Suraj Iyer

2021300045

SE Comps A

Batch C

DAA EXPERIMENT 1B

Aim - Experiment on finding the running time of an algorithm.

Details - The understanding of running time of algorithms is explored by implementing two basic sorting algorithms namely Insertion and Selection sorts. These algorithms work as follows.

Insertion sort - It works similar to the sorting of playing cards in hands. It is assumed that the first card is already sorted in the card game, and then we select an unsorted card. If the selected unsorted card is greater than the first card, it will be placed at the right side; otherwise, it will be placed at the left side. Similarly, all unsorted cards are taken and put in their exact place.

Selection sort - It first finds the smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array. In this algorithm, the array is divided into two parts, first is the sorted part, and another one is the unsorted part. Initially, the sorted part of the array is empty, and the unsorted part is the given array. Sorted part is placed at the left, while the unsorted part is placed at the right. In selection sort, the first smallest element is selected from the unsorted array and placed at the first position. After that second smallest element is selected and placed in the second position. The process continues until the array is entirely sorted.

Code -

| #include <stdio.h>  #include <stdlib.h>  #include <time.h>  const int limit = 100000;  const int block = 100;  void insertion\_sort (FILE \*f) {  FILE \*fp;  fp = fopen("daa\_2\_insertion\_sort.txt", "w");  fprintf(fp,"Block Size\tTime Taken\n");  int size = 0;  for (int times = 0; times<limit/block; times++) {  size+=block;  int arr [size];  for (int i = 0; i<size; ++i)  fscanf(f,"%d",&arr[i]);  // now our array is ready, we perform insertion sort  clock\_t t;  t = clock();  int i, key, j;  for (i = 1; i<size; i++) {  key = arr[i];  j=i-1;  while (j>=0&&arr[j]>key) {  arr[j+1] = arr[j];  j=j-1;  }  arr[j+1] = key;  }  t = clock()-t;  double time\_taken = ((double)t)/CLOCKS\_PER\_SEC;  // storing the result in a file  fprintf(fp,"%d\t%lf\n",size,time\_taken);  }  fclose(fp);  }  void selection\_sort (FILE \*f) {  FILE \*fp;  fp = fopen("daa\_2\_selection\_sort.txt", "w");  fprintf(fp,"Block Size\tTime Taken\n");  int size = 0;  for (int times = 0; times<limit/block; times++) {  size+=block;  int arr [size];  for (int i = 0; i<size; ++i)  fscanf(f,"%d",&arr[i]);  // now our array is ready, we perform selection sort  clock\_t t;  t = clock();  int i, j, mini;  for (i = 0; i<size-1; i++) {  mini = i;  for (j = i+1; j<size; j++) {  if (arr[j]<arr[mini])  mini = j;  }  if(mini!=i) {  int temp = arr[mini];  arr[mini] = arr[i];  arr[i] = temp;  }  }  t = clock()-t;  double time\_taken = ((double)t)/CLOCKS\_PER\_SEC;  // storing the result in a file  fprintf(fp,"%d\t%lf\n",size,time\_taken);  }  fclose(fp);  }  int main () {  // generating 1,00,000 integers and storing them in a file  FILE \*f;  f = fopen("daa\_2\_random\_integers.txt", "w");  for (int i = 0; i<limit; ++i)  fprintf(f,"%d\n",rand());  // insertion sort  insertion\_sort(f);  // selection sort  selection\_sort(f);  fclose(f);  return 0;  } |
| --- |

1,00,000 randomly generated integers -

|  |
| --- |

Size of block along with time taken to sort using insertion sort -

|  |
| --- |

Size of block along with time taken to sort using selection sort -

|  |
| --- |

Graph of time taken to sort using insertion sort and selection sort against size of blocks to be sorted -

|  |
| --- |

Conclusion - It is with simple observation that we can remark that insertion sort is a much more efficient algorithm to sort numbers than selection sort. What took insertion sort a mere 8 seconds, took selection sort an entire 50 minutes. We also observe that even though the time complexity of selection sort is much greater than that of insertion sort, the space complexity for both is the same. Both insertion and selection sort have O(1) space complexity. By conducting the experiment I have revised myself with the concepts of file handling in C programming, and also acquainted myself with the time.h library which helps us calculate precisely the time required to process and compile several functions and algorithms.