Justin Phillips

CS 325 spring 2019

HW 1–30 points

1)(6pts) For each of the following pairs of functions, either f(n) is O(g(n)), f(n) is Ω(g(n)), or f(n)

is Θ(g(n)) best describes the relationship.

Select one and explain

.

a. f(n) = n.25; g(n) = n.5

limit method:

if lim→∞f(n)/g(n)

lim→∞n^.25/n^.5 = 1/n^.25 = 0

we conclude f(n) is 0(g(n) since we got a constant divided n^.25 the closer we approach infinity the closer to zero we get.

b. f(n) = log n2; g(n) = ln n

limit method:

if lim→∞f(n)/g(n)

lim→∞log n^2/ln n = a positive constant

f(n) is Θ(g(n))

all logs have the same growth rate no matter the base is

c. f(n) =nlog n+ n2; g(n) =n√n

limit method:

if lim→∞f(n)/g(n)

lim_(n-> ± ∞) log(n + n^2)/sqrt(n) = 0

f(n) = O(g(n)

d. f(n) = e2; g(n) =2n

limit method:

if lim→∞f(n)/g(n)

lim_(n->∞) 2^(-n) e^2 = 0

f(n) = O(g(n)

e. f(n) = 2n; g(n) = 2n+1

limit method:

if lim→∞f(n)/g(n)

2^n/2^(n + 1)= ½ which is a constant therefore

f(n) is Θ(g(n))

f. f(n) =nn; g(n) = n!

limit method:

if lim→∞f(n)/g(n)

n^n/(n!)

factorial has a faster rate of growth then compared the exponential

f(n) = O(g(n)

2)(4pts)

Let f1 and f2 be asymptotically positive non-decreasing functions. Prove or disprove each

of the following conjectures.To disprove give a counter example.

a.If f1(n)=Θ(g(n)) and f2(n)=Θ(g(n)) then f1(n)=Θ(f2(n)).

1. by definition f1(n)=Θ(g(n)) implies there exist positive constants c\_1, c\_2 and n\_0 such that

0 ≤ c\_1g(n) ≤f\_1(n) ≤ c\_2g(n) for all n ≥ n\_0

2.by defintion f2(n)=Θ(g(n)) )) implies there exist positive constants c\_3, c\_4 and n\_1 such that

0 ≤ c\_3g(n) ≤f\_2(n) ≤ c\_4g(n) for all n ≥ n\_1

3. show f1(n)=Θ(f2(n)) that is there exist a positive constants c\_5, c\_6 and n\_2 such that

0 ≤ c\_5f\_2(n) ≤f\_1(n) ≤ c\_4f\_2(n) for all n ≥ n\_2

by combining 1 and 2: (c\_1)( c\_3)f\_2(n) ≤ c\_1g(n) ≤ f\_1(n) let c\_5 = (c\_1)(c\_3) so (c\_5)f\_2(n) ≤ f1(n), again from 1 and 2. f\_2(n) ≤ (c\_2)(c\_4)g(n) ≤ c\_2g(n) ,let c\_6 = (c\_2)(c\_4) so (c\_6)f\_2(n) ≤ f1(n) and let n\_2 = max{n\_0,n\_1}

f1(n)=Θ(f2(n)) is TRUE

b. If f1(n)=O(g1(n)) and f2(n)= O(g2(n)) then (f1(n)/f2(n))=O(((g1(n))/(f2(n)))

counter example is f1(n) = n, f2(n)=n2, g1(n)=n3, g2(n)=n4

n = O(n3), n2 = O(n4) then

n/n2= O(n3/n2) = 1/n = O(n) which if we multiply both sides by n we get n= O(n2) which does not equal n = O(n3)

so then (f1(n)/f2(n))=O(((g1(n))/(f2(n))) is NOT TRUE

4a. insertTime.cpp

#include <iostream>

#include <fstream>

#include <ctime>

#include <time.h>

#include <cstdlib>

/\*\*\*\*Reference: https://www.geeksforgeeks.org/insertion-sort/ \*\*\*\*/

void insertSort(int myArray[], int size)

{

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

{

int key = myArray[i]; //key is the value we are trying to find

int j = i - 1; //j is the value we are comparing key

while (j >= 0 && myArray[j] > key)

{

myArray[j + 1] = myArray[j]; //move it one position ahead

j = j - 1; //decreemnt it to look at the one below

}

myArray[j + 1] = key;

}

}

void arrayFill(int myArray[], int size) {

//int newArr[10];

//int n = 5000;

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

myArray[i] = rand() % 10000 + 1;

std::cout << myArray[i] << std::endl;

}

}

void printArr(std::ostream &stream, int arr[], int n) {

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

stream << arr[i] << " ";

}

stream << arr[n - 1];

stream << std::endl;

}

int main() {

std::ofstream timeFile;

timeFile.open("times.txt");

srand(time(NULL));

clock\_t t1, t2;

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

int n = rand() % 100000 + 1; //size of array

float timer = 0;

float diff = 0;

float seconds = 0;

int myArray[n];

arrayFill(myArray, n);

t1 = clock();

insertSort(myArray, n);

t2 = clock();

diff = t2 - t1;

std::cout << "Array Size " << n << std::endl;

std::cout << "difference " << diff << std::endl;

seconds = diff / CLOCKS\_PER\_SEC;

std::cout << "Seconds " << seconds << std::endl;

timeFile << n << "size " << seconds << "seconds. " << std::endl;

}

timeFile.close();

return 0;

}

mergeTime.cpp

#include <iostream>

#include <fstream>

#include <ctime>

#include <time.h>

#include <cstdlib>

/\*Code citation: https://www.sanfoundry.com/cpp-program-implement-merge-sort/ \*/

void merge(int \*array, int low, int high, int mid)

{

int i;

int j;

int k;

int temp[high - low + 1];

i = low; //start

k = 0; //first spot in temp

j = mid + 1; //start of second

// Merge

while (i <= mid && j <= high) //check if valid

{

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

{

temp[k] = array[i];

k++;

i++;

}

else

{

temp[k] = array[j];

k++;

j++;

}

}

while (i <= mid)// Insert all the remaining values

{

temp[k] = array[i];

k++;

i++;

}

while (j <= high) // insert the rest

{

temp[k] = array[j];

k++;

j++;

}

for (i = low; i <= high; i++)//Transfer the data from temp

{

array[i] = temp[i - low];

}

}

void mergeSort(int \*array, int low, int high)

{

if (low < high)

{

int mid = (low + high) / 2;

mergeSort(array, low, mid);

mergeSort(array, mid + 1, high);

merge(array, low, high, mid);

}

}

void arrayFill(int myArray[], int size) {

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

myArray[i] = rand() % 10000 + 1;

std::cout << myArray[i] << std::endl;

}

}

void printArr(std::ostream &stream, int arr[], int n) {

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

stream << arr[i] << " ";

}

stream << arr[n - 1];

stream << std::endl;

}

int main() {

std::ofstream timeFile;

timeFile.open("mergetimes.txt");

srand(time(NULL));

clock\_t t1, t2;

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

int n = rand() % 100000 + 1; //size of array

float timer = 0;

float diff = 0;

float seconds = 0;

int myArray[n];

arrayFill(myArray, n);

t1 = clock();

mergeSort(myArray, 0, n - 1);

t2 = clock();

diff = t2 - t1;

std::cout << "Array Size " << n << std::endl;

std::cout << "difference " << diff << std::endl;

seconds = diff / CLOCKS\_PER\_SEC;

std::cout << "Seconds " << seconds << std::endl;

timeFile << n << "size " << seconds << "seconds. " << std::endl;

}

timeFile.close();

return 0;

}

4b. collect running times

insertsort

|  |  |
| --- | --- |
| size | second |
| 24056 | 0.78 |
| 27378 | 1.09 |
| 31758 | 1.59 |
| 39158 | 1.83 |
| 51669 | 3.39 |
| 54456 | 3.82 |
| 60961 | 4.54 |
| 70627 | 6.06 |
| 82451 | 8.02 |
| 90790 | 9.89 |

log insert

|  |  |
| --- | --- |
| size | second |
| 4.165689 | -0.10791 |
| 4.437402 | 0.037426 |
| 4.501853 | 0.201397 |
| 4.592821 | 0.262451 |
| 4.71323 | 0.5302 |
| 4.736046 | 0.582063 |
| 4.785052 | 0.657056 |
| 4.848971 | 0.782473 |
| 4.916196 | 0.904174 |
| 4.958038 | 0.995196 |

merge sort

|  |  |  |
| --- | --- | --- |
| size | seconds | |
| 14645 | 0.01 | |
| 43332 | 0.02 | |
| 45107 | 0.03 | |
| 54064 | 0.03 | |
| 61499 | 0.03 | |
| 65029 | 0.03 | |
| 76840 | 0.03 | |
| 80185 | 0.04 | |
| 81965 | 0.04 | |
| 92986 | 0.04 | |
| log merge | | | |
| size | | second | |
| 4.165689 | | -2 | |
| 4.636809 | | -1.698970004 | |
| 4.654244 | | -1.522878745 | |
| 4.732908 | | -1.522878745 | |
| 4.788868 | | -1.522878745 | |
| 4.813107 | | -1.522878745 | |
| 4.885587 | | -1.522878745 | |
| 4.904093 | | -1.397940009 | |
| 4.913628 | | -1.397940009 | |
| 4.968418 | | -1.397940009 | |

4c.

4d.

4e. Theoretical running time for insertion sort is O(n2) and my formula

y= 1.203x2-9.5314x+18.702 proves it is O(n2)

Theoretical running time for merge sort O(n lg n) and my formula is

y = 3.423ln(x) – 6.8779 which fits with the theoretical