Michael Smith

Lab 2 – Time Complexity

CSE 330 Data Structures

Winter 2017

1. The program is 100% complete. I was successful in implementing the sorting methods and measuring the time complexity of each sorting method to measure the constants.
2. Each sorting method has a O(n^2) time complexity. While the storage complexity of each sorting method is O(N) for the N terms inside the vectors of the function
3. Source Code
   1. Bubble.cpp

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Michael Smith

\* Bubble.cpp

\* 01/23/2017

\* This program will use the bubble sort method in order to sort random numbers

\* generated through rand in numerical order. And time the execution time in seconds.

\* The main will ask user for number of numbers wanting to sort. Will declare a vector

\* and fill with randomly generated numbers. The vector is then passed to bubble\_sort()

\* method to sort the numbers in numerical order. In order to measure the constant from the

\* time complexity of the program.

\* O(n^2)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <vector>

#include <cstdlib>

#include <time.h>

using namespace std;

void bubble\_sort(vector<int> &v, int n);

// The main that asks user for the number input of wanting to sort

// then creates a vector of n numbers. Then randomly generates and fills

// the vector. Then passes the vector to the bubble\_sort() method for sorting.

int main()

{

int n;

cout << "Enter the number of numbers you want to sort"<< endl;

cin >> n;

vector <int> v(n);

srand(time(0));

for(int i = 0; i < n; i++)

v[i] = rand()%1000000;

bubble\_sort(v, n);

return 0;

}

// This function will receive the vector and n. It processes through the

// vector to sort the numbers in numerical order using the bubble sort method.

void bubble\_sort(vector<int> &v, int n)

{

int i, j;

for (i = n-1; i > 0; i--){

for (j = 0; j < i; j++){

if (v[j] > v[j+1])

swap(v[j], v[j+1]);

}

}

}

* 1. Insert.cpp

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Michael Smith

\* insert.cpp

\* 01/23/2017

\* This program will use the insertion sort method in order to sort random numbers

\* generated through rand in numerical order. And time the execution time in seconds.

\* The main will ask user for number of numbers wanting to sort. Will declare a vector

\* and fill with randomly generated numbers. The vector is then passed to insert\_sort()

\* method to sort the numbers in numerical order. In order to measure the constant from the

\* time complexity of the program.

\* O(n^2)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <vector>

#include <cstdlib>

#include <time.h>

using namespace std;

void insert\_sort(vector<int> &v, int n);

// The main that asks user for the number input of wanting to sort

// then creates a vector of n numbers. Then randomly generates and fills

// the vector. Then passes the vector to the insert\_sort() method for sorting.

int main()

{

int n;

cout << "Enter the number of numbers you want to sort"<< endl;

cin >> n;

vector <int> v(n);

srand(time(0));

for(int i = 0; i < n; i++)

v[i] = rand()%1000000;

insert\_sort(v, n);

return 0;

}

// This function will receive the vector and n. It processes through the

// vector to sort the numbers in numerical order using the insertion sort method.

void insert\_sort(vector<int> &v, int n)

{

int i, j, elem;

for (i = 1; i < n; i++){

for (elem = v[i], j = i-1; j >=0 and elem < v[j]; j--){

v[j+1] = v[j];

}

v[j+1] = elem;

}

}

c. Selection.cpp

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Michael Smith

\* selection.cpp

\* 01/23/2017

\* This program will use the selection sort method in order to sort random numbers

\* generated through rand in numerical order. And time the execution time in seconds.

\* The main will ask user for number of numbers wanting to sort. Will declare a vector

\* and fill with randomly generated numbers. The vector is then passed to selection\_sort()

\* method to sort the numbers in numerical order. In order to measure the constant from the

\* time complexity of the program.

\* O(n^2)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <vector>

#include <cstdlib>

#include <time.h>

using namespace std;

void selection\_sort(vector<int> &v, int n);

// The main that asks user for the number input of wanting to sort

// then creates a vector of n numbers. Then randomly generates and fills

// the vector. Then passes the vector to the selection\_sort() method for sorting.

int main()

{

int n;

cout << "Enter the number of numbers you want to sort"<< endl;

cin >> n;

vector <int> v(n);

srand(time(0));

for(int i = 0; i < n; i++)

v[i] = rand()%1000000;

selection\_sort(v, n);

return 0;

}

// This function will receive the vector and n. It processes through the

// vector to sort the numbers in numerical order using the selection sort method.

void selection\_sort(vector<int> &v, int n)

{

int i, j;

for (i = 0; i < n-1; i++){

for (j = i+1; j < n; j++){

if (v[i] > v[j])

swap(v[i], v[j]);

}

}

}

1. Sample Run