DATE: / /
PAGE NO: Usha Gold

Lab 8

20/6/24

Heap sort

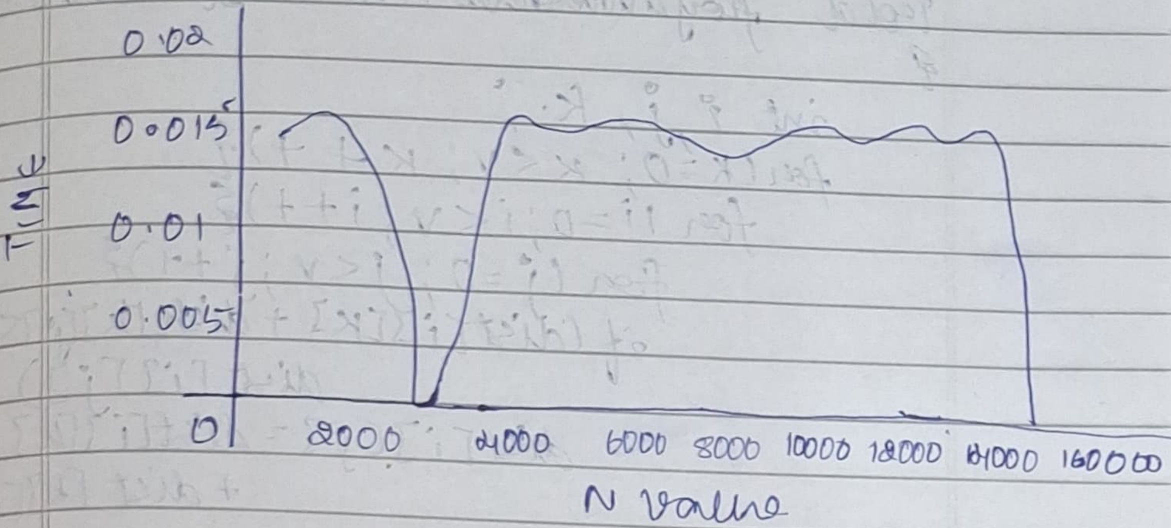
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void swap(int* a, int* b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void heapify(int a[], int n, int i)
{
    int large = i;
    int left = 2*i + 1;
    int right = 2*i + 2;
    if (left< n && a[left]> a[large])
        large = left;
    if (right<n && a[right] > a[large])
        large = right;
    if (large != i){
        swap(&a[i], &a[large]);
        heapify(a, n, large);
    }
}

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

N 29/6/24

```
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 [illegible]");
    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();
      ~~split~~(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 % 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<500000000; j++)
        { temp = 38/600; }
        end = clock();
        printf("Time taken to sort %d
            numbers is %f secs \n", n,
            ((double)(end - start)) /CLOCKSPERSEC
        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;
3.To exit

Enter your choice: 1
Enter the no of elements; 5
Enter array elements: 2 7 33 12 1
Sorted array is: 1 2 7 12 33
Time taken to sort 5 no is 0 secs
1. For manual entry of N value and array
elements
2. To display time taken for sorting no of
elements N.
3. To exit
Enter your choice: 2.
Time taken to sort 500 no is 0 secs
Time taken to sort 1500 no is 0.016 secs
Time taken to sort 2500 no is 0.016 secs
Time taken to sort 3500 no is 0.015 secs
Time taken to sort 4500 no is 0.016 secs
Time taken to sort 6500 no is 0.015 secs
Time taken to sort 7500 no is 0.016 secs
Time taken to sort 8500 no is 0.015 secs
Time taken to sort 9500 no is 0 secs
Time taken to sort 10500 no is 0.016 secs
Time taken to sort 11500 no is 0.16 secs
Time taken to sort 13500 no is 0.016 secs
Time taken to sort 14500 no is 0.016 secs
```



2. Floyd Algorithm.

```
#include <stdio.h>
#define V5
#define INF9999

void printSolution(int dist[][V])
{
  printf("Shortest distance matrix");
  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");
   }
}
```

NP
20/6/24

```
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);
}.

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

```
Shortest distance Matrix
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
```