Experiment 5

# Comparative Analysis of Sorting Techniques using Time Complexity

## Objective

Implementation of Heap sort.

## C++-Program

#include <iostream>

using namespace std;

void swap( double \*a, double \*b )

{

double temp = \*a;

\*a = \*b;

\*b = temp;

}

double max( double a, double b )

{

if( a >= b )

return a;

else

return b;

}

void displayVector( double \*vector, int size )

{

int i;

cout<<"[ ";

for( i = 0; i < size; i++ )

cout<<vector[i]<<" ";

cout<<"]";

}

void bottomUpMaxHeapify( double \*vector, int index )

{

int i;

for( i = index+1; i != 1; i = i/2 )

{

if( vector[i-1] > vector[i/2-1] )

swap( &vector[i-1], &vector[i/2-1] );

else

break;

}

}

void topDownMaxHeapify( double \*vector, int index, int size )

{

int i = index+1;

int maxIndex;

while(1)

{

if( 2\*i <= size )

if( 2\*i+1 <= size )

if( vector[2\*i-1] >= vector[2\*i] )

maxIndex = 2\*i;

else

maxIndex = 2\*i + 1;

else

maxIndex = 2\*i;

else

break;

if( vector[i-1] < vector[maxIndex-1] )

swap( &vector[i-1], &vector[maxIndex-1] );

else

break;

i = maxIndex;

}

}

void createMaxHeap( double \*vector, int size )

{

double \*maxHeap = new double[size];

int i;

for( i = 0; i < size; i++ )

{

maxHeap[i] = vector[i];

bottomUpMaxHeapify( maxHeap, i );

}

for( i = 0; i < size; i++ )

vector[i] = maxHeap[i];

}

void maxHeapSort( double \*vector, int size )

{

createMaxHeap( vector, size ); displayVector( vector, size );

while( size > 1 )

{

swap( &vector[0], &vector[size-1] );

size--;

topDownMaxHeapify( vector, 0, size );

}

}

int main( void )

{

double vector[5] = {5, 4, 6, 3, 1};

cout<<"Orginal array: ";

displayVector( vector, 5 );

maxHeapSort( vector, 5 );

cout<<endl<<"Sorted array: ";

displayVector( vector, 5 );

cout<<endl;

return 0;

}