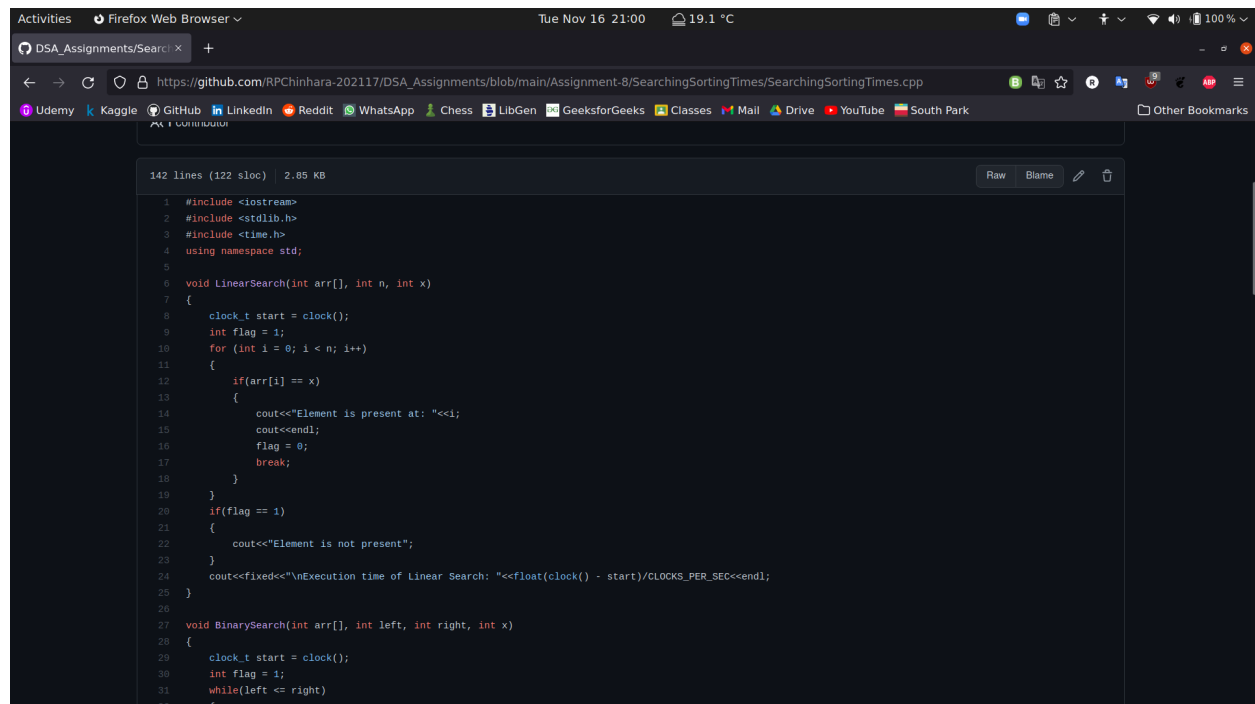


# Assignment - 8

Rudra Chinhara, 202117  
B. Tech CSE  
Central University of Haryana

Write a program to simulate the various searching and sorting algorithms and compare their timings of a list of 1000 elements

## Code:



The screenshot shows a web browser displaying a GitHub repository page for a C++ program. The page title is "SearchingSortingTimes/SearchingSortingTimes.cpp". The code is displayed in a dark-themed editor with line numbers. The code implements two functions: LinearSearch and BinarySearch, both of which take an array, its size, and a target value as input. The LinearSearch function uses a for loop to iterate through the array and a flag to track if the element is found. The BinarySearch function uses a while loop to perform a binary search. Both functions output the execution time in seconds using the clock() function and the CLOCKS\_PER\_SEC constant. The code is 142 lines long, with 122 statements and a size of 2.85 KB.

```
142 lines (122 sloc) | 2.85 KB
1 #include <iostream>
2 #include <stdlib.h>
3 #include <time.h>
4 using namespace std;
5
6 void LinearSearch(int arr[], int n, int x)
7 {
8     clock_t start = clock();
9     int flag = 1;
10    for (int i = 0; i < n; i++)
11    {
12        if(arr[i] == x)
13        {
14            cout<<"Element is present at: "<<i<<endl;
15            cout<<endl;
16            flag = 0;
17            break;
18        }
19    }
20    if(flag == 1)
21    {
22        cout<<"Element is not present";
23    }
24    cout<<fixed<<"\nExecution time of Linear Search: "<<float(clock() - start)/CLOCKS_PER_SEC<<endl;
25 }
26
27 void BinarySearch(int arr[], int left, int right, int x)
28 {
29     clock_t start = clock();
30     int flag = 1;
31     while(left <= right)
32     {
```

Activities Firefox Web Browser Tue Nov 16 21:00 19.1 °C

DSA\_Assignments/Search: x +

https://github.com/RPChinhara-202117/DSA\_Assignments/blob/main/Assignment-8/SearchingSortingTimes/SearchingSortingTimes.cpp

Udemy Kaggle GitHub LinkedIn Reddit WhatsApp Chess LibGen GeeksforGeeks Classes Mail Drive YouTube South Park Other Bookmarks

```
25 }
26
27 void BinarySearch(int arr[], int left, int right, int x)
28 {
29     clock_t start = clock();
30     int flag = 1;
31     while(left <= right)
32     {
33         int mid = left + ((right - left)/2);
34
35         if(arr[mid] == x)
36         {
37             cout<<"Element is present at: "<<mid;
38             flag = 0;
39             break;
40         }
41         if(arr[mid] < x)
42             left = mid + 1;
43         else
44             right = mid - 1;
45     }
46     if(flag == 1)
47         cout<<"Element not present";
48
49     cout<<fixed<<"\nExecution time of Binary Search: "<<float(clock() - start)/CLOCKS_PER_SEC<<endl;
50 }
51
52 void swap(int &x, int &y)
53 {
54     int temp = x;
55     x = y;
56     y = temp;
57 }
58
59 void BubbleSort(int arr[], int n)
60 {
61     clock_t start = clock();
```

Activities Firefox Web Browser Tue Nov 16 21:00 19.1 °C

DSA\_Assignments/Search: x +

https://github.com/RPChinhara-202117/DSA\_Assignments/blob/main/Assignment-8/SearchingSortingTimes/SearchingSortingTimes.cpp

Udemy Kaggle GitHub LinkedIn Reddit WhatsApp Chess LibGen GeeksforGeeks Classes Mail Drive YouTube South Park Other Bookmarks

```
59 void BubbleSort(int arr[], int n)
60 {
61     clock_t start = clock();
62     for (int i = 0; i < n-1; i++)
63     {
64         for (int j = 0; j < n-1-i; j++)
65         {
66             if(arr[j] > arr[j+1])
67             {
68                 swap(arr[j], arr[j+1]);
69             }
70         }
71     }
72     cout<<fixed<<"\nExecution time of Bubble Sort: "<<float(clock() - start)/CLOCKS_PER_SEC<<endl;
73 }
74
75 int partition(int arr[], int low, int high)
76 {
77     int pivot = arr[high];
78     int i = low - 1;
79
80     for(int j = low; j <= high - 1; j++)
81     {
82         if(arr[j] < pivot)
83         {
84             i++;
85             swap(arr[i], arr[j]);
86         }
87     }
88     swap(arr[i+1], arr[high]);
89     return i+1;
90 }
91
92 void QuickSort(int arr[], int low, int high)
93 {
94     if (low < high)
```

Activities Firefox Web Browser Tue Nov 16 21:00 19.1 °C

DSA\_Assignments/Search x +

https://github.com/RPChinhara-202117/DSA\_Assignments/blob/main/Assignment-8/SearchingSortingTimes/SearchingSortingTimes.cpp

Udemy Kaggle GitHub LinkedIn Reddit WhatsApp Chess LibGen GeeksforGeeks Classes Mail Drive YouTube South Park Other Bookmarks

```
92 void QuickSort(int arr[], int low, int high)
93 {
94     if (low < high)
95     {
96         int pi = partition(arr, low, high);
97         QuickSort(arr, low, pi - 1);
98         QuickSort(arr, pi + 1, high);
99     }
100 }
101
102 void printArray(int arr[], int size)
103 {
104     for(int i = 0; i < size; i++)
105         cout<<arr[i]<<" ";
106     cout<<endl;
107 }
108
109 int main()
110 {
111     int Array[1000], newArray[1000], element;
112     for(int i = 0; i < 1000; i++)
113     {
114         Array[i] = rand()%1000;
115     }
116     for(int i = 0; i < 1000; i++)
117     {
118         newArray[i] = Array[i];
119     }
120
121     cout<<"The array is:\n";
122     printArray(Array, 1000);
123
124     cout<<"Enter a element to search: ";
125     cin>>element;
126
127     LinearSearch(Array, 1000, element);
```

Activities Firefox Web Browser Tue Nov 16 21:00 19.1 °C

DSA\_Assignments/Search x +

https://github.com/RPChinhara-202117/DSA\_Assignments/blob/main/Assignment-8/SearchingSortingTimes/SearchingSortingTimes.cpp

Udemy Kaggle GitHub LinkedIn Reddit WhatsApp Chess LibGen GeeksforGeeks Classes Mail Drive YouTube South Park Other Bookmarks

```
110 {
111     int Array[1000], newArray[1000], element;
112     for(int i = 0; i < 1000; i++)
113     {
114         Array[i] = rand()%1000;
115     }
116     for(int i = 0; i < 1000; i++)
117     {
118         newArray[i] = Array[i];
119     }
120
121     cout<<"The array is:\n";
122     printArray(Array, 1000);
123
124     cout<<"Enter a element to search: ";
125     cin>>element;
126
127     LinearSearch(Array, 1000, element);
128
129     BubbleSort(Array, 1000);
130     printArray(Array, 1000);
131
132     BinarySearch(Array, 0, 999, element);
133
134
135     clock_t start = clock();
136     QuickSort(newArray, 0, 999);
137     cout<<fixed<<"\nExecution time of Quick Sort: "<<float(clock() - start)/CLOCKS_PER_SEC<<endl;
138
139     printArray(newArray, 1000);
140
141     return 0;
142 }
```

# Output:

```
Activities Visual Studio Code Tue Nov 16 20:57 20.1 °C
SearchingSortingTimes.cpp - Untitled (Workspace) - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
v UNTITLED (WORKSPA...
  v DSA_Assign...
    > Assignment-1
    > Assignment-2
    > Assignment-3
    > Assignment-4
    > Assignment-5
    > Assignment-6
    > Assignment-7
    v Assignment...
      ① README.md U
      Searching... U
      Searching... U
      Assignment...
      ① README.md U
      ① README.md

OUTLINE
TIMELINE

main* 0 0 0 0 0
```

```
cd "/home/rudrachinhara/Documents/CUH/3rd Sem/Data Structure and Algorithms/DSA_Assignments/Assignment-8/SearchingSortingTimes/"
" g++ SearchingSortingTimes.cpp -o SearchingSortingTimes && "/home/rudrachinhara/Documents/CUH/3rd Sem/Data Structure and Al
gorithms/DSA_Assignments/Assignment-8/SearchingSortingTimes/"SearchingSortingTimes
rudrachinhara@dell-vostro:~/Documents/CUH/3rd Sem/Data Structure and Algorithms/DSA_Assignments$ cd "/home/rudrachinhara/Docume
nts/CUH/3rd Sem/Data Structure and Algorithms/DSA_Assignments/Assignment-8/SearchingSortingTimes/" && g++ SearchingSortingTimes
.cpp -o SearchingSortingTimes && "/home/rudrachinhara/Documents/CUH/3rd Sem/Data Structure and Algorithms/DSA_Assignments/Assi
gnment-8/SearchingSortingTimes/"SearchingSortingTimes
The array is:
383 886 777 915 793 335 386 492 649 421 362 27 690 59 763 926 540 426 172 736 211 368 567 429 782 530 862 123 67 135 929 802 22
58 69 167 393 456 11 42 229 373 421 919 784 537 198 324 315 370 413 526 91 980 956 873 862 170 996 281 305 925 84 327 336 505
846 729 313 857 124 895 582 545 814 367 434 364 43 750 87 808 276 178 788 584 403 651 754 399 932 60 676 368 739 12 226 586 94
539 795 570 434 378 467 601 97 902 317 492 652 756 301 280 286 441 865 689 444 619 440 729 31 117 97 771 481 675 709 927 567 85
6 497 353 586 965 306 683 219 624 528 871 732 829 503 19 270 368 708 715 340 149 796 723 618 245 846 451 921 555 379 488 764 22
8 441 350 193 960 34 764 124 814 987 856 743 491 227 365 859 936 432 551 437 228 275 407 474 121 858 395 29 237 335 793 818 428
143 11 928 529 776 404 443 763 613 538 606 840 904 818 128 688 369 917 917 996 324 743 470 183 490 499 772 725 644 590 505 139
954 786 669 82 542 464 197 507 355 804 348 611 622 828 299 343 746 568 340 422 311 810 605 801 661 730 878 305 320 736 444 626
522 465 708 416 282 258 924 637 62 624 600 36 452 899 379 550 468 71 973 131 881 930 933 894 660 163 199 981 899 996 959 773 8
13 668 190 95 926 466 84 340 90 684 376 542 936 107 445 756 179 418 887 412 348 172 659 9 336 210 342 587 206 301 713 372 321 2
55 819 599 721 904 939 811 940 667 705 228 127 150 984 658 920 224 422 269 396 81 630 84 292 972 672 850 625 385 222 299 640 42
898 713 298 190 524 590 209 581 819 336 732 155 994 4 379 769 273 776 850 255 860 142 579 884 993 205 621 567 504 613 961 754
326 259 944 202 202 506 784 21 842 868 528 189 872 908 958 498 36 808 753 248 303 333 133 648 890 754 567 746 368 529 500 46 78
8 797 249 990 303 33 363 497 253 892 686 125 152 996 975 188 157 729 436 460 414 921 460 304 28 27 50 748 556 902 794 697 699 4
3 39 2 428 403 500 681 647 538 159 151 535 134 339 692 215 127 504 629 49 964 285 429 343 335 177 900 238 971 949 289 367 988 2
92 795 743 144 829 390 682 340 541 569 826 232 261 42 360 117 23 761 81 309 190 425 996 367 677 234 690 626 524 57 614 168 205
358 312 386 100 346 726 994 916 552 578 529 946 290 647 970 51 80 631 593 857 627 312 886 214 355 512 90 412 479 610 969 189 27
4 355 641 620 433 987 888 338 566 770 284 856 417 606 260 849 237 205 59 217 518 945 783 873 458 873 637 289 483 607 478 757 31
4 471 729 100 459 618 438 25 388 74 233 157 681 493 358 270 699 417 839 569 363 622 794 173 847 431 462 682 390 292 791 57 115
521 157 574 491 947 951 231 21 537 740 54 30 98 325 81 516 516 2 231 139 796 404 338 580 218 21 970 862 812 379 977 685 536 904
176 483 207 759 857 744 499 911 127 950 236 560 818 105 563 49 244 711 805 934 291 375 955 614 589 768 993 818 805 882 822 982
717 30 93 574 126 593 486 253 543 74 814 713 179 377 762 775 88 919 710 732 294 17 346 235 137 691 153 943 573 328 925 291 710
18 217 836 963 55 90 858 130 904 571 661 633 685 789 73 604 851 805 250 868 503 485 6 195 639 949 120 967 226 763 677 596 981
865 560 36 955 770 518 211 342 532 196 379 321 270 984 172 427 234 40 283 72 398 830 63 347 950 30 573 714 59 522 47 924 82 435
232 204 954 443 898 486 640 278 159 262 262 683 41 848 723 324 272 122 154 335 821 457 365 747 171 776 269 218 701 703 653 933
907 959 728 806 797 720 84 308 334 698 991 376 898 715 52 171 189 559 506 10 16 224 109 539 0 378 109 53 81 114 338 989 426 67
147 223 787 231 532 122 281 875 850 179 590 254 350 131 813 857 494 181 81 603 720 433 982 181 487 415 296 825 404 722 892 551
297 32 134 181 506 415 57 708 595 999 962 297 483 776 154 977 309 587 932 382 21 266 563 860 682 211 685 86 285 930 990 583 31
4 476 116 820 892 525 528 839 525 490 136 360 618 643 337 928 582 621 310 955 888 225 815 570 437 853 8 722 783 350 657 97 827
```

```
Activities Visual Studio Code Tue Nov 16 20:57 20.1 °C
SearchingSortingTimes.cpp - Untitled (Workspace) - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
v UNTITLED (WORKSPA...
  v DSA_Assign...
    > Assignment-1
    > Assignment-2
    > Assignment-3
    > Assignment-4
    > Assignment-5
    > Assignment-6
    > Assignment-7
    v Assignment...
      ① README.md U
      Searching... U
      Searching... U
      Assignment...
      ① README.md U
      ① README.md

OUTLINE
TIMELINE

main* 0 0 0 0 0
```

```
907 959 728 806 797 720 84 308 334 698 991 376 898 715 52 171 189 559 506 10 16 224 109 539 0 378 109 53 81 114 338 989 426 67
147 223 787 231 532 122 281 875 850 179 590 254 350 131 813 857 494 181 81 603 720 433 982 181 487 415 296 825 404 722 892 551
297 32 134 181 506 415 57 708 595 999 962 297 483 776 154 977 309 587 932 382 21 266 563 860 682 211 685 86 285 930 990 583 31
4 476 116 820 892 525 528 839 525 490 136 360 618 643 337 928 582 621 310 955 888 225 815 570 437 853 8 722 783 350 657 97 827
126 269 71 651 149 910 528 639 398 888 610 393 577 890 976 199 552 931 87 777 99 657 566 952 17 641 735 368 298 184 195 776 805
266 428 954 528 308 593 278 197 555 672 774 445 0 325 997 283 412 127 382 421
Enter a element to search: 593
Element is present at: 539

Execution time of Linear Search: 0.000031

Execution time of Bubble Sort: 0.005512
0 0 2 4 6 8 9 10 11 11 12 16 17 17 18 19 21 21 21 21 22 23 25 27 27 28 29 30 30 31 32 33 34 36 36 36 39 40 41 42 42 42 43
43 46 47 49 49 50 51 52 53 54 55 57 57 57 58 59 59 59 60 62 63 67 67 69 71 71 72 73 74 74 80 81 81 81 81 82 82 84 84 84 84 8
6 87 87 88 90 90 90 91 93 94 95 97 97 97 98 99 100 100 105 107 109 109 114 115 116 117 117 120 121 122 122 123 124 124 125 126
126 127 127 127 127 128 130 131 131 133 134 134 135 136 137 139 139 142 143 144 147 149 149 150 151 152 153 154 154 155 157 157
157 159 159 163 167 168 170 171 171 172 172 172 173 176 177 178 179 179 179 181 181 181 183 184 188 189 189 189 190 190 190 19
3 195 195 196 197 197 198 199 199 202 202 204 205 205 205 206 207 209 210 211 211 211 211 214 215 217 217 218 218 219 222 223 224 2
24 225 226 226 227 228 228 228 229 231 231 231 232 232 233 234 234 235 235 236 237 237 238 244 245 249 250 250 253 253 254 255
255 258 259 260 261 262 262 266 266 269 269 270 270 270 272 273 274 275 276 278 278 280 281 281 282 283 283 285 285 285 286
289 289 290 291 291 292 292 292 294 296 297 297 298 298 299 299 301 301 303 303 304 305 305 306 308 308 309 309 310 311 312 31
2 313 314 314 315 317 320 321 321 324 324 324 325 325 326 327 328 333 334 335 335 335 336 336 336 337 338 338 339 340 340 340
40 340 342 342 343 343 346 346 347 348 348 350 350 350 353 355 355 355 358 358 360 360 362 363 363 364 365 365 367 367 367 369
368 368 368 368 369 370 372 373 375 376 377 378 378 379 379 379 379 382 382 383 385 386 386 388 390 390 393 393 395 396
398 398 399 403 403 404 404 404 407 412 412 412 413 414 415 415 416 417 417 418 421 421 421 422 422 425 426 426 427 428 428 42
8 429 429 431 432 433 433 434 434 435 436 437 437 438 440 441 443 443 444 444 445 445 451 451 456 457 458 459 460 460 462 464 4
65 466 467 468 470 471 474 476 478 479 481 483 483 483 485 486 486 487 488 490 490 491 491 492 492 493 494 497 497 498 498 499
500 500 500 503 503 504 504 505 505 506 506 506 507 512 516 516 518 518 521 522 522 524 524 525 525 526 528 528 528 529
529 529 530 532 532 535 536 537 537 538 538 539 539 540 541 542 542 543 545 550 551 551 552 552 555 555 556 559 560 560 563 56
3 566 566 567 567 567 568 569 569 570 570 571 573 573 574 574 577 578 579 580 581 582 582 583 584 586 586 587 587 589 590 5
90 590 593 593 593 595 596 599 600 601 603 604 605 606 606 607 610 610 611 613 613 614 614 618 618 618 619 620 621 621 622 622
624 624 625 626 626 627 629 630 631 633 637 637 637 639 639 640 640 641 641 643 644 647 647 648 649 651 651 652 653 657 657 658 659
660 661 661 667 668 669 672 672 675 676 677 677 681 681 682 682 682 683 683 684 685 685 685 686 688 689 690 690 691 692 697 69
8 699 699 701 703 705 708 708 708 709 710 710 711 713 713 713 714 715 715 717 720 720 721 722 722 723 723 725 726 728 729 729 7
29 729 730 732 732 733 736 736 736 739 740 743 743 743 744 746 746 747 748 750 753 754 754 754 756 756 757 759 761 762 763 763
763 764 764 768 769 770 770 771 772 773 774 775 776 776 776 776 777 777 782 783 783 784 784 786 787 788 788 791 793 793
794 794 795 795 796 796 797 797 801 802 804 805 805 805 805 806 808 808 810 811 812 813 813 814 814 815 818 818 818 819 819 82
```

```
763 764 764 768 769 770 770 771 772 773 774 775 776 776 776 776 776 777 777 782 783 783 784 784 786 787 788 788 789 791 793 793
794 794 795 795 796 796 797 797 801 802 804 805 805 805 806 806 808 810 811 812 813 813 814 815 818 818 818 819 819 82
0 821 822 825 826 827 828 829 829 830 836 839 839 840 841 842 846 846 847 848 849 850 850 851 851 853 856 856 857 857 857 8
57 858 858 859 860 860 862 862 862 865 865 868 868 871 872 873 873 873 875 878 881 882 884 886 886 887 888 888 888 890 890 892
892 892 894 895 898 898 898 899 899 900 902 902 904 904 904 904 907 908 910 911 914 915 916 917 917 918 919 919 920 921 921 924
924 925 925 926 926 927 928 928 929 930 930 931 932 932 933 933 934 936 936 939 940 943 944 945 946 947 949 949 950 950 951 95
2 954 954 954 955 955 955 956 958 959 959 961 962 963 964 965 967 969 970 970 971 972 973 975 976 977 977 980 981 981 982 982 9
84 984 987 987 988 989 990 990 991 993 993 994 994 996 996 996 996 996 997 999
Element is present at: 602
Execution time of Binary Search: 0.000004

Execution time of Quick Sort: 0.000171
0 0 2 2 4 6 8 9 10 11 11 12 16 17 17 18 19 21 21 21 21 22 23 25 27 27 28 29 30 30 30 31 32 33 34 36 36 36 39 40 41 42 42 42 43
43 46 47 49 49 50 51 52 53 54 55 57 57 57 58 59 59 59 60 62 63 67 67 69 71 71 72 73 74 74 80 81 81 81 81 82 82 84 84 84 84 8
6 87 87 88 90 90 90 91 93 94 95 97 97 97 98 99 100 100 105 107 109 109 114 115 116 117 117 120 121 122 122 123 124 124 125 126
126 127 127 127 127 128 130 131 131 133 134 134 135 136 137 139 139 142 143 144 147 149 149 150 151 152 153 154 154 155 157 157
157 159 159 163 167 168 170 171 171 172 172 172 173 176 177 178 179 179 181 181 181 183 184 188 189 189 189 190 190 190 19
3 195 195 196 197 197 198 199 199 202 202 204 205 205 205 206 207 209 210 211 211 211 214 215 217 217 218 218 219 222 223 224 2
24 225 226 226 227 228 228 228 229 231 231 231 232 232 233 234 234 235 235 236 237 237 238 244 245 248 249 250 253 253 254 255
255 258 259 260 261 262 262 266 266 269 269 270 270 270 272 273 274 275 276 278 278 280 281 281 282 283 283 284 285 285 286
289 289 290 291 291 292 292 292 294 296 297 297 298 298 299 299 301 301 303 303 304 305 305 306 308 308 309 309 310 311 312 31
2 313 314 314 315 317 320 321 321 324 324 324 325 325 326 327 328 333 334 335 335 335 336 336 337 338 338 338 339 340 340 3
40 340 342 342 343 343 346 346 347 348 348 350 350 350 353 355 355 355 358 358 360 360 362 362 363 363 364 365 365 367 367 367 368
368 368 368 368 369 370 372 373 375 376 376 377 378 378 379 379 379 379 382 382 383 385 386 386 388 390 390 393 393 395 396
398 398 399 403 403 404 404 404 407 412 412 412 413 414 415 415 416 417 417 418 421 421 421 422 422 425 426 426 427 428 428 42
8 429 429 431 432 433 433 434 434 435 436 437 437 438 440 441 443 443 444 444 445 445 451 452 456 457 458 459 460 460 462 464 4
65 466 467 468 470 471 474 476 478 479 481 483 483 483 485 486 486 487 488 490 490 491 491 492 492 493 494 497 497 498 499 499
500 500 500 503 503 504 504 505 505 506 506 506 507 512 516 516 518 518 521 522 522 524 524 525 525 526 528 528 528 528 529
529 529 530 532 532 535 536 537 537 538 538 539 539 540 541 542 542 543 545 550 551 551 552 552 555 555 556 559 560 560 563 56
3 566 566 567 567 567 568 569 569 570 570 571 573 573 574 574 577 578 579 580 581 582 582 583 584 586 586 587 587 589 590 5
90 590 593 593 593 595 596 599 600 601 603 604 605 606 606 607 610 610 611 613 613 614 614 618 618 618 619 620 621 621 622 622
624 624 625 626 626 627 629 630 631 633 637 637 639 639 640 640 641 641 643 644 647 647 648 649 651 651 652 653 657 657 658 659
660 661 661 667 668 669 672 672 675 676 677 677 681 681 682 682 682 683 683 684 685 685 685 686 688 689 690 690 691 692 697 69
8 699 699 701 703 705 708 708 708 709 710 710 711 713 713 713 714 715 715 717 720 720 721 722 722 723 723 725 726 728 729 729 7
29 729 730 732 732 732 735 736 736 739 740 743 743 743 744 746 746 747 748 750 753 754 754 754 756 756 757 759 761 762 763 763
763 764 764 768 769 770 770 771 772 773 774 775 776 776 776 777 777 782 783 783 784 784 786 787 788 788 789 791 793 793
794 794 795 795 796 796 797 797 801 802 804 805 805 805 805 806 808 808 810 811 812 813 813 814 814 815 818 818 818 819 819 82
```

```
Element is present at: 602
Execution time of Binary Search: 0.000004

Execution time of Quick Sort: 0.000171
0 0 2 2 4 6 8 9 10 11 11 12 16 17 17 18 19 21 21 21 21 22 23 25 27 27 28 29 30 30 30 31 32 33 34 36 36 36 39 40 41 42 42 42 43
43 46 47 49 49 50 51 52 53 54 55 57 57 57 58 59 59 59 60 62 63 67 67 69 71 71 72 73 74 74 80 81 81 81 81 82 82 84 84 84 84 8
6 87 87 88 90 90 90 91 93 94 95 97 97 97 98 99 100 100 105 107 109 109 114 115 116 117 117 120 121 122 122 123 124 124 125 126
126 127 127 127 127 128 130 131 131 133 134 134 135 136 137 139 139 142 143 144 147 149 149 150 151 152 153 154 154 155 157 157
157 159 159 163 167 168 170 171 171 172 172 172 173 176 177 178 179 179 181 181 181 183 184 188 189 189 189 190 190 190 19
3 195 195 196 197 197 198 199 199 202 202 204 205 205 205 206 207 209 210 211 211 211 214 215 217 217 218 218 219 222 223 224 2
24 225 226 226 227 228 228 228 229 231 231 231 232 232 233 234 234 235 235 236 237 237 238 244 245 248 249 250 253 253 254 255
255 258 259 260 261 262 262 266 266 269 269 270 270 270 272 273 274 275 276 278 278 280 281 281 282 283 283 284 285 285 286
289 289 290 291 291 292 292 292 294 296 297 297 298 298 299 299 301 301 303 303 304 305 305 306 308 308 309 309 310 311 312 31
2 313 314 314 315 317 320 321 321 324 324 324 325 325 326 327 328 333 334 335 335 335 336 336 337 338 338 338 339 340 340 3
40 340 342 342 343 343 346 346 347 348 348 350 350 350 353 355 355 355 358 358 360 360 362 362 363 363 364 365 365 367 367 367 368
368 368 368 368 369 370 372 373 375 376 376 377 378 378 379 379 379 379 382 382 383 385 386 386 388 390 390 393 393 395 396
398 398 399 403 403 404 404 404 407 412 412 412 413 414 415 415 416 417 417 418 421 421 421 422 422 425 426 426 427 428 428 42
8 429 429 431 432 433 433 434 434 435 436 437 437 438 440 441 443 443 444 444 445 445 451 452 456 457 458 459 460 460 462 464 4
65 466 467 468 470 471 474 476 478 479 481 483 483 483 485 486 486 487 488 490 490 491 491 492 492 493 494 497 497 498 499 499
500 500 500 503 503 504 504 505 505 506 506 506 507 512 516 516 518 518 521 522 522 524 524 525 525 526 528 528 528 528 529
529 529 530 532 532 535 536 537 537 538 538 539 539 540 541 542 542 543 545 550 551 551 552 552 555 555 556 559 560 560 563 56
3 566 566 567 567 567 568 569 569 570 570 571 573 573 574 574 577 578 579 580 581 582 582 583 584 586 586 587 587 589 590 5
90 590 593 593 593 595 596 599 600 601 603 604 605 606 606 607 610 610 611 613 613 614 614 618 618 618 619 620 621 621 622 622
624 624 625 626 626 627 629 630 631 633 637 637 639 639 640 640 641 641 643 644 647 647 648 649 651 651 652 653 657 657 658 659
660 661 661 667 668 669 672 672 675 676 677 677 681 681 682 682 682 683 683 684 685 685 685 686 688 689 690 690 691 692 697 69
8 699 699 701 703 705 708 708 708 709 710 710 711 713 713 713 714 715 715 717 720 720 721 722 722 723 723 725 726 728 729 729 7
29 729 730 732 732 732 735 736 736 739 740 743 743 743 744 746 746 747 748 750 753 754 754 754 756 756 757 759 761 762 763 763
763 764 764 768 769 770 770 771 772 773 774 775 776 776 776 777 777 782 783 783 784 784 786 787 788 788 789 791 793 793
794 794 795 795 796 796 797 797 801 802 804 805 805 805 805 806 808 808 810 811 812 813 813 814 814 815 818 818 818 819 819 82
0 821 822 825 826 827 828 829 829 830 836 839 839 840 841 842 846 846 847 848 849 850 850 850 851 853 856 856 856 857 857 857 8
57 858 858 859 860 860 862 862 862 865 865 868 868 871 872 873 873 873 875 878 881 882 884 886 886 887 888 888 888 890 890 892
892 892 894 895 898 898 898 899 899 900 902 902 904 904 904 904 907 908 910 911 914 915 916 917 917 918 919 919 920 921 921 924
924 925 925 926 926 927 928 928 929 930 930 931 932 932 933 933 934 936 936 939 940 943 944 945 946 947 949 949 950 950 951 95
2 954 954 954 955 955 955 956 958 959 959 961 962 963 964 965 967 969 970 970 971 972 973 975 976 977 977 980 981 981 982 982 9
84 984 987 987 988 989 990 990 991 993 993 994 994 996 996 996 996 996 997 999
rudrachinharade@vostro:~/Documents/CUH/3rd Sem/Data Structure and Algorithms/DSA_Assignments/Assignment-8/SearchingSortingTi
mes$
```

# Conclusion:

As we can see,

- Linear Search takes longer than Binary Search
- Bubble Sort takes more time to execute than Quick Sort