main.cpp

/\*\*

\* FileName: main.cpp

\* Author: James Muoghalu (j286m692@ku.edu)

\* Date: 3/11/18

\* Description: driver file for program

\*/

#include <string>

#include <iostream>

#include <fstream>

#include "Hakimi.h"

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

{

Hakimi\* program = new Hakimi();

bool continue\_loop = true;

int input\_choice = 0;

while(continue\_loop)

{

std::cout << "\n.....................................................\n"

<< "Pick One of the Following Options for Running Hakimi-Havel's Algorithm:"

<< "\n1) Read from a Text File"

<< "\n2) Exit Program"

<< "\nYour Choice (1 or 2): ";

std::cin >> input\_choice;

std::cout << std::endl;

if(input\_choice < 1 || input\_choice > 2)

{

std::cout << "Invalid Input" << std::endl;

}

else if(input\_choice == 1)

{

std::string file\_name = "";

std::string seq\_input = "";

std::cout << "Provide the Name of the Input File: ";

std::cin >> file\_name;

std::ifstream input\_file(file\_name);

while(std::getline(input\_file, seq\_input))

{

// if there are empty lines in the input file

if(seq\_input == "\n" || seq\_input == "\r")

{

continue;

}

delete program->sequence;

program->sequence = new std::vector<int>();

delete program->original\_sequence;

program->original\_sequence = new std::vector<int>();

bool valid\_sequence = true;

std::string sub = "";

for(std::size\_t i = 0; i < seq\_input.length() ; i++)

{

// read the next digit character

if(std::isdigit(seq\_input[i]))

{

sub += seq\_input[i];

}

// a number has been read for the sequence

else if(seq\_input[i] == ' ' && sub != "")

{

program->original\_sequence->push\_back(std::stoi(sub));

program->sequence->push\_back(std::stoi(sub));

sub = "";

}

// an invalid string appears in the file

else if(seq\_input[i] != '\n' && seq\_input[i] != '\0' && seq\_input[i] != '\r')

{

valid\_sequence = false;

break;

}

}

if(valid\_sequence && sub != "")

{

program->original\_sequence->push\_back(std::stoi(sub));

program->sequence->push\_back(std::stoi(sub));

}

if(!valid\_sequence)

{

std::cout << "\n--------------------------------------------------------" << std::endl;

std::cout << "Error: All values in the sequence must be non-negative integers" << std::endl;

std::cout << "--------------------------------------------------------\n" << std::endl;

}

// this line of input is valid, so the algorithm will run

else

{

program->quickSort(program->sequence, 0, (program->sequence->size()-1));

program->quickSort(program->original\_sequence, 0, (program->original\_sequence->size()-1));

std::cout << "\n--------------------------------------------------------" << std::endl;

std::cout << "Original Sequence: " << program->original\_sequence->at(0);

for(std::size\_t i = 1; i < program->original\_sequence->size(); i++)

{

std::cout << " " << program->original\_sequence->at(i);

}

std::cout << std::endl;

// check if the degree of the highest-degree vertex is greater than the order of the graph

if( program->original\_sequence->at(0) > (int)program->original\_sequence->size() )

{

std::cout << "Result: The sequence is not graphical." << std::endl;

}

else

{

bool answer = program->runAlgorithm();

if(answer)

{

std::cout << "Result: The sequence is graphical.\n" << std::endl;

}

else

{

std::cout << "Result: The sequence is not graphical." << std::endl;

}

}

std::cout << "--------------------------------------------------------\n" << std::endl;

}

}

}

else if(input\_choice == 2)

{

std::cout << "Goodbye." << std::endl;

continue\_loop = false;

}

std::cout << std::endl;

}

delete program;

return(0);

}

Hakimi.cpp

/\*\*

\* FileName: Hakimi.cpp

\* Author: James Muoghalu (j286m692@ku.edu)

\* Date: 3/11/18

\* Description: implementation file for Hakimi class

\*/

#include "Hakimi.h"

#define PDEBUG 0 // set to 0 or 1

/\*

#if PDEBUG == 1

#endif

\*/

Hakimi::Hakimi()

{

this->sequence = nullptr;

this->original\_sequence = nullptr;

}

Hakimi::~Hakimi()

{

delete this->sequence;

delete this->original\_sequence;

}

bool Hakimi::runAlgorithm()

{

bool answer = true;

bool end = false;

#if (PDEBUG == 1)

int reduction\_number = 1;

#endif

while(!end)

{

int first = this->sequence->at(0);

std::vector<int>\* next\_iteration = new std::vector<int>();

std::size\_t i = 1;

// perform the algorithm arithmetic

for(; i <= (std::size\_t) first; i++)

{

next\_iteration->push\_back((this->sequence->at(i))-1);

}

for(std::size\_t j = i; j < this->sequence->size(); j++)

{

next\_iteration->push\_back(this->sequence->at(j));

}

quickSort(next\_iteration, 0, (next\_iteration->size()-1));

bool found\_non\_zero = false;

for(std::size\_t k = 0; k < next\_iteration->size(); k++)

{

if(next\_iteration->at(k) > 0)

{

found\_non\_zero = true;

break;

}

}

if(found\_non\_zero) // continue to the next iteration of the while loop

{

delete this->sequence;

this->sequence = next\_iteration;

#if (PDEBUG == 1)

std::cout << "\nReduction " << reduction\_number++ << ": " << this->sequence->at(0);

for(std::size\_t i = 1; i < this->sequence->size(); i++)

{

std::cout << " " << this->sequence->at(i);

}

std::cout << std::endl;

#endif

}

else // else, ignore the new reduction and determine if this->sequence is graphical

{

end = true;

delete next\_iteration;

}

}

if(this->sequence->at(0) >= (int) this->sequence->size())

{

return false;

}

int sum = 0;

for(std::size\_t i = 0; i < this->sequence->size(); i++)

{

sum += this->sequence->at(i);

}

if(sum % 2)

{

return false;

}

else

{

return true;

}

return answer;

}

// sort the given sequence

void Hakimi::quickSort(std::vector<int>\* seq\_to\_sort, std::size\_t low, std::size\_t high)

{

if(low < high && low >= 0 && high < seq\_to\_sort->size())

{

int pivot\_index = partition(seq\_to\_sort, low, high);

#if (PDEBUG == 1)

//std::cout << "\t\t\tPivot Index: " << pivot\_index << std::endl;

#endif

quickSort(seq\_to\_sort, low, (pivot\_index-1));

quickSort(seq\_to\_sort, (pivot\_index+1), high);

}

}

// quickSort helper function

int Hakimi::partition(std::vector<int>\* seq\_to\_sort, std::size\_t low, std::size\_t high)

{

int pivot = seq\_to\_sort->at(high);

std::size\_t i = low-1;

for(std::size\_t j = low; j < high; j++)

{

if( seq\_to\_sort->at(j) > pivot)

{

i++;

int temp = seq\_to\_sort->at(i);

seq\_to\_sort->at(i) = seq\_to\_sort->at(j);

seq\_to\_sort->at(j) = temp;

}

}

int temp = seq\_to\_sort->at(i+1);

seq\_to\_sort->at(i+1) = seq\_to\_sort->at(high);

seq\_to\_sort->at(high) = temp;

return i+1;

}

Hakimi.h

/\*\*

\* FileName: Hakimi.h

\* Author: James Muoghalu (j286m692@ku.edu)

\* Date: 3/11/18

\* Description: header file for Hakimi class

\*/

#ifndef HAKIMI\_H

#define HAKIMI\_H

#include <iostream>

#include <vector>

#include <ctype.h>

#include <fstream>

class Hakimi

{

public:

/\*

\* @brief

\* @param

\* @return

\*/

Hakimi();

~Hakimi();

bool runAlgorithm();

void quickSort(std::vector<int>\* seq\_to\_sort, std::size\_t low, std::size\_t high);

int partition(std::vector<int>\* seq\_to\_sort, std::size\_t low, std::size\_t high);

std::vector<int>\* sequence;

std::vector<int>\* original\_sequence;

};

#endif

Makefile

program := hakimi

directory := HomeworkExtra

submission := Muoghalu\_ExtraCredit

$(program): main.o Hakimi.o

g++ -std=c++11 -g -Wall $^ -o $(program)

main.o: main.cpp

g++ -std=c++11 -g -Wall -c main.cpp

Hakimi.o: Hakimi.cpp

g++ -std=c++11 -g -Wall -c Hakimi.cpp

clean:

rm \*.o $(program)

echo clean done

clean2:

rm \*.zip \*.tar.gz

echo clean done

tar:

mkdir $(submission)

rsync --exclude=$(submission) --exclude=\*.tar.gz --exclude=\*.zip ../$(directory)/\* $(submission)

tar cvzf $(submission).tar.gz $(submission)

rm -rf $(submission)

zip:

mkdir $(submission)

rsync --exclude=$(submission) --exclude=\*.tar.gz --exclude=\*.zip ../$(directory)/\* $(submission)

zip -r $(submission).zip $(submission)

rm -rf $(submission)

data.txt

6 4 4 4 4 3 3 3 2 2 1

4 4 3 2 1 2 6 3 4 4 3

4 4 4 4 4 4 4

1 2 2 1 4 1 1

1 3 1 1 1 1 4

1 1 1 1 6 1 1

2 2 3 3 3 3 2 2

3 3 2 2 4 2

2 1 -1

8 1 5 3 a 4 33

50 9 8 1