1. Name: Kourun Sok – [kourun@csu.fullerton.edu](mailto:kourun@csu.fullerton.edu)

Linh Cao - [linhcao@csu.fullerton.edu](mailto:l%69n%68%63%61o@cs%75%2e%66%75%6c%6c%65%72%74%6f%6e%2e%65%64%75)

This is a Submission for Project 3 – Bubble sort and merge sort

Class CPSC 335: M/W 1-2.15PM

Due: Thursday, 5/7 /2015

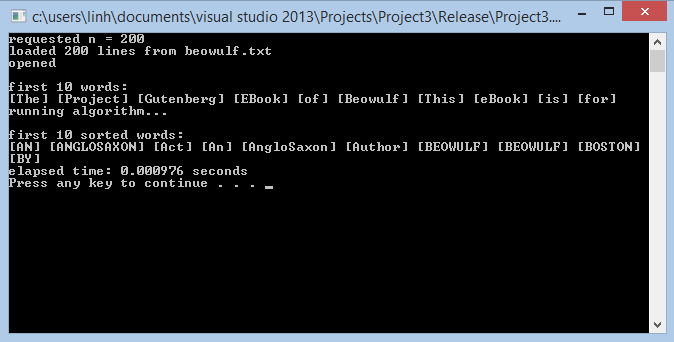
2. Two scatter plots:

a. One showing the run time of all three implementations, zoomed out so that the

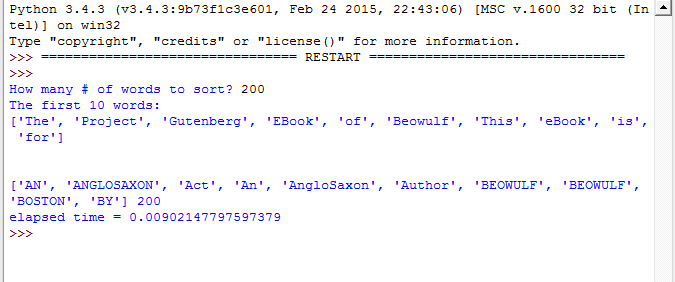
quadratic curves are clear.

b. One zoomed in to show the crossover point.

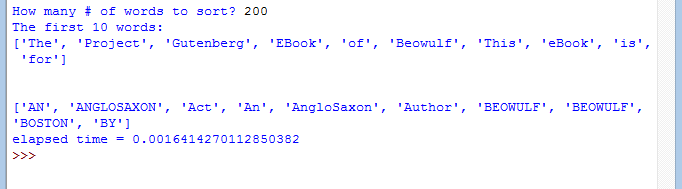
1. An output for *n* = 200 for all three implementations.

**C++**

**Python bubble Sort**



**Python merge sort**



1. Your complete Python source code for implementations 1 and 3.

###############################################################################

# CPSC 335 Project 3

# Spring 2015

#

# Authors: <Kourun Sok, Linh Cao>

###############################################################################

import time

def bubblesort(array):

for x in range(len(array)):

for y in range(len(array)-1):

if array[x] < array[y]:

#swap

array[x],array[y] = array[y],array[x]

return array

def main():

#open the file

f = open('beowulf.txt','r')

temp = f.read()

#grab the text and put into a list so we can sort

text = []

new\_list= []

for i in temp.split():

text.append(i)

n = input('How many # of words to sort? ')

lengthinput = int(n)

print ('The first 10 words: \n' + str(text[:10]) + '\n\n')

#Setting time , code from the professor stub code 2

start = time.perf\_counter()

#since python start from 0 ,[:n] => 0->n-1

new\_list= bubblesort(text[:lengthinput])

end = time.perf\_counter()

#print the first 10 word as a quick check to see if it is correct

print(new\_list[:10] ,len(new\_list))

#print it takes to run the algorithms

print('elapsed time = ' + str(end - start))

if \_\_name\_\_ == '\_\_main\_\_':

main()

###############################################################################

# CPSC 335 Project 3

# Spring 2015

#

# Authors: <Kourun Sok, Linh Cao>

###############################################################################

import time

#implementation from the lecture notes

def merge\_sort(array):

if len(array) <=1:

return array

else:

#spilt the array in half

half = len(array) // 2

return merge(merge\_sort(array[:half]),merge\_sort(array[half:]))

def merge(a,b):

s=[]

ai = 0

bi = 0

while ai <len(a) and bi < len(b):

if a[ai] <= b[bi]:

x = a[ai]

ai +=1

else:

x = b[bi]

bi+=1

s.append(x)

return s + a[ai:] + b[bi:]

def main():

#open the file

f = open('beowulf.txt','r')

temp = f.read()

#grab the text and put into a list so we can sort

text = []

new\_list= []

for i in temp.split():

text.append(i)

n = input('How many # of words to sort? ')

lengthinput = int(n)

print ('The first 10 words: \n' + str(text[:10]) + '\n\n')

#Setting time , code from the professor stub code 2

start = time.perf\_counter()

#since python start from 0 ,[:n] => 0->n-1

new\_list= merge\_sort(text[:lengthinput])

end = time.perf\_counter()

#print the first 10 word as a quick check to see if it is correct

print(new\_list[:10])

#print it takes to run the algorithms

print('elapsed time = ' + str(end - start))

if \_\_name\_\_ == '\_\_main\_\_':

main()

1. Your complete source code (in C++ or similar) for implementation 2.

// CPSC 335 - assignment 3

// Linh Cao,Kourun Sok

#include <chrono>

#include <iostream>

#include <fstream>

#include <string>

#include <istream>

using namespace std;

const int WORD\_NUM = 20000;

void bubleSort(string arr[], int size)

{

string temp;

for (int i = 0; i < size -1; i++)

{

for (int j = 0; j<size; j++)

{

//Swapping element in if statement

if (arr[i]<arr[j])

{

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

}

int main(int argc, char\*\* argv) {

// read strings into array

ifstream fin;

fin.open("beowulf.txt");

string myArray[WORD\_NUM];

string temp;

int n;

cout << "requested n = ";

cin >> n;

cout << "loaded "<<n<<" lines from beowulf.txt\n";

// read strings from file into array

if (fin.is\_open())

{

cout << "opened\n";

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

{

getline(fin, myArray[i]);

}

}

// display first 10 words

cout << "\nfirst 10 words:\n";

for (int i = 0; i < 10; i++)

{

cout << "[" << myArray[i] << "] ";

}

cout << endl;

cout << "running algorithm..." << endl;

auto start = chrono::high\_resolution\_clock::now();

bubleSort(myArray, n);

auto end = chrono::high\_resolution\_clock::now();

// display first 10 sorted words

cout << "\nfirst 10 sorted words:\n";

for (int i = 0; i < 10; i++)

{

cout << "[" << myArray[i] << "] ";

}

cout << endl;

int microseconds = chrono::duration\_cast<chrono::microseconds>(end - start).count();

double seconds = microseconds / 1E6;

cout << "elapsed time: " << seconds << " seconds" << endl;

system("pause");

return 0;

}

6. Answers to the following questions, using complete sentences.

**a. Which *O*(*n*2) -time algorithm did you choose to implement, and why?**

Answer: We chose Bubble sort since it is the most fundamental and well known Algorithm and it is one of the easiest to implement.

**b. Which *O*(*n* log *n*) -time algorithm did you choose to implement, and why?**

Answer: We chose the Merge sort, since we want to implement the code from the lecture notes and get a fully handout approach to work on it.

**c. Which of the three implementations did you find most difficult, and why?**

**1 Recall that, as stated on the syllabus, you may work in a group of up to three students.**

Answer: The merge sort was more difficult since bubble sort was straight forward to implement. Merge sort it has to call a merge function and also in python I had problem with dividing “int” by half which requires to use the “//” instead of one “/” .

**d. Are your empirical results consistent or inconsistent with hypothesis 1? In other**

**words, do the run times of both implementation 1 and 2 fit quadratic curves?**

Answer: Yes It is consistent with the Hypothesis 1. The graph starts slow and then show a quadratic curves.

**e. Are your empirical results consistent or inconsistent with hypothesis 2? In other**

**words, are the run times of your implementation 1 greater than those of**

**implementation 2 by a constant factor? If so, approximately what is that factor, as**

**a percentage? Does this result surprise you?**

Answer: Yes it is consistent with the hypothesis 2. The implementation of c++ is faster than the implementation of python using the same algorithm. It doesn’t really surprise us to say c++ faster than python since python is a higher level languages. There is a 27.9004% difference for data point 400.

**f. Are your empirical results consistent or inconsistent with hypothesis 3? In other**

**words, is there a crossover point for which implementation *nc* 3 is faster than**

**implementations 1 and 2? If so, what is the approximate value of *nc* ? How much**

**faster is implementation 3 over implementation 2, as a percentage, for the full**

***n* = 40, 707? Does this result surprise you?**

Answer: Yes , it is consistent with the hypothesis 3. The last implementation is faster than both bubble sort in python and c++ implementation. The crossover seems to be when n is

**g. Based on these results, which approach do you think is a better way of**

**implementing algorithms efficiently: implementing a slow algorithm in a fast low**

**level language (implementation 2), or implementing a fast algorithm in a slow**

**high level language (implementation 3)? Why? What are the implications on**

**software development in general?**

Answer: Base on the result , it seems to be implementing a fast algorithm in a high level language (implementation 3) since the data shown that at a certain point , there will be a crossover that makes the fast algorithm in higher language faster than a lower level algorithm. Which at the beginning the implementation 3 was slower than implementation 2. The software development in general would be to use the best algorithm that fit best for the type of application and data point they deal with.