COA Prog1 B25

By: Christian Sieh

Analysis

	tape_sort	Sort1	Sort2
	$n^2 + n(\frac{n}{8})$	$n + \sum_{k=1}^{n}(k + \frac{k}{8})$	$2n + 2\left(\sum_{k=1}^{n/2}(k + \frac{k}{8})\right) + 2n$
N = 8	72	52	54.5
N = 16	288	169	145
N = 32	1152	626	434

tape_sort: This equation is the one you gave us in class
sort1: This equation is made by using the firstPass function which reads through the file so n and then we are reading/writing n times and then rewinding while decrementing 1 after each time so a summation.
sort2: This equation is take from sort1 but we split the list of numbers in half and do it twice. The first 2n is from reading the list of number and then writing the list of numbers. The summation is the summation from sort1 but with half the number of values since we split the list. We then multiply this summation by 2 since we ran it once on each half of the numbers. Finally, the last 2n is when we combine the two tapes into the original tape so we have to read all the numbers and write them to the main tape.

Based on this analysis sort1 is better for all sizes less than 8 while sort2 is better for sizes greater than 8. This is because the reads and writes of splitting the list in sort2 is not worth the benefit of dividing such a small list in half.

Function Description

Sort1.cpp

```
// Program: B25 Program
// Class: Computer Organization & Architecture
// Professor: Dr. Karlsson
// Date: 2/19/2016
// Author: Christian Sieh
// Usage: sort1 tape.bin
//
// This program will open the file specified with argv[1] and
// then sort the numbers. This is accomplished by first callling
// the function firstPass which will get the length of the numbers
// in the file, the last number (lastValue) from the file, the
// highest number from the file, and the index for the highest
// number. Next the program calls sort which will use these
// values to increment through the file a length number of times
// and write the highest number at the end of the file each time.
```

```
// While incrementing through the numbers in the file it will
// also write the last value in the highIndex, essentially swapping
// the high value and the last value. Once all this is done
// wholeTape is called to write out the sorted file to console out
```

Sort2.cpp

```
// Program: B25 Program
// Class: Computer Organization & Architecture
// Professor: Dr. Karlsson
// Date: 2/19/2016
// Author: Christian Sieh
// Usage: sort2 tape.bin
//
// This program will open the file specified with argv[1] and
// then sort the numbers. This is accomplished by first
// opening two other tapes tape2.bin and tape3.bin. Then the
// function splitTape is called to take the numbers in tape
// and write half of them to tape2 and the other half to tape3.
// After that firstPass and sort are called on tape2 and then
// on tape3 to end up with two sorted lists in each tape.
// Finally, these two tapes are combined back together
// in tape and in order to give us a fianl sorted
// file of numbers.

//This function will take tape and put half of the numbers in tape2
//and the other half of the numbers in tape3. It will then
//rewind tape so we are at the beginning of the file
void splitTape(fstream& tape, fstream& tape2, fstream& tape3, char *arg)

//This function will read from tape2 and tape3 and write
//out the values in ascending order to tape. This will
//end up giving us a sorted array of numbers on tape.
void combineTape(fstream& tape, fstream& tape2, fstream& tape3,
     char* arg, int length)

// This function is called before the sort function. This function
// gets the last value, the high value, the high value's index,
// and the length of the file which are all needed before we can
// begin sorting
void firstPass(fstream& tape, int& length, int& lastValue, int& highValue,
     int& highIndex, char *arg)

//This is the main function that will do all of the sorting.
//It will take in a file, the length of numbers in the file, the
//last value in the list of numbers, the highest value, the highest
//value's index, and the name of the tape. This function starts
//by assigning sortedIndex which we will use to keep track of how
```

//many numbers we have written. It starts at length and then will
//decrement once a length number of numbers are written which is
//how we will know to stop looping. Index is used to keep track of
//how far we have read or write into the file. This is used so we
//can get the next swapIndex and it's nicer to look at instead
//of putting everything in another for loop. The for loop will
//iterate through the list of numbers, if the swapIndex is found
//then it will write out the lastValue and then the if statement
//after the write is used in case the lastValue we just wrote would
//end up being the highValue for this pass through the file. Else
//if it wasn't the swap index we will read the next number and then
//if the next number is larger then the current high number for this
//read we will set that as the new high number. Once we are up to
//sortedIndex - 1 through the file we then write the high number.
//The if statement after the write is to stop the function from
//accidently creating two of the same number which in turn would
//break the sort. Once sortedIndex is 0 then we have sorted all the
//numbers in the file.
void sort(fstream& tape, int& length, int& lastValue, int& highValue,
int swapIndex, char* arg)

// This function will read and cout values up to length
// for the file name specified
void wholeTape(fstream& tape, int& length, char* arg)

// Author: Dr. Karlsson
// Function to read a single element from the tape
int read(fstream& tape)

// Author: Dr. Karlsson
// Function to write a single element to the tape
void write(fstream& tape, int data)

// Dr. Karlsson
// Function to rewaind the tape to the beginning
void rewind(fstream& tape, char *arg, int& index)