|  |  |
| --- | --- |
| **NAME:** | Arnav Hoskote |
| **UID:** | 2021300044 |
| **SUBJECT** | Design and Analysis of Algorithms |
| **EXPERIMENT NO :** | 01b |
| **DATE OF PERFORMANCE** | 06/02/2023 |
| **DATE OF SUBMISSION** | 13/02/2023 |
| **AIM:** | To sort 100,000 values using insertion sort and selection sort. |
| **PROBLEM STATEMENT 1:** |  |
| **ALGORITHM:** | **Insertion Sort:**  procedure insertionSort(A: list of sortable items)  n = length(A)  for i = 1 to n - 1 do  j = i  while j > 0 and A[j-1] > A[j] do  swap(A[j], A[j-1])  j = j - 1  end while  end for  end procedure  **Selection Sort:**  SELECTION SORT(arr, n)  Step 1: Repeat Steps 2 and 3 for i = 0 to n-1  Step 2: CALL SMALLEST(arr, i, n, pos)  Step 3: SWAP arr[i] with arr[pos]  [END OF LOOP]  Step 4: EXIT  SMALLEST (arr, i, n, pos)  Step 1: [INITIALIZE] SET SMALL = arr[i]  Step 2: [INITIALIZE] SET pos = i  Step 3: Repeat for j = i+1 to n  if (SMALL > arr[j])  SET SMALL = arr[j]  SET pos = j  [END OF if]  [END OF LOOP]  Step 4: RETURN pos |
| **PROGRAM:** | #include<stdio.h>  #include<time.h>  #include<stdlib.h>  void insertion(long int n,long int a[])  {  long int i,j,temp;  for(i=0;i<n;i++)  {  temp=a[i];  j=i-1;  while(j>=0 && a[j]>temp)  {  a[j+1]=a[j];  j--;  }  a[j+1]=temp;  }  }  void selection(long int n,long int a[])  {  long int i,j,min,temp;  for(i=0;i<n-1;i++)  {  min=i;  for(j=i+1;j<n;j++)  {  if(a[j]<a[min])  {  min=j;  }  }  if(min!=i)  {  temp=a[min];  a[min]=a[i];  a[i]=temp;  }  }  }  void main()  {  long int i,t=0,n=100,numbers;  double time1[10],time2[10];  printf("intervals\tInsertion\t\tSelection\n");  while(t++<1000)  {  long int b[n],c[n];  for(int i=0;i<n;i++)  {  srand(time(NULL));  numbers=(rand()%100000);  b[i]=numbers;  c[i]=numbers;  }  clock\_t end,start;  start=clock();  insertion(n,b);  end=clock();  time1[t]=((double)(end-start)/CLOCKS\_PER\_SEC));  start=clock();  selection(n,c);  end=clock();  time2[t]=((double)(end-start)/CLOCKS\_PER\_SEC);  printf("%li\t\t%lf\t\t%lf\n",n,time1[t]+=((double)(end-start)/CLOCKS\_PER\_SEC),time2[t]+=((double)(end-start)/CLOCKS\_PER\_SEC));  n+=100;  }  } |
| **CONCLUSION:** | By performing the above experiment I have understood insertion sort and selection sort algorithms along with their time complexities. |