Time complexity - Lab 4

Fabio Steven Tovar Ramos

October 15th, 2018

1 Table

Algorithm	Function	Time complexity			Space complexity
		Worst case	Best case	Average case	
Simplest primality by trial division	$f(n) = \sqrt{n}$	$O(\sqrt{n})$	$\Omega(\sqrt{n})$	$\Theta(\sqrt{n})$	$O(1)$
Binary Search	$f(n) = \lg_2 n$	$O(\lg_2 n)$	$\Omega(1)$	$\Theta(\lg_2 n)$	$O(1)$
Finding the smallest or largest item in an unsorted array	$f(n) = n \lg_2 n$	$O(n)$	$\Omega(n)$	$\Theta(n \lg_2 n)$	$O(1)$
Kadane's algorithm	$f(n) = n$	$O(n)$	$\Omega(n)$	$\Theta(n)$	$O(1)$
Sieve of Eratosthenes	$f(n) = n$	$O(n \lg(\lg n))$	$\Omega()$	$\Theta(n)$	$O(\sqrt{n})$
Merge Sort	$f(n) = n \lg n$	$O(n \lg n)$	$\Omega(n \lg n)$	$\Theta(n \lg n)$	$O(n)$
Heap Sort	$f(n) = n \lg n$	$O(n \lg n)$	$\Omega(n \lg n)$	$\Theta(n \lg n)$	$O(1)$
Quick Sort	$f(n) = n \lg n$	$O(n^2)$	$\Omega(n \lg n)$	$\Theta(n \lg n)$	$O(n)$
Tim Sort	$f(n) = n \lg n$	$O(n \lg n)$	$\Omega(n)$	$\Theta(n \lg n)$	$O(n)$
Divide and conquer (Convex Hull)	$f(n) = n \lg n$	$O(n^2)$	$\Omega(n \lg n)$	$\Theta(n \lg n)$	$O(\lg n)$
Insertion Sort	$f(n) = n^2$	$O(n^2)$	$\Omega(n)$	$\Theta(n^2)$	$O(1)$
Dijkstra's algorithm	$f(n) = \lvert E\rvert + \lvert V\rvert \lg \lvert V\rvert$	$O(\lvert E\rvert + \lvert V\rvert \lg \lvert V\rvert)$	$\Omega(\lvert E\rvert + \lvert V\rvert \lg \lvert V\rvert)$	$\Theta(\lvert E\rvert + \lvert V\rvert \lg \lvert V\rvert)$	$O(V^2)$
Naive Matrix Multiplication	$f(n) = n^3$	$O(n^3)$	$\Omega(n^3)$	$\Theta(n^3)$	$O(n)$
Floyd–Warshall algorithm	$f(n) = n^3$	$O(n^3)$	$\Omega(n^3)$	$\Theta(n^3)$	$O(n^2)$
Naive Matrix Inversion	$f(n) = n^3$	$O(n^3)$	$\Omega(n^3)$	$\Theta(n^3)$	$O(n)$
Calculate the permutations of n distinct elements without repetitions	$f(n) = n!$	$O(n!)$	$\Omega(n!)$	$\Theta(n!)$	$O(n!)$
Calculate the permutations of n distinct elements with repetitions	$f(n) = n * n!$	$O(n * n!)$	$\Omega(n * n!)$	$\Theta(n * n!)$	$O(n * n!)$

2 Cormen Exercises

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 $8n^2$ steps, while merge sort runs in $64n \lg n$ steps. For which values of n does insertion sort beat merge sort?

Insertion sort have less steps than merge sort for $1 < n \leq 43$, as is shown in this the Figure 1. For other values Merge sort is better, i.e for $n > 43$.

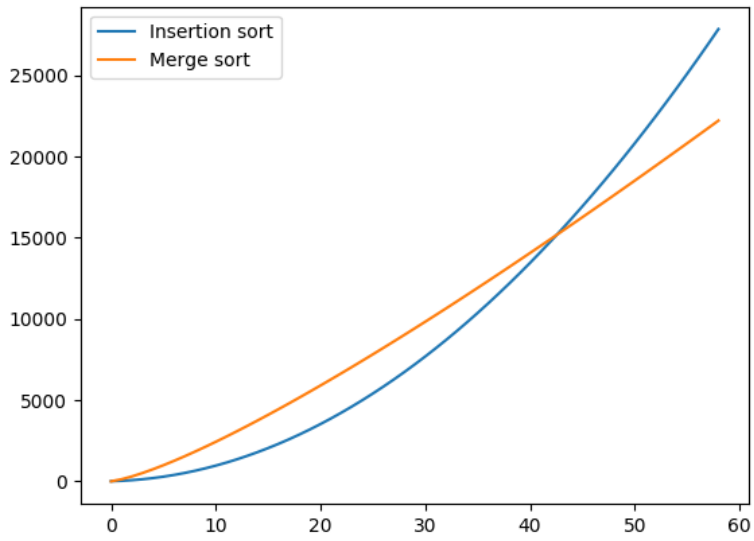

Figure 1: Comparison of Insertion sort and Merge sort implementations

1.2-3 What is the smallest value of n such that an algorithm whose running time is $100n^2$ runs faster than an algorithm whose running time is 2^n on the same machine?

The algorithm 2 runs faster for $0 < n \leq 14$, so that the smallest value of n where algorithm 1 runs faster than algorithm 2 is $n = 15$

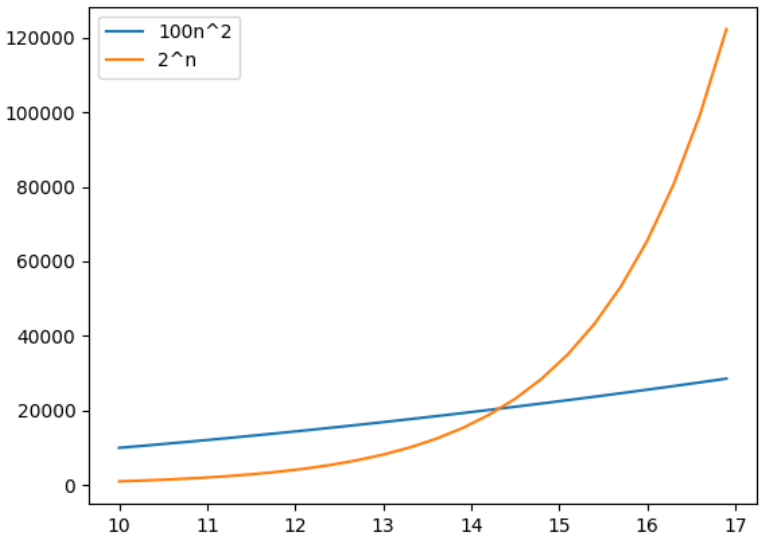

Figure 2: Comparison between both time complexities

Problem 1-1 Comparison of running times for each function $f(n)$ 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 $f(n)$ microseconds.

1 microsecond (10^{-6} seg)

	1 second	**1 minute**	**1 hour**	**1 day**	**1 month**	**1 year**	**1 century**
$\lg n$	2^{10^6}	$2^{10^6} * 60$	$2^{10^6} * 360$	$2^{10^6} * 8640$	$2^{10^6} * 259200$	$2^{10^6} * 3110400$	$2^{10^6} * 31104 * 10^4$
$sqrtn$	10^{12}	$6 * 10^{13}$	$3.6 * 10^{15}$	$8.64 * 10^{16}$	$2.592 * 10^{18}$	$3.1104 * 10^{19}$	$3.1104 * 10^{21}$
n	10^6	$6 * 10^7$	$36 * 10^8$	$8.6 * 10^{10}$	$2.592 * 10^{12}$	$3.1104 * 10^{13}$	$3.1104 * 10^{15}$
$n \lg n$	1201	72060	4323600	103766400	$3112992 * 10^3$	$3.7355904 * 10^{10}$	$3.7355904 * 10^{12}$
n^2	1000	$6 * 10^4$	$36 * 10^5$	$864 * 10^5$	$2592 * 10^6$	$3.1104 * 10^{10}$	$3.1104 * 10^{12}$
n^3	100	$6 * 10^3$	$36 * 10^4$	$864 * 10^4$	$2592 * 10^5$	$3.1104 * 10^9$	$3.1104 * 10^{11}$
2^n	19	1140	68400	1641600	49248000	590976000	$590976 * 10^5$
$n!$	9	540	32400	777600	23328000	279936000	$279936 * 10^5$

Table 1: Maximum operations for a computer that makes a computational step in one microsecond

1 nanosecond (10^{-9} seg)

	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
$\lg n$	2^{10^9}	$2^{10^9} * 60$	$2^{10^9} * 360$	$2^{10^9} * 8640$	$2^{10^9} * 259200$	$2^{10^9} * 3110400$	$2^{10^9} * 31104 * 10^4$
$sqrtn$	10^{18}	$6 * 10^{19}$	$3.6 * 10^{21}$	$8.64 * 10^{22}$	$2.592 * 10^{24}$	$3.1104 * 10^{25}$	$3.1104 * 10^{27}$
n	10^9	$6 * 10^10$	$36 * 10^11$	$8.6 * 10^{13}$	$2.592 * 10^{15}$	$3.1104 * 10^{15}$	$3.1104 * 10^{17}$
$n \lg n$	3.96210^7	2.37710^9	1.42610^{11}	$3.4231 * 10^{12}$	1.0269510^{14}	1.2323410^{15}	1.2323410^{17}
n^2	31623	1897380	$1.138428 * 10^8$	$2.7322272 * 10^8$	$8.1966 * 10^{10}$	$9.836 * 10^{11}$	$9.836 * 10^{13}$
n^3	1000	$6 * 10^4$	$36 * 10^5$	$864 * 10^6$	$2592 * 10^6$	$3.1104 * 10^{10}$	$3.1104 * 10^{12}$
2^n	29	1740	104400	$2.505 * 10^6$	$7.516 * 10^7$	$9.0201 * 10^8$	$9.0201 * 10^{10}$
$n!$	12	720	$4.32 * 10^4$	$1.036 * 10^6$	$3.11 * 10^7$	$3.732 * 10^8$	$3.732 * 10^{10}$

Table 2: Maximum operations for a computer that makes a computational step in one nanosecond

3-1 Asymptotic behavior of polynomials

Let

$$p(n) = \sum_{i=0}^{d} a_i n^i \tag{1}$$

where $a_d > 0$, be a degree-d polynomial in n, and let k be a constant. Use the definitions of the asymptotic notations to prove the following properties.

a. If $k \geq d$, then $p(n) = O(n^k)$

O-notation definition: $f(n) = O(g(n))$ if exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$

We start building the following polynomial.

$$q(n) = \sum_{i=0}^{d} a_i n^i + \sum_{j=d+1}^{k-d} a_j n^j = p(n) + \sum_{j=d+1}^{k-d} a_j n^j \tag{2}$$

Note that $q(n)$ have the same coefficients that $p(n)$ but have one of more extra terms. So that for some n_0, $cg(n) \leq f(n)$ for all $n \geq n_0$ and some c.

By the Θ-notation definition, and showing that k is the dominant coefficient, we can affirm that $\Theta(q(n)) = \Theta(n^k)$ and $q(n) = \Theta(n^k)$ implies that $q(n) = O(n^k)$.

Due that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$ and some c, we conclude that $f(n) = O(g(n)) = O(n^k)$.

b. If $k \leq d$, then $p(n) = \Omega(n^k)$

Ω-notation definition: $f(n) = \Omega(g(n))$ if exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$

We start building the following polynomial.

$$q(n) = p(n) - \sum_{j=k+1}^{d} a_j n^j = \sum_{i=0}^{k} a_i n^i \tag{3}$$

Since $a_d > 0$, we can affirm that $cg(n) \leq f(n)$ for all $n \geq n_0$ and some c.

Further, we can affirm that $\Theta(q(n)) = \Theta(n^k)$ and $q(n) = \Theta(n^k)$ implies that $q(n) = \Omega(n^k)$ due the Θ-notation definition.

Due that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$ and some c, we conclude that $f(n) = \Omega(g(n)) = \Omega(n^k)$.

c. If $k = d$, then $p(n) = \Theta(n^k)$

Θ-notation definition: $f(n) = \Theta(g(n))$ if there positive constants c_1, c_2 and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$

For $p(n)$ we have that $p(n) = \Theta(n^d)$ because d is the dominant term in $p(n)$. Now, as $k = d$, we can affirm that $\Theta(n^d) = \Theta(n^k)$, so that $p(n) = \Theta(n^k)$

d. If $k > d$, then $p(n) = o(n^k)$

o-notation definition: $o(g(n)) = f(n)$ if for any positive constant $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq f(n) < cg(n)$ for all $n \geq n_0$

The prove are very similar to the a. point, only we have to make a small modification. As $k > d$, will be one or more new terms in the $q(n)$ polynomial, so that $cq(n) > p(n)$ for all c and for n such that $n > n_0$.

e. If $k < d$, then $p(n) = \omega(n^k)$

ω-notation definition: $\omega(g(n)) = f(n)$ if for any positive constant $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq cg(n) < f(n)$ for all $n \geq n_0$

The prove are very similar to b. point too. As $k < d$, will be one or more terms eliminated of $p(n)$ in the building of $g(n)$, so that $cq(n) > p(n)$ for all c and for n such that $n > n_0$.

3 Dasgupta, Papadimitriou and Vazirani Exercises

0.1. In each of the following situations, indicate whether $f = O(g)$, or $f = \Omega(g)$, or both (in which case $f = \Theta(g)$).

	$f(n)$	$g(n)$	
(a)	$n-100$	$n-200$	$f=\Theta(g)$
(b)	$n^{\frac{1}{2}}$	$n^{\frac{2}{3}}$	$f=O(g)$
(c)	$100n+\lg(n)$	$n+(\lg(n))^2$	$f=O(g)$
(d)	n	$10n\lg 10n$	$f=\Theta(g)$
(e)	$\lg 2n$	$\lg 3n$	$f=O(g)$
(f)	$10\lg n$	$\lg(n^2)$	$f=\Omega(g)$
(g)	$n^{1.01}$	$n\lg^2 n$	$f=O(g)$
(h)	$\frac{n^2}{\lg n}$	$n(\lg n)^2$	$f=\Omega(g)$
(i)	$n^{0.1}$	$n\lg^2 n$	$f=O(g)$
(j)	$(\lg n)^{\lg n}$	$\frac{n}{\lg n}$	$f=O(g)$
(k)	$\sqrt{n}$	$(\lg n)^3$	$f=\Omega(g)$
(l)	$n^{\frac{1}{2}}$	$5^{\lg_2 n}$	$f=O(g)$
(m)	$n2^n$	3^n	$f=O(g)$
(n)	2^n	2^{n+1}	$f=O(g)$
(o)	$n!$	2^n	$f=\Omega(g)$
(p)	$(\lg n)^{\lg n}$	$2^{(\lg_2 n)^2}$	$f=O(g)$
(q)	$\sum_{i=1}^{n} i^k$	n^{k+1}	$f=\Theta(g)$

Table 3: Solution 0.1

0.2. Show that, if c is a positive real number, then $g(n)=1+c+c^2+...+c^n$ is:

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

For $g(n)$ terms, as $c<1$ we have that $c_i<c_{i+1}$ for all $i\in[0,n)$, i.e. the geometric serie is strictly decreasing, so that we can affirm $g(c)=\Theta(c_0)=\Theta(1)$.

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

For $g(n)$ terms, we have that $c_i=c_{i+1}$ for all $i\in[0,n)$ due the $c=1$ hypothesis. So we can affirm that the big-Θ term is the sum of the terms, so that $g(n)=\Theta(n+1)=\Theta(n)$

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

For $g(n)$ terms, we have that $c_i>c_{i+1}$ for all $i\in[0,n)$ so that we can affirm that $g(n)=\Theta(c_n)=\Theta(c^n)$

The moral: in big-Θ terms, the sum of a geometric series is simply the first term if the series is strictly decreasing, the last term if the series is strictly increasing, or the number of terms if the series is unchanging.

3.1 Last exercise

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

3.1.1 Recursive substitution

$$
\begin{aligned}
T(n) &= 2T(n-2)+2 \\
&= 2[2T(n-4)+2]+2 = 2\times 2T(n-4)+2\times 2+2 \\
&= 2[2\times 2T(n-6)+2\times 2+2]+2 = 2^3T(n-6)+2^3+2^2+2 \\
&= 2[2^3T(n-8)+2^3+2^2+2]+2 = 2^4T(n-8)+2^4+2^3+2^2+2 \\
&\vdots \\
&= 2^kT(n-2k)+\sum_{i=1}^{k}2^i \qquad (4)
\end{aligned}
$$

$$
\begin{aligned}
T(n-2k) &= T(0) \\
n-2k &= 0 \\
2k &= n \\
k &= \frac{n}{2}
\end{aligned}
$$

So that, replacing k in equation 4:
For $T(0)=0$:

$$
\begin{aligned}
2^{(n/2)}T(0)+\sum_{i=1}^{n/2}2^i &= 2^{(n/2)}\times 0+\sum_{i=1}^{n/2}2^i \\
&= 2^{(n/2)+1}-1 \\
&= 2^{(3n/2)}-1 \\
&= O(2^{3n/2}) = O((\sqrt{8})^n)
\end{aligned}
$$

For $T(0)=1$:
So that, replacing k in equation 4:

$$
\begin{aligned}
2^{(n/2)}T(0)+\sum_{i=1}^{n/2}2^i &= 2^{(n/2)}\times 1+\sum_{i=1}^{n/2}2^i \\
&= 2^{(n/2)}+2^{(n/2)+1}-1 \\
&= 2^{(3n/2)}+2^{(n/2)}-1 \\
&= O(2^{3n/2}) = O((\sqrt{8})^n)
\end{aligned}
$$

3.1.2 Graphical method

asdsad