Zhang, Chunyang

Homework Assignment 1

Instructor Schutfort

June 26, 2015

1) (CLRS) 1.2-2. Suppose we are comparing implementations of insertion sort and merge sort on the same machine. For inputs of size n, insertion sort runs in 8n2 steps, while merge sort runs in 64nlgn steps. For which values of n does insertion sort beat merge sort?

Answer:

From Figure 1, we can see that these two step functions meet in two points. The first point is between n = 1 and n =2, and the section point happens between n = 43 and n = 44. Therefore, insertion sort beats merge sort when 8n2 ≤ 64nlgn, which is 2 ≤ n ≤ 43.

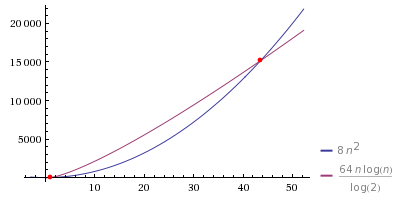


Figure 1: Plot for 8n2 and 64nlgn

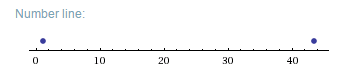


Figure 2: Plot for the two points

2) (CLRS) Problem 1-1 on pages 14-15. Fill in the given table.

*1-1 Comparison of running times*

For each function and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes microseconds.

Answer:

In this question we assume 1 month = 30 days and 1 year = 12 month.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 second | 1 minute | 1 hour | 1 day | 1 month | 1 year | 1 century |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  | 627476 |  |  |  |  |  |  |
|  | 1000 | 7746 | 60000 | 293939 | 1609969 | 5577096 | 55770960 |
|  | 100 | 391 | 1532 | 4420 | 13737 | 31449 | 145973 |
|  | 19 | 25 | 31 | 36 | 41 | 44 | 51 |
|  | 9 | 11 | 12 | 13 | 15 | 16 | 17 |

3) (CLRS) 2.3-3 on page 39. Use mathematical induction to show that when n is an exact power of 2, the solution of the recurrence

is .

Answer:

When , .

When ,

As .

Therefore,

.

Overall, the solution of the recurrence is .

4) For each of the following pairs of functions, either f(n) is O(g(n)), f(n) is Ω(g(n)), or f(n) = Θ(g(n)). Determine which relationship is correct and explain.

a. ;

b.

c.

d.

e.

f.

g.

h.

i.

j.

k.

l.

5) Let f1 and f2 be asymptotically positive functions. Prove or disprove each of the following conjectures.

a. implies .

As , then there exists a constant that

In this case we cannot find another constant that

Therefore, does not imply .

b. If and , then .

As , then there exists a constant that

As , then there exists a constant that

Then we get

Assume and

.

Therefore, .

c. .

If

We know that

and

Therefore, if we set and

Then

.

6) **Fibonacci Numbers:**

The Fibonacci sequence is given by: 0, 1, 1, 2, 3, 5, 8, 13, 21, ….. By definition the Fibonacci sequence starts at 0 and 1 and each subsequent number is the sum of the previous two. In mathematical terms, the sequence Fn of Fibonacci number is defined by the recurrence relation

An algorithm for calculating the nth Fibonacci number can be implemented either recursively or iteratively.

1. Implement both recursive and iterative algorithms to calculate Fibonacci Numbers in the programming language of your choice. Provide a copy of your code with your written homework assignment.

The C code is written as follows:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int recursivefib(int n)

{

if (n == 0)

return 0;

else if (n == 1)

return 1;

else

return (recursivefib(n - 1) + recursivefib(n - 2));

}

long iterativefib(long n)

{

long fib = 0, a = 1, t = 0;

for (long k = 0; k < n; k++)

{

t = fib + a;

a = fib;

fib = t;

}

return fib;

}

int main()

{

clock\_t timer;

/\*recursive algorithm\*/

int n = 30;

int recursivesum;

timer = clock();

recursivesum = recursivefib(n);

timer = clock() - timer;

printf("The %dth number of this Fibonacci Numbers using recursive algorithm is: %d\n", n, recursivesum);

printf("recursive algorithm for n = %d ran in %f seconds\n\n", n, (float)timer / (float)CLOCKS\_PER\_SEC);

/\*iterative algorithm\*/

long n = 10000000;

long iterativesum;

timer = clock();

iterativesum = iterativefib(n);

timer = clock() - timer;

printf("The %ldth number of this Fibonacci Numbers using iterative algorithm is: %ld\n", n, iterativesum);

printf("iterative algorithm for n = %ld ran in %f seconds\n\n", n, (float)timer / (float)CLOCKS\_PER\_SEC);

}

1. Use the system clock to record the running times of each algorithm for n = 5, 10, 15, 20, 30, 50, 100, 1000, 2000, 5000, 10,000. You may need to modify the values of n if an algorithm runs too fast or too slow.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n for recursive algorithm | 25 | 30 | 35 | 40 | 45 | 50 |
| Running time (s) | 0 | 0.01 | 0.17 | 1.97 | 21.92 | 242.360001 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n for iterative algorithm () | 1 | 10 | 100 | 1000 | 2000 | 5000 | 10000 | 20000 | 50000 |
| Running time (s) | 0 | 0.03 | 0.28 | 2.78 | 5.56 | 13.89 | 27.809999 | 55.630001 | 139.009995 |

c) Plot the running time data you collected on graphs with n on the x-axis and time on the y-axis. What type of function (curve) best fits each data set? Discuss the differences in the running times of each algorithm.

The matlab code is written as follows:

n = [0 25 30 35 40 45 50];

t = [0 0 0.01 0.17 1.97 21.92 242.360001];

figure(1);

plot(n,t);

xlabel('n (recursive algorithm)');

ylabel('Running time (s)');

x = [0 1 10 100 1000 2000 5000 10000 20000 50000].\*1e6;

y = [0 0 0.03 0.28 2.78 5.56 13.89 27.809999 55.630001 139.009995];

figure(2);

plot(x,y);

xlabel('n (iterative algorithm)');

ylabel('Running time (s)');

Then we got two plots as follows:

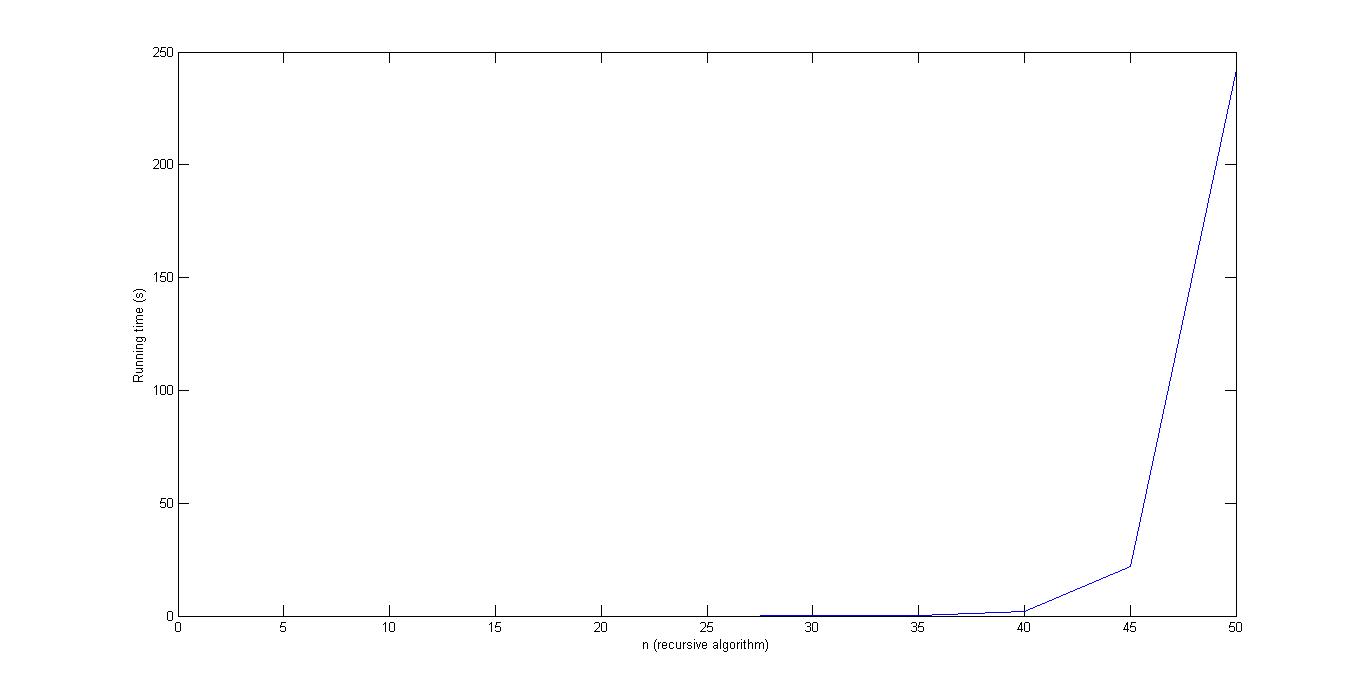


Figure 3: Plot for recursive algorithm

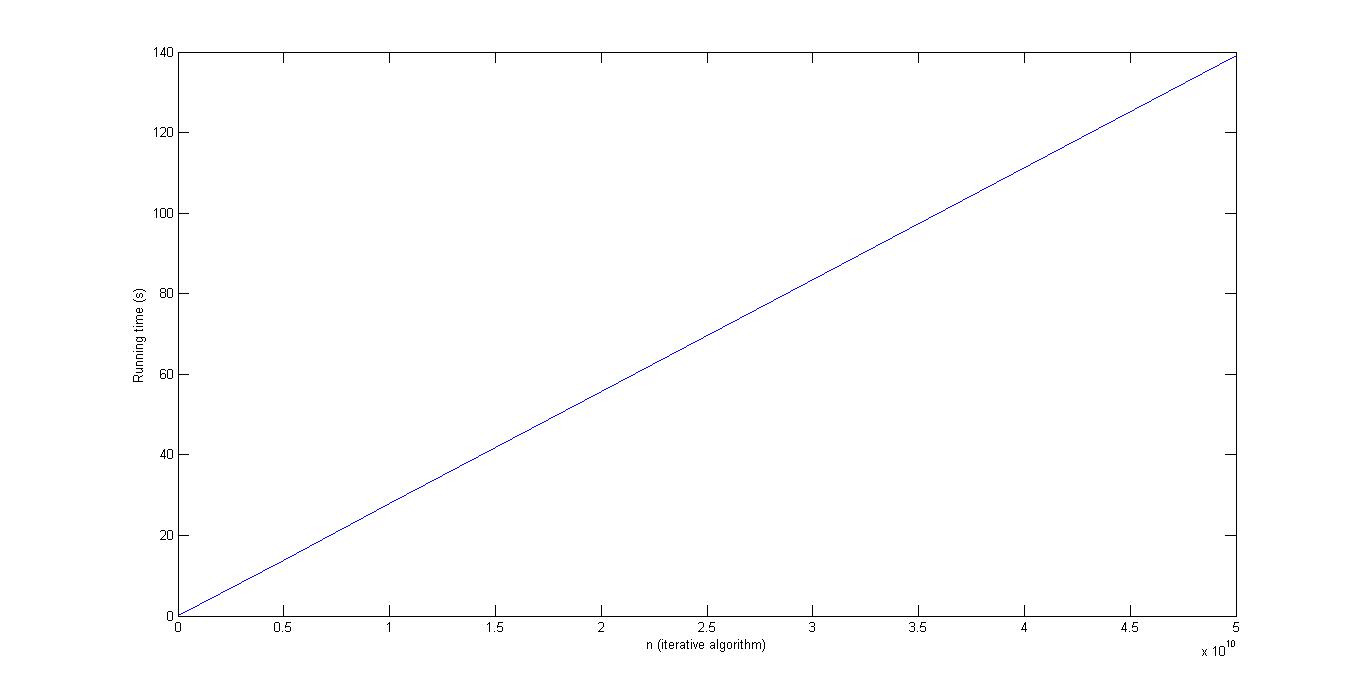


Figure 4: Plot for iterative algorithm

From these data and two graphs, we can easily tell that the recursive algorithm is exponential function, while the iterative algorithm is linear function. Therefore, the recursive algorithm is while the iterative algorithm is . In this case, the iterative algorithm beats the recursive algorithm.

7) Describe a - time algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x. Give an example.

Answer:

First, insert every integer of set S into a binary search tree (), and use in-order tree traversal to sort these values into an array (). Then, pick the first value from the sorted list and calculate . Use binary search algorithm () to find from the rest of the sorted list. If cannot be found in the sorted list, then pick the second value as and calculate again. Look for again and redo the algorithm if it cannot be found until is found in the sorted list. The algorithm for picking the value overall is , therefore the whole algorithm for searching is for the worst case. On the whole, the time for the entire algorithm is .

For instance, given the set S = [4, 6, 8, 0, 10, 2], and x = 12.

The sorted list of S is [0, 2, 4, 6, 8, 10]. First pick , so that .

Use binary search algorithm to find 12, and it cannot be found, therefore we pick , and . Use binary search again and we can find 10 in the list. In this case, the algorithm is completed.

Extra Credit:

For , we get that .

Because ,

then we can assume that

when ,

Therefore, .