Compute the runtime of the following recursive function that returns the nth Fibonacci number.

1 1 2 3 5 8 13 … fib(n)=fib(n-1)+fib(n-2)

{1 if n=1, 2

Fib {fib(n-1)+fib(n-2) if n>2

#include <iostream>

#include <ctime>

Using namespace std;

Long int Fib(int n)

{

If(n==1) || (n==2) return 1;

Else return Fib(n-1)+Fib(n-2);

}

Int main()

{

int start=clock();

cout<<”This is the 42 Fib number:” <<Fib(42);

cout<<endl;

int finish-clock();

cout<<”Run-time=”(finish-start)/1000; //1000 bec of the time counted

cout<<endl;

system(“pause”);

return 0;

}

//FOR QUESTION 1 FROM PROJECT 2

<string> <cstring>

String state1=”California”; char state2[20]=”California”;

0 1 2 3 4 5 6 7 8 9 10

C a l I f o r n i a \0

cout<<state; //both of them are the same we’ll see California

cout<<state[4]; //see f

cout<<state2;//california

cout<<state2[4]; //f

cout<<state1.lrngth(); //10

cout<<strlen(state2); //we use cstring and we will see length of that string it does the same thing as above we also will see 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T | e | x | a | s |

state1 = “Texas”;

strcopy(state2, “texas”); //state2 is now Texas as well

string states1[4]={“California”, “Texas”, “Iowa”, “Idaho“};

char states2[4][15]={“California”,”Texas”,”Iowa”,”Idaho”}; //4 states, maximum length is 15

[0][1][2][3][4][5][6][7][8][9][10]

[0] C a l i f o r n I a \0

[1] T e x a s \0

[2] I o w a \0

[3] I d a h o \0

cout<<state[0]; //California

cout<<state2[0]; //California

//lets change T of texas to lowercase

state1([1][0])=tolower(~~state[1][~~0]);

state2([1][0])=tolower(state2[1][0]); //same for state1 change //and state2

//display the length of California in both arrays

//California is in row 0

cout<<state1[0].length();

cout<<strlen(state2[0]); //bec we talking bout 2 diff functions this one is from cstring and uses strlen

//for project 2 homework 3-b

trace

int f(int n, int m)

{

if(n==m) return 1;

else return n+f(n,m-1);

}

cout<<f(2, 5) //what is f of 2 and 5

2+f(2,4); //now is 2==4? from the function above

2+f(2, 3) //is 2==3? no so we return 2+f(2,2)

2+f(2,2) // is 2==2? yes so we return 1

1

the final ourput is 7 bec of the waiting values

cout<<f(~~2, 5~~ 7) //what is f of 2 and 5

2+f(~~2,~~4 5); //now is 2==4? from the function above

2+f(~~2, 3~~ 3) //is 2==3? no so we return 2+f(2,2)

2+f(~~2,2~~ 1) // is 2==2? yes so we return 1

1

//computing the run-time of functions on paper.

Ex1 Compute the runtime of the following if it takes q millisecond to execute cout statement

Assume n is large

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

{

cout<<i;

}

runtime=q+q+q+q+q+q+q. . . .+q //altogether is nq

in mathematics

if we have the

lim(n^2+5n)

n->∞

lim(n^2+5n)=lim n^2

n->∞

in programming

same concept but diff notation

n^2+5 = O(n^2) //big O notation

read n^2+5 is Big-oh of n^2

Rule 1

an^k+bn^(k-1)+. . . = O(n^k)

a>0

ex. 2n^5 + 3n^3 – 2n= O(n^5) //term with the highest power

(n+1)^5 = n^5+ . . . . = O(n^5)

(n+2)^3 (n-1)^4 = //if w eraise it to the power we get more terms(n^3+. . .)(n^4+. . .)=n^7. . . = O(n^7) //we only care about the highest power

this is Rule 1

Rule 2

notation

(n+2)^5 = O(n^5)

f(n) F(n)

therefore f(n)=O(F(n))

let f(n)=O(F(n))

g(n)=O(G(n))

f(n)+g(n)=O(F(n)+O(G(n))=O(max(F(n),G(n))

f(n).g(n)=O(F(n).O(G(n))=O(F(n).G(n))

example

f(n)=(n-2)^4

g(n)=(n+1)^5

find the big oh estimate of

1. f(n).g(n)

= f(n)=(n-2)^4 = n^4 + . . .= O(n^4)

g(n)=(n+1)^5 = n^5+. . . .= O(n^5)

f(n).g(n)=O(n^4).O(n^5)

=O(n^4 . n^5)

=O(n^9)

1. f(n)+g(n)

= f(n)=(n-2)^4 = O(n^4)

= g(n)=(n+1)^5 = O(n^5)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

f(n) +g(n) = O(n^4) + O(n^5)

=O(max(n^4, n^5))

= O(n^5)

so going back to the runtime with q

runtime=q+q+q+q. . . .+q

= nq //n^1, highest degree is 1.. but q is constant

= nq = O(n)

Example 2

for(i=1; i<=n-10; ++i)

cout<<i;

run-time = q + q + q + q + . . . .+q

=(n-10)q

=qn-10q

say q=2

=2n-20

=O(n)

//it takes about same time for compiler to execute

example 3

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

{

cout<<i; //this is q

cout<<2\*i //this is q

}

runtime = 2q + 2q + 2q + 2q + . . .+2q

total of 2 seconds

how much is the total?

=2qn //q is fixed, the degree of the polynomial is 1

=2qn = O(n)

example 4

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

cout<< i;

for(j=1; j<=n^2 ; ++j

cout<<j

//what is the total runtime of these two together? the loop will execute n times

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

cout<< i;

for(j=1; j<=n^2 ; ++j O(n^2)

cout<<j

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

runtime = O(n)+O(n^2)

= O(max(n,n^2))

=O(n^2)

//if u have any loop, the max will be the number you execute????!?!?????

Example 5

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

{

for (int j=1; j<n; ++j)

{ cout<<i+j;} O(n) O(n)

}

runtime = O(n).O(n) = O(n.n)

= O(n^2)

use these calling statements for the hw part 3 a and b

1. cout<<sum(5,10);
2. b cout<<g(10);