**NAME : ROHAN NYATI**

**ROOL NO : R177219148**

**SAP ID : 500075940**

**BATCH : 5 (AI & ML)**

**LAB 3**

**CODE:**

#include<stdio.h>

#include<time.h>

void display();

int partition(int [], int, int);

void quick(int [], int, int);

void merge(int [], int, int, int);

void merge\_sort(int [], int, int);

void selectionSort(int [], int );

void bubble\_sort(int[],int);

int main()

{

int start, end, mid;

clock\_t start\_time,end\_time;

double time\_required;

start = 0;

int i,n;

scanf("%d",&n);

int arr[n],a1[n],a2[n],a3[n],a4[n];

for(i = 0; i< n; i++){

scanf("%d",&arr[i]);

}

for(i=0;i<n;i++){

a1[i]=a2[i]=a3[i]=a4[i]=arr[i];

}

end = n - 1;

mid = (start + end) / 2;

start\_time=clock();

quick(a1, start, end);

end\_time=clock();

time\_required=("%d\n",((double)(end\_time-start\_time))\*10e6)/CLOCKS\_PER\_SEC;

printf("total time taken by quick sort to sort the elements:%6.3f\n",time\_required);

start\_time=clock();

merge(a2, start, mid, end);

merge\_sort(a2, start, end);

end\_time=clock();

time\_required=("%d\n",((double)(end\_time-start\_time))\*10e6)/CLOCKS\_PER\_SEC;

printf("total time taken by merge sort:%6.3f\n",time\_required);

start\_time=clock();

selectionSort(a3,n);

end\_time=clock();

time\_required=("%d\n",((double)(end\_time-start\_time))\*10e6)/CLOCKS\_PER\_SEC;

printf("total time taken by selection sort:%6.3f\n",time\_required);

start\_time=clock();

bubble\_sort(a4,n);

end\_time=clock();

time\_required=("%d\n",((double)(end\_time-start\_time))\*10e6)/CLOCKS\_PER\_SEC;

printf("total time taken by bubble sort:%6.3f\n",time\_required);

}

void bubble\_sort(int array[], int n)

{

for ( int i = 0; i < n ; i++)

{

for ( int j = 0; j < n-i -1 ; j++)

{

if (array[j] > array[j+1])

{

int temp=array[j];

array[j]=array[j+1];

array[j+1]=temp;

}

}

}

}

void swap(int \*a, int \*b) {

int t = \*a;

\*a = \*b;

\*b = t;

}

int partition(int array[], int low, int high) {

int pivot = array[high];

int i = (low - 1);

for (int j = low; j < high; j++) {

if (array[j] <= pivot) {

i++;

swap(&array[i], &array[j]);

}

}

swap(&array[i + 1], &array[high]);

return (i + 1);

}

void quick(int a[], int start, int end) {

int loc;

if(start < end) {

loc = partition(a, start, end);

quick(a, start, loc - 1);

quick(a, loc + 1, end);

}

}

void merge(int a[], int start, int mid, int end) {

int i, j, k, index = start, temp[20000];

i = start;

j = mid +1;

while((i<= mid) && (j <= end)) {

if(a[i] < a[j]) {

temp[index] = a[i];

i++;

}

else {

temp[index] = a[j];

j = j + 1;

}

index++;

}

if(i> mid) {

while(j <= end) {

temp[index] = a[j];

j++;

index++;

}

}

else {

while(i<= mid) {

temp[index] = a[i];

i++;

index++;

}

}

for(k = start; k < index; k++) {

a[k] = temp[k];

}

}

void merge\_sort(int a[], int start, int end) {

int mid;

if(start < end) {

mid = (start + end) / 2;

merge\_sort(a, start, mid);

merge\_sort(a, mid + 1, end);

merge(a, start, mid, end);

}

}

void selectionSort(int arr[], int n)

{

int i, j, min\_idx;

for (i = 0; i < n-1; i++)

{

min\_idx = i;

for (j = i+1; j < n; j++)

{

if (arr[j] < arr[min\_idx])

min\_idx = j;

}

int temp=arr[i];

arr[i]=arr[min\_idx];

arr[min\_idx]=temp;

}

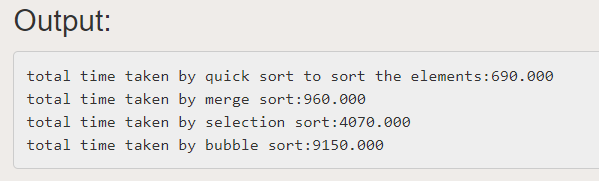
}

**INPUT:**

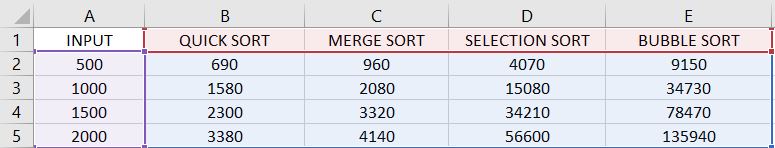
500

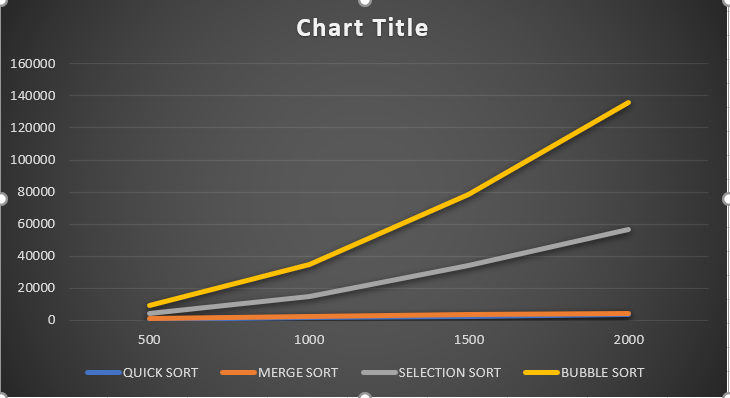
882 666 258 817 661 824 852 416 110 578 203 456 600 258 814 24 495 775 813 804 281 311 603 774 38 642 254 163 481 306 20 52 973 969 559 633 793 411 48 593 989 251 48 279 199 552 994 695 326 806 188 297 807 481 70 535 122 14 698 294 321 409 36 293 377 596 616 169 697 354 763 375 606 501 654 805 53 337 189 69 833 377 57 640 859 127 174 671 142 563 655 463 972 692 446 348 978 61 208 674 105 661 48 711 852 393 206 595 730 85 355 563 463 412 893 11 539 757 372 371 319 27 524 981 409 970 329 386 721 537 59 827 888 107 227 739 500 123 334 921 209 689 173 362 100 65 63 329 512 435 391 832 152 915 812 561 575 831 947 986 57 5 812 945 804 730 684 994 543 708 914 752 86 777 804 186 532 867 205 44 993 596 566 144 201 67 706 776 899 652 761 646 348 263 591 151 683 965 144 225 672 749 668 448 525 471 634 57 28 529 792 20 125 357 164 326 114 870 792 703 212 552 349 560 505 940 711 187 594 855 413 265 603 80 713 819 241 36 876 269 255 357 289 380 404 144 396 518 704 187 221 916 430 570 475 625 199 876 813 793 421 916 57 714 686 460 532 617 186 97 887 441 454 866 512 858 9 908 66 714 786 287 319 906 857 485 530 55 360 32 848 781 948 595 495 323 745 717 941 621 815 827 752 959 382 263 507 392 862 573 796 337 860 114 242 406 599 462 151 650 495 689 430 132 974 615 456 408 22 396 28 837 913 780 485 985 734 992 376 595 254 862 932 804 977 864 900 265 15 50 915 510 429 345 643 402 650 789 500 672 875 528 198 477 999 374 461 732 55 528 16 310 389 638 113 365 191 13 321 206 754 235 717 183 270 49 275 921 838 776 282 712 994 171 879 992 545 339 413 600 867 119 600 256 757 404 311 638 107 632 845 861 868 561 734 137 300 700 748 137 475 721 539 158 892 107 841 126 446 253 416 313 63 16 259 510 420 570 148 527 893 993 77 450 243 812 277 543 511 25 370 676 436 599 834 17 706 674 143 842 618 249 154 681 265 103 190 375 364 338 592 256 20 670 706 263 171 983 806 372 698 866 47 824 464 571 841 860 936 674 702 553 924 546 924 879 650 804 254 13

**OUTPUT:**

****

**CHART:**

****

****