN.U4CSE22246

August 23, 2024

Lab Evaluation Questions

1. First Question Analysis of Sorting algorithms

   1.1. In-Place Quick Sort (with last element as the pivot)

   Quick sort applies the Divide-And-Conquer strategy to sort a subarray A[p..n]:

   Divide : The array is rearranged into two subarrays such that A[p .. q-1] is less than that of A[q] and the other subarray A[q+1 .. n] greater than A[q]. Here A[q] is the pivot element.

   Conquer : We sort the subarrays A[p .. q-1] and A[q+1 .. n]

   Combine: Now, we combine the sorted arrays together.

   The time complexity of the In-Place Quick Sort depends on whether the partitioning is balanced or not, and depends on which elements are used for partitioning.

   If the partitioning is balanced: It's the best case where the algorithm runs as fast as merge sort. Since the partitioning is balanced, we get two even halves where one is of the size $\lfloor n/2 \rfloor$ and the other one of the size $\lfloor n/2 \rfloor - 1$. So the recurrence relation for this becomes: $T(n) = 2T(n/2)+\Theta(n)$ and by Master's Theorem, it becomes $T(n) = \Theta(n \log n)$.

   If the partitioning is unbalanced: Its the worst case where we have one subarray with size of $n-1$ and the other subarray with the size of 0. The time complexity of the partitioning costs $\Theta(n)$ time and the recurrence relation for this case becomes: $T(n) = T(n-1) + \Theta(n)$ and making the overall time complexity $T(n) = \Theta(n^2)$ and this case mostly happens when the array is already sorted.

   If the partitioning is based on proportionality: Its the average case where the partitioning is done based on some proportions where the partitioning is like x-to-10$-x$, then in this case the recurrence relation becomes like: $T(n) = T(x \cdot n/10) + T(n/10) + c \cdot n$ in which just reaching the depth of the recursion tree for partition takes $\log_{10/x} n$ which is $\Theta(\log n)$ and the cost at each level is n making the time complexity: $T(n) = O(n \log n)$

The Space complexity for In-Place Quicksort is $O(\log n)$ for best and average case partitioning where $logn$ represents the size of the recursion tree and in the worst case partitioning the space complexity becomes $O(n)$ as the recursion tree becomes skewed leading to the size of the tree to be equal to the number of elements in an array.

`Test Case 1: For size 100:`

```
Time taken: 0.03425 ms
Memory used: 983 KB
Number of comparisons: 651
Number of swaps: 488
Number of basic operations: 781
Sorted array:
1 5 5 6 6 7 7 8 9 10 11 12 13 14 15 16 16 18 19 20 21 22 25 25 26 28 28 30 31 33 33 34 35 35 36 3
7 37 37 40 40 41 45 46 50 51 52 54 54 54 55 57 57 59 59 61 61 62 62 63 63 65 66 67 67 69 70 70 71
 72 74 74 75 76 77 77 79 82 82 83 84 84 84 84 87 88 88 89 89 89 89 90 91 91 93 93 95 95 95 97 99
%
```

`Test case 2: For size 300`

```
Time taken: 0.125084 ms
Memory used: 983 KB
Number of comparisons: 2406
Number of swaps: 1301
Number of basic operations: 2818
Sorted array:
1 2 3 6 7 8 8 9 10 10 13 14 14 15 15 19 20 21 21 22 30 31 31 32 33 34 35 35 36 36 37 37 39 39
 40 40 41 42 43 44 47 48 48 50 50 50 50 51 54 57 58 58 58 59 60 62 64 64 65 65 67 67 67 67 68
 69 69 70 71 71 71 76 78 79 86 88 89 93 93 94 96 97 98 99 99 100 100 101 101 103 105 105 106
106 107 107 107 107 109 110 110 111 112 115 115 115 116 116 117 117 117 117 117 118 121 121 1
22 123 125 125 126 126 129 130 131 133 134 137 137 137 139 141 142 146 149 151 151 153 153 15
4 157 157 158 159 159 160 160 161 161 161 161 162 162 163 164 165 165 166 168 169 169 173 173
 174 174 175 176 176 176 176 177 179 180 180 180 181 181 182 184 184 185 185 185 187 187 188
189 191 192 192 192 192 192 193 197 197 197 198 198 198 200 200 200 200 202 206 207 208 208 2
09 209 211 211 212 213 213 214 215 216 216 218 219 221 221 223 223 225 226 227 229 231 231 23
3 233 233 235 236 236 236 237 238 238 239 239 241 242 243 243 243 245 245 247 247 248 249 249
 250 250 252 254 254 254 257 258 259 260 261 263 266 266 267 268 269 269 271 272 272 272 272
273 276 276 279 280 281 282 282 283 284 285 286 289 289 291 292 293 295 297 299 300 %
```

`Test case 3: For size 500`

```
Time taken: 0.201917 ms
Memory used: 983 KB
Number of comparisons: 4573
Number of swaps: 2848
Number of basic operations: 5235
Sorted array:
4 5 5 7 8 9 9 10 10 11 13 18 19 19 19 21 21 22 25 25 28 28 28 29 29 30 31 32 32 33 33 34 35 35 37 38 39 43 44 44 44 46 48 48 50 50
 51 52 52 52 54 54 55 55 57 57 58 59 59 61 61 61 64 64 66 68 68 68 69 69 72 73 74 78 78 78 79 82 82 83 84 84 90 91 92 93 93 96 97
97 99 99 99 101 102 103 105 108 108 108 109 109 110 111 111 111 111 112 113 113 114 115 118 119 119 120 121 127 128 128 129 132 13
3 135 138 138 139 139 140 140 142 142 143 144 145 145 146 146 146 147 147 148 149 150 151 152 152 153 154 155 157 158 158 159 160
161 162 162 163 163 165 168 168 169 170 171 171 173 173 176 176 176 177 177 180 181 181 181 182 184 186 189 189 193 193 193 194 19
5 196 196 196 197 198 199 199 200 201 201 202 203 205 206 206 206 206 207 207 207 208 208 208 211 211 211 212 212 213 214 215 215
216 216 216 218 220 220 221 223 224 226 226 228 228 229 231 231 232 232 232 233 233 234 234 234 235 236 236 237 238 238 239 240 24
0 241 241 242 244 244 247 247 248 251 251 251 251 252 253 253 253 260 263 264 264 264 266 266 271 271 272 273 274 275 277 277 277
278 279 279 280 280 281 282 283 290 290 292 293 294 296 296 300 300 301 301 302 303 303 303 303 305 306 306 307 308 309 309 309 30
9 310 310 310 311 311 311 314 315 316 316 318 319 319 321 323 323 324 324 328 330 332 333 333 334 335 335 335 336 336 336 337 337
338 342 345 347 347 349 352 353 353 354 354 357 357 358 359 361 361 364 366 367 369 369 370 371 373 373 373 373 374 374 374 376 37
7 377 378 380 381 385 385 386 386 388 390 391 391 391 392 392 392 394 395 396 396 398 398 398 401 402 403 404 405 405 407 408 410
411 412 412 412 412 415 415 416 416 419 419 419 420 420 420 421 422 422 423 425 426 427 428 429 431 431 432 434 436 438 438 439 43
9 440 442 443 446 446 446 447 448 449 449 450 454 455 455 456 458 461 461 462 462 464 464 468 469 471 471 472 473 474 475 475 475
476 478 482 483 484 484 485 485 486 488 490 492 493 495 495 496 496 498 500 500 %
```

```
Test case 4: For size 1000
```

```
Time taken: 0.446833 ms
Memory used: 1966 KB
Number of comparisons: 11545
Number of swaps: 7037
Number of basic operations: 12875
Sorted array:
1 3 4 6 7 9 9 10 11 12 13 13 14 16 18 19 21 21 22 26 26 28 31 34 37 38 41 41 45 48 48 52 53 53 55 55 58 58 58 59 59 63 64 64
 66 66 67 67 71 71 71 73 76 76 77 78 78 79 79 79 80 81 81 83 85 85 85 86 87 88 91 91 92 93 94 95 97 97 97 98 99 100 101 104
104 105 106 106 107 111 111 112 115 115 116 116 116 116 116 117 119 119 122 123 123 123 124 127 128 129 130 130 132 134 134
134 136 140 143 144 144 146 147 148 148 152 154 154 155 156 158 158 158 158 159 160 160 161 161 163 165 166 166 166 168 168
170 171 172 175 176 178 179 185 185 187 187 189 190 191 192 192 193 194 195 197 200 200 201 204 209 212 213 213 214 214 215
215 216 217 218 220 220 220 221 222 223 225 229 231 231 232 233 234 236 239 239 240 240 241 243 243 245 246 247 248 248 251
251 252 252 252 253 255 255 256 256 256 257 257 258 260 260 262 263 264 265 266 266 266 266 266 266 267 267 268 268 269 271
271 271 272 272 272 273 273 274 274 275 275 275 275 275 275 277 279 280 281 283 284 285 285 287 288 289 290 290 292 292 293
293 293 295 301 301 302 303 303 304 306 306 307 309 309 311 311 312 313 314 314 315 317 317 318 319 320 320 322 322 322 325
325 326 328 328 330 333 333 334 334 337 337 337 338 338 339 339 340 340 341 342 343 343 344 346 348 349 350 350 350 351 352
353 353 354 354 356 357 357 358 358 359 359 361 364 366 366 366 369 369 371 372 373 373 373 373 378 380 380 381 382 382 382
382 383 383 384 386 386 387 387 387 388 388 389 390 391 391 391 392 395 395 398 401 402 402 403 403 404 405 405 405 407 409
409 410 411 412 412 412 415 416 418 418 419 419 419 419 420 421 424 425 426 426 426 426 427 427 428 429 431 431 432 433 433
435 438 439 439 442 442 444 444 445 446 447 447 448 448 450 450 450 451 451 451 452 452 453 457 459 461 462 462 462 462 463
465 469 469 471 472 475 476 476 478 478 479 479 479 479 480 481 481 482 482 483 485 485 487 489 489 489 489 490 490 491 491
492 494 495 495 498 500 500 501 503 503 503 503 504 505 506 507 507 507 508 511 512 513 514 515 515 515 516 517 517 518 518
519 519 522 523 523 524 525 525 527 527 528 529 529 530 530 530 532 535 537 539 541 542 542 544 544 545 545 547 547 548 550
550 551 552 553 554 556 557 558 560 561 561 562 563 563 566 566 567 571 572 572 573 576 577 578 578 579 580 584 586 588 590
591 593 594 595 596 598 599 600 600 607 608 608 610 611 611 612 612 613 615 615 616 618 618 619 621 622 622 622 623 624 624
624 624 626 627 628 628 628 629 631 631 632 634 634 634 637 637 641 641 641 642 642 644 645 645 648 648 654 654 655 657 657
657 659 661 661 662 662 662 663 663 665 665 665 666 666 667 667 668 668 668 670 672 672 673 675 676 676 677 677 677 677 678
682 683 684 685 685 687 688 688 688 688 689 690 691 693 695 695 696 697 697 698 698 699 700 700 700 700 703 703 704 704 705
706 708 709 710 711 714 714 715 715 716 717 718 718 721 725 728 728 733 733 733 733 735 735 735 735 736 737 738 738 738 738
738 738 741 742 745 746 747 747 747 749 749 751 753 755 758 759 759 759 759 762 766 766 767 767 768 769 770 770 771 773 774
774 775 776 777 777 777 779 779 782 783 783 784 784 787 787 788 788 790 790 791 792 792 794 795 796 798 801 801 802 802 804
804 804 804 808 808 809 810 810 811 813 813 815 816 817 818 818 818 819 819 820 823 823 823 824 827 830 832 833 836 836 836
837 837 840 840 841 845 846 848 849 850 850 851 852 852 852 854 858 860 861 861 862 863 863 864 865 866 871 872 874 876 876
877 877 877 877 878 878 880 882 883 884 885 885 885 886 886 888 889 890 891 894 896 896 897 899 900 901 902 902 903 903 904
904 905 905 906 906 907 907 908 909 911 912 913 913 913 914 914 917 917 917 918 920 921 921 921 922 922 926 927 929 929 930
931 931 932 932 933 934 934 936 937 937 937 938 939 939 940 942 942 943 944 944 945 946 947 948 953 955 956 958 960 960 961
962 962 963 964 965 966 967 969 969 970 972 973 973 974 974 976 976 977 977 978 978 979 980 980 980 981 981 981 981 983 984
986 986 986 986 986 987 988 989 989 989 991 993 995 995 996 999 1000 %
```

1.2. Subdivision 2

```
% Code Section 1.2.1
```

```
% Code Section 1.2.2
```

```
% Code Section 1.2.3
```

```
% Code Section 1.2.4
```

path/to/image5.png

Figure 1: Description of the image

path/to/image6.png

Figure 2: Description of the image

path/to/image7.png

Figure 3: Description of the image

path/to/image8.png

Figure 4: Description of the image