classmate
Date
Page

20/6/24

Lab - 8

1) Heap Sort

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void swap (int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
void heapify (int arr[], int N, int i) {
    int largest = i;
    int left = 2*i + 1;
    int right = 2*i + 2;
    if (left < N && arr[left] > arr[largest])
        largest = left;
    if (right < N && arr[right] > arr[largest])
        largest = right;
    if (largest != i) {
        swap (&arr[i], &arr[largest]);
        heapify (arr, N, largest);
    }
}
```

N
20/6/24

```
void heapSort (int arr[], int N) {
    for (int i = N/2 - 1; i >= 0; i--)
        heapify (arr, N, i);
    for (int i = N-1; i >= 0; i --) {
        swap (& arr[0], & arr[i]);
        heapify (arr, i, 0);
    }
}
```

```
int main() {
    int a[15000], n, i, j, ch, temp;
    clock_t start, end;
    while (1) {
        printf("1: For manual entry of N value and array elements");
        printf("\n2: To display time taken for sorting number of elements
                N in the range 500 to 14500");
      printf("\n3: To exit");
      printf("\n Enter your choice: ");
      scanf("%d", &ch);
      switch (ch) {
          case 1:
              printf(" Enter the no of elements ");
              scanf("%d", &n);
              printf(" Enter array elements: ");
              for (i=0; i<n; i++)
                  scanf("%d", &a[i]);
              start = clock();
              heapSort (a, n);
              end = clock();
              printf ("Sorted array is: ");
              for (i=0; i<n; i++)
                   printf("%d\t", a[i]);
             printf("Time taken to sort %d numbers is %f secs\n", n,
                 ((double)(end-start)) / CLOCKS_PER_SEC());
             break;
      case 2:
             n = 500;
             while (n<=14500) {
                 for (i=0; i<n; i++)
                     a[i] = n-i;
                 start = clock();
                 heapSort (a, n);
                 for (j=0; j<50000000; j++) { temp = 38/600; }
```

```
            end = clock();
            printf("Time taken to sort %d numbers is %f secs", n,
                ((double) (end - start)) / CLOCKS_PER_SEC);
            n += 1000;
        }
        break;
    case 3:
        exit(0);
    default:
        printf("\n Invalid choice! Please try again.\n");
    }
}
return 0;
}
```

Output:-

```
1. For manual entry of N value and array elements
2. To display time taken for sorting number of elements N in the range
   500 to 14500
3. To exit
Enter your choice: 1
Enter the number of elements: 5
Enter array elements: 4 3 5 2 1
Sorted array is 1 2 3 4 5
```

(5) Floyd Algorithm

```
#include <stdio.h>
#define V 5
#define INF 99999

void printSolution (int dist[][V]);
void floydWarshall (int dist[][V]){
    int i, j, k;
    for (k=0; k<V; k++) {
        for (i=0; i<V; i++){
            for (j=0; j<V; j++) {
                if (dist[i][k] + dist[k][j] < dist[i][j])
                    dist[i][j] = dist[i][k] + dist[k][j];
            }
        }
    }
    printSolution (dist);
}

void printSolution (int dist[][V]) {
    printf ("The following matrix shows the shortest distances "
            "between every pair of vertices \n");
    for (int i=0; i<V; i++) {
        for (int j=0; j<V; j++) {
            if (dist[i][j] == INF)
                printf ("%7s", "INF");
            else
                printf ("%7d", dist[i][j]);
        }
        printf ("\n");
    }
}
```

```
int main(){
    int graph[v][v] = {{ 0, 4, INF, 5, INF },
                       {INF, 0, 1, INF, 6 },
                       {2, INF, 0, 3, INF},
                       { INF, INF, 1, 0, 2 },
                       {1, INF, INF, 4, 0} };
  floydWarshall(graph);
  return 0;
}
```

Output:-
The following matrix shows the shortest distances between every pair of vertices

```
    0   4   5   5   7
    3   0   1   4   6
    2   6   0   3   5
    3   7   1   0   2
    1   5   5   4   0
```

[illegible] 20/6/24