Lab 4

Juan Camilo Monterrosa Sanchez

October 2018

1 Complete the following table

Algorithm	Function	Worst case time complexity	Best case time complexity	Average case time complexity	Space complexity
The simplest primality by trial division: Given an input number n, check whether any prime integer m from 2 to √n evenly divides n (the division leaves no remainder). If n is divisible by any m then n is composite, otherwise it is prime.		O(√n)	θ(1)	O(√n)	O(n)
Binary Search	Finds the position of a target value within a sorted array	O(1)	O(1)	O(log n)	O(log n)
Finding the smallest or largest item in an unsorted array		O(n)	O(1)	O(n)	O(n)
Kadane's algorithm	Maximum Sum of Subarray	O(n)	O(n)	O(n)	O(n)
Sieve of Eratosthenes	Find all prime numbers smaller than a given natural number n	O(n(logn)(loglogn))	O(n(logn)(loglogn))	O(n(logn)(loglogn))	O(n)
Merge Sort	Sorting algorithm stable external order based on the divide and conquer technique	O(nlogn)	O(nlogn)	O(nlogn)	O(n)
Heap Sort	Sorting algorithm of non-recursive ordering, not stable	O(nlogn)	O(nlogn)	O(nlogn)	O(1)
Quick Sort	Sort	O(n^2)	O(nlogn)	O(nlogn)	O(n) / O(logn)
Tim Sort	Hybrid stable classification algorithm, derived from the fusion genre and the type of insertion	O(nlogn)	O(n)	O(nlogn)	O(n)
Divide and conquer (Convex Hull)	Find smallest convex set that contains X	O(nlogn)	O(nlogn)	O(nlogn)	O(nlogn)
Insertion Sort	Sort	O(n^2)	O(n) comparisons, O(1) swaps	O(n^2)	O(n) / O(1)
Dijkstra's algorithm	Algorithm for the determination of the shortest route, given a vertex origin, towards the rest of the vertices in a graph that has weights in each edge	O(E+VlogV)	O(V^2)	O(V^2)	O(V^2)
Naive Matrix Multiplication	Executing matrix multiplication	O(n^3/M)	O(n^3/M)	O(n^3/M)	O(V^2)

Floyd–Warshall algorithm	Graph analysis algorithm to find the minimum path in weighted directed graphs.	O(V^3)	O(V^3)	O(V^3)	O(V^2)
Naive Matrix Inversion	Find the Inverse of a Matrix	O(n^3)	O(n^3)	O(n^3)	O(V^2)
Calculate the permutations of n distinct elements without repetitions		O(n^2*n)	O(n^2*n)	O(n^2*n)	O(1)
Calculate the permutations of n distinct elements with repetitions		O(n^2*n!)	O(n^2*n!)	O(n^2*n!)	O(1)

2 Cormen, Leiserson, Rivest and Stein

Exercise 1.2-2:

For insertion sort to beat merge sort for inputs of size n, $8n^2$ must be less than $64nlgn$.

$$8n^2 < 64n \lg n \implies \frac{n}{8} < \lg n \implies 2^{n/8} < n$$

```
n = 2
while 2 ** (n / 8.0) < n:
    n += 1

print n - 1
```

Maximum value of n for which insertion sort beats merge sort is: **43**

Exercise 1.2-3:

```
n = 1
while 100 * n * n > 2 ** n:
    n += 1

print n
```

Minimum value of n for which $100n^2$ runs faster than 2^n is: **15**

Problem 1-1 - solve from 1 microsecond ($10^{-6}s$) for step to for 1 nanoseconds (10^{-9}s) for step. :

Problem 3-1:

Exists a constants $c_1, c_2, n_0 > 0$ such that:

$0 \leq c_1$ (f(n) + g(n)) $\leq \max(f(n), g(n)) \leq c_2$ (f(n) + g(n)) for all n $\geq n_0$.

So for n $\geq$n$_0$, f(n)+g(n) $\geq max(f(n), g(n))$.

Also note that, f(n) $\leq$ max$(f(n), g(n)) and g(n) \leq$ max$(f(n), g(n))$

f(n) + g(n) $\leq 2\max(f(n), g(n))$
$\Rightarrow \frac{1}{2}(f(n) + g(n)) \leq \max(f(n), g(n))$

Therefore, we can combine the above two inequalities as follows:

$0 \leq \frac{1}{2}(f(n) + g(n)) \leq \max(f(n), g(n)) \leq (f(n) + g(n)) for n \geq$n$_0$

So,max$(f(n), g(n)) =$$\Theta$(f(n) + g(n)) because there exists:
$c_1 = 1/2$ and $c_2 = 1$.

3 Dasgupta, Papadimitriou and Vazirani

Exercise 0.1:

	$f(n)$	$g(n)$
(a)	$n - 100$	$n - 200$
(b)	$n^{1/2}$	$n^{2/3}$
(c)	$100n + \log n$	$n + (\log n)^2$
(d)	$n \log n$	$10n \log 10n$
(e)	$\log 2n$	$\log 3n$
(f)	$10 \log n$	$\log(n^2)$
(g)	$n^{1.01}$	$n \log^2 n$
(h)	$n^2/\log n$	$n(\log n)^2$
(i)	$n^{0.1}$	$(\log n)^{10}$
(j)	$(\log n)^{\log n}$	$n/\log n$
(k)	$\sqrt{n}$	$(\log n)^3$
(l)	$n^{1/2}$	$5^{\log_2 n}$
(m)	$n2^n$	3^n
(n)	2^n	2^{n+1}
(o)	$n!$	2^n
(p)	$(\log n)^{\log n}$	$2^{(\log_2 n)^2}$
(q)	$\sum_{i=1}^{n} i^k$	n^{k+1}

a:

The case which matches with the function is: $f(n) = \Theta(g(n))$ and the function can be written as $n - 100 = \Theta(n - 200)$

b:

The case which matches with the function is: $f(n) = O(g(n))$ and the function can be written as $n^{1/2} = O(n^{2/3})$

c:

The case which matches with the function is: $f(n) = \Theta(g(n))$ and the function can be written as $100n + \log n = \Theta\left(n + (\log n)^2\right)$

d:

The case which matches with the function is: $f(n) = \Theta(g(n))$ and the function can be written as $n \log n = \Theta(10n \log 10n)$

e:

The case which matches with the function is: $f(n) = \Theta(g(n))$ and the function can be written as $\log 2n = \Theta(\log 3n)$

f:

The case which matches with the function is: $f(n) = \Theta(g(n))$ and the function can be written as $10\log n = \Theta(\log(n^2))$

g:

The case which matches with the function is: $f(n) = \Omega(g(n))$ and the function can be written as $n^{1.01} = \Omega\ n\log^2 n$

h:

The case which matches with the function is: $f(n) = \Omega(g(n))$ and the function can be written as $n^2 / \log n = \Omega\ n(\log n)^2$

i:

The case which matches with the function is: $f(n) = \Omega(g(n))$ and the function can be written as $n^{0.1} = \Omega\ (\log n)^{10}$

j:

The case which matches with the function is: $f(n) = \Omega(g(n))$ and the function can be written as $(\log n)^{\log n} = \Omega(n / \log n)$

k:

The case which matches with the function is: $f(n) = \Omega(g(n))$ and the function can be written as $\sqrt{n} = \Omega((\log n)^3)$

l:

The case which matches with the function is: $f(n) = O(g(n))$ and the function can be written as $n^{1/2} = O(5^{\log_2 n})$

m:

The case which matches with the function is: $f(n) = O(g(n))$ and the function can be written as $n2^n = O(3^n)$

n:

The case which matches with the function is: $f(n) = \Theta(g(n))$ and the function can be written as $2^n = \Theta(2^{n+1})$

o:

The case which matches with the function is: $f(n) = \Omega(g(n))$ and the function can be written as $n! = \Omega((2)^n)$

p:

The case which matches with the function is: $f(n) = O(g(n))$ and the function can be written as $(\log n)^{\log n} = O\left(2(\log_2 n)^2\right)$

q:

The case which matches with the function is: $f(n) = \Theta(g(n))$ and the function can be written as $\sum_{i=1}^{n} i^k = \Theta\left(n^{k+1}\right)$

Exercise 0.2:

(a) $\Theta(1)$ **if** $c < 1$.

(b) $\Theta(n)$ **if** $c = 1$.

(c) $\Theta(c^n)$ **if** $c > 1$.

The formula for the sum of a partial geometric series is simplified as follows:

$$g(n) = 1 + c + c^2 + \ldots + c^n = \frac{c^{n+1} - 1}{c - 1} \quad \ldots\ldots (1)$$

(a)

$\Theta(1)$ if $c < 1$

If $c < 1$, using the formula for the sum of a partial geometric series, equation (1) can be written as follows:

$$\lim_{n \to \infty} g(n) = \frac{0 - 1}{c - 1}$$
$$= \frac{1}{1 - c}$$

Since the value of $\lim_{n \to \infty} c^{n+1} = 0$.

$$\lim_{n \to \infty} g(n) = \frac{1}{1 - c}$$

$$\frac{1}{1 - c} > g(n) > 1$$

So, it can be concluded that if the value of $c < 1$, the value of the terms is decreasing. Hence, the big-O notation for the above term is $\Theta(1)$.

4 Solve T(n) = 2 T(n-2) + 2, with n = 2k and for T(0) = 0, and T(0)= 1

T(n) = 2T(n - 2) + 2, n = 2k
= 2(2T(n - 4) + 2) + 2 = 4T(n - 4) + 2*2 + 2
= 4(2T(n - 6) + 2) + 2*2 + 2 = 2^3T(n - 2^3) + 2^3) + 2^2 + 2

para T(0) = 1
= 2^kT(0) + $2^{(k-1)}$ + $2^{(k-2)}$ + ... + 2^2 + 2
= $2^k - 1 - 1$ = $2^k - 2$ = $2^{(n/2)}$ - 2
para T(0) = 0
= 2^kT(0) + $2^{(k-1)}$ + $2^{(k-2)}$ + ... + 2^2 + 2
= $2^k - 1$ = $2^{(n/2)}$ - 1