CS340 ⇒ Assignment 1

Maksim Sharoika
Due: Sept. 22/2022

Problem 1) a) $5.3N^{7/2} + 22N^3 \log(N) + 100N^{3/2}$

$N^3 N^{1/2} : N^3 \log(N)$ → $\log(N) \in o(N^{1/2})$

$N^{1/2} : \log(N)$

$\therefore T(N) \in \Theta(N^{7/2})$

b) $2.4\frac{\log^2(N)}{N} + 12.5N$ · $\log^k(N) \in o(N)$

$\therefore T(N) \in \Theta(N)$ ↳ $\frac{\log^k}{N} \in o(N)$

c) $0.11N \cdot \log^2(N) + \frac{5N}{\log(N)}$ $N \cdot \log^4(N) > CN$

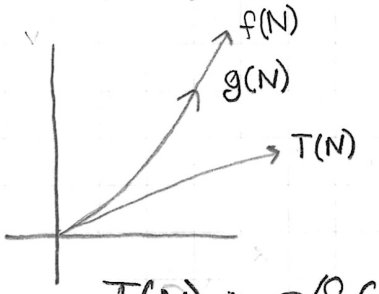
$\log^4(N) > C$

$\therefore T(N) \in \Theta N\log^3(N)$

d) $36\log^2(N)/N^2 + 3^N \cdot 3\log(N)/N^2 + 57\log(N)/N^2$

$\therefore T(N) \in \Theta((3^N \log(N))/N^2)$

Problem 2) a) given: $T(N) = of(N)$ ∴ $f(N) = \Theta(g(N))$ prove $T(N) = o(g(N))$



$f(N) = \Theta(g(N))$ implies $f(N) = O(g(N))$ ∴ $f(N) = \Omega(g(N))$ (B, C)

$T(N) = of(N)$ implies $T(N) = O(f(N))$ ∴ $T(N) \neq \Omega(f(N))$ (A)

A(B) ⇒ $T(N) = O(g(N))$ ∴ Θ means same growth rate or tight bound.

$T(N) \neq \Omega(f(N))$ ∴ $f(N) = O(g(N))$ implies $T(N) \neq \Omega(g(N))$

↳ this therefore implies that $\boxed{T(N) = o(g(N))}$

b) given: $T(N) = O(g(N))$ ∴ $f(N) = o(g(N))$ ∴ $T(N) \neq \Omega(f(N))$

provide a function(s) T: provide the function $T(N)$, $f(N)$, and $g(N)$

$T(N) = C$ i.e (5)

$g(N) = N$

$f(N) = \log(N)$

Always little o of differer larger class

$T(N) = O(g(N))$

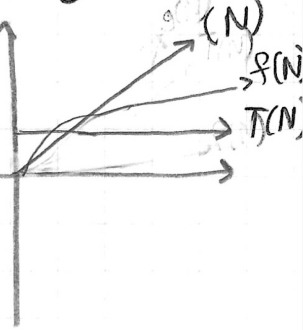
$5 = O(N)$ ← Correct!

$f(N) = o(g(N))$ ← Correct!

$\log(N) = o(N)$

$T(N) \neq \Omega(f(N))$ ← Correct.

$5 \neq \Omega\log(N)$



Problem 3)

a) `for (i=n; i>0; i--)`
`  for (j=0; j<n; j=j+2)`
`  Cout << "It's not winter yet." << endl;` | O(1) | $O(n/2)$ | $O(n)$

$\therefore n \cdot n/2 \cdot 1 = n^2/2$ so the algorithm is $\boxed{O(n^2)}$

b) `for(i=1; i<n; i=2*i)`
`  for (j=0; j<floor(n/2); j++)`
`  Cout << "Text" << endl;` | O(1) | $O(n/2)$ | $O(\log(N))$

$\therefore \log(n) \cdot 1 \cdot n/2 = \frac{\log(n) \cdot n}{2}$ so the algorithm is $\boxed{O(n \cdot \log(n))}$

c) `int myFunction(int n)` } T(N)
`{`
`  if (n<=1)`
`  return 1;`
`  else`
`  return myFunction(n-1)*n;`
`}`

$T(1) = 1$ } Base Case
$T(N) = T(N-1) \cdot N$

N=1 -> 1 = 1
N=2 -> 1 · 2 = 2 (×3)
N=3 -> (1 · 2) · 3 = 6 (×4)
N=4 -> (6) · 4 = 24 (×5)
N=5 -> (24) · 5 = 120

Proof by induction

induction base = [k=1] $T(k!)$ for some fixed k.

$T(K!) = T(1!) = 1 = (1) \cdot (0!) = K \cdot (k-1)! = k!$

Assume that $T(k!) = K \cdot (k-1)! = K!$

now $k \rightsquigarrow k+1$

$T((k+1)!) = (k+1) \cdot (k)!$

$T((k+1)!) = (k+1)\, T(k!)$

[This proves the conjecture.

L> Therefore the function is $T(N) = T(N-1) \cdot N$ which will have as many operations / loops as from N to 0.

L> This gives this algorithm $\boxed{O(N)}$

Problem 4)

a) Program will be attached to submission.

b)

n	10	100	1000	10000	100000	1000000	10000000	20000000	50000000	100000000	300000000
Assignment1Alg(n)	30	600	9000	130000	$1.6 \cdot 10^6$	$1.9 \cdot 10^7$	$2.3 \cdot 10^8$	$4.8 \cdot 10^8$	$1.25 \cdot 10^9$	$2.6 \cdot 10^9$	$8.4 \cdot 10^9$
n / Result	0.33	0.166	0.11	0.08	0.06	0.05	0.04	0.04	0.04	0.04	0.04
$n \cdot \log_2(n)$/result	1.11	1.11	1.11	1.02	1.038	1.037	1.011	1.01	1.02	1.02	1.04
$n^{1.5}$	1.05	1.67	3.51	7.69	19.76	52.63	137	187	282	384	618
timeUsed	$2 \cdot 10^{-6}$	$3 \cdot 10^{-6}$	$1.9 \cdot 10^{-5}$	$3.5 \cdot 10^{-4}$	$4.4 \cdot 10^{-3}$	0.0522	0.66	1.4	3.83	8.12	27.75
n/timeUsed	$5 \cdot 10^6$	$3.3 \cdot 10^7$	$5.2 \cdot 10^7$	$2.83 \cdot 10^7$	$2.23 \cdot 10^7$	$1.9 \cdot 10^7$	$1.5 \cdot 10^7$	$1.4 \cdot 10^7$	$1.3 \cdot 10^7$	$1.2 \cdot 10^7$	$1.08 \cdot 10^7$
$n \cdot \log_2(n)/_{timeUsed}$	$1.66 \cdot 10^7$	$2.21 \cdot 10^8$	$5.2 \cdot 10^8$	$3.76 \cdot 10^8$	$3.71 \cdot 10^8$	$3.8 \cdot 10^8$	$3.5 \cdot 10^8$	$3.44 \cdot 10^8$	$3.33 \cdot 10^8$	$3.3 \cdot 10^8$	$3.04 \cdot 10^8$
$n^{1.5}/_{timeUsed}$	$1.58 \cdot 10^7$	$3.3 \cdot 10^8$	$1.66 \cdot 10^9$	$2.83 \cdot 10^9$	$7.06 \cdot 10^9$	$1.91 \cdot 10^{10}$	$4.74 \cdot 10^{10}$	$6.35 \cdot 10^{10}$	$9.22 \cdot 10^{10}$	$1.23 \cdot 10^{11}$	$1.8 \cdot 10^{11}$

c) $f_1(n) = n \quad f_2(n) = n \cdot \log(n) \quad f_3(n) = n\sqrt{n}$

i) The growth rate of result in relation to n in assignment1Algorithm is $\Theta(n \cdot \log(n))$ because it is the only one that converges / shows constant value.

$$\boxed{\Theta(n \cdot \log(n))}$$

The growth rate of timeUsed in relation to n in assignment1Algorithm is $O(\log(n) \cdot n)$ by similar logic it converges to a constant value.

$$\boxed{\Theta(n \cdot \log(n))}$$

$n\sqrt{n}$ grows alot faster than $n \cdot \log(n)$ which grows a little bit faster than n.

relation of $f_1(n), f_2(n), f_3(n)$

$n = O(n \cdot \log(n)) \quad n \cdot \log(n) = O(n\sqrt{n})$