Assignment 1

Zu Yu Zhang

Problem 1

- $f1(n) = 10^n$ -> exponential
- $f2(n) = n^{\frac{1}{3}}$ -> polynomial
- $f3(n) = n^n$ -> exponential
- $f4(n) = \log_2 n$ -> logarithmic
- $f5(n) = n\log_2 n$ -> poly-log

  f4 < f2 < f5 < f1 < f3

- logarithmic functions grow slower than polynomial functions, therefore f4 comes first, followed by f2 and f5.
- f1 and f3 are both exponential functions, let's compute the limit:

  $$\lim_{n\to\infty} \frac{10^n}{n^n} = \lim_{n\to\infty} \left(\frac{10}{n}\right)^n = n \lim_{n\to\infty} \left(\frac{10}{n}\right) = n * 0 = 0$$

  Therefore, $10^n = O(n^n)$

Problem 2

a. $f(n) = n - 100 \quad g(n) = n - 200;$

   f = $\Theta(g)$

- Constants can be ignored in asymptotic analysis; therefore, both f and g run in linear time.

b. $f(n) = n^{\frac{1}{2}} \quad g(n) = n^{\frac{2}{3}}$

   $f = O(g)$

- $\frac{2}{3} > \frac{1}{2};$ g grows faster than f.

c. $f(n) = 100n + log(n) \quad g(n) = n + (logn)^2$

   f = $\Theta(g)$

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 100$
- Since the limit converged to a constant (not zero), f and g grow at the same rate.

d. $f(n) = nlog(n) \quad g(n) = 10nlog(10n)$

   f = $\Theta(g)$

- Multiplicative constants can be ignored

e. $f(n) = \ log2\,(n) \quad g(n) = \ log3\,(n)$

f = $\Theta\,(g)$

- Multiplicative constants can be ignored.

f. $f(n) = \ 10logn \quad g(n) = log\,(n^2)$

f = $\Theta\,(g)$

- According to logarithm product rule:
  - $log\,(n)^2 = log\,(n\ *\ n) = log\,(n) + log\,(n) = 2log\,(n);$
- Multiplicative constants can be ignored; f and g grow at the same rate.

g. $f(n) = \ n^{1.01} \quad g(n) = \ nlog^2 n$

f = Ω(g)

- $f(n) = \ n\ *\ n^{0.01}$
- $g(n) = \ n\ *\ (logn)\ * (logn)$
- Polynomials grows faster than logarithms.

h. $f(n) = \dfrac{n^2}{logn} \quad g(n) = \ n(logn)^2$

f = $\Omega(g)$

- $\lim_{n \to \infty} \dfrac{n}{(logn)^3}$
- polynominals grow faster than logarithmic functions, therefore this limit evaluates to ∞ and f grows faster than g.

i. $f(n) = \ n^{0.1} \quad g(n) = \ (logn)^{10}$

f = $\Omega(g)$

- polynomials grow faster than logarithms.

j. $f(n) = \ (logn)^{logn} \quad g(n) = \dfrac{n}{logn}$

f = $\Omega\,(g)$

- $b^{logb(x)} = x,\ therefore\ b^{log(logn)} = logn$
- f(n) = $\left(b^{log(logn)}\right)^{logn} = \left(b^{logn}\right)^{log(logn)} = n^{log(logn)}$
- $\lim_{n \to \infty} \dfrac{n^{log(logn)}}{\frac{n}{logn}} = \lim_{n \to \infty} n^{log(logn)\,-1}$ * logn

$$= \lim_{n\to\infty} n^{log(log\,n)\,-1} + \lim_{n\to\infty} log\,n$$ by product rule of limit laws

$$= \infty + \infty$$

$= \infty$ -> f grows faster than g

k. $f(n) = \sqrt{n} \quad g(n) = (log\,n)^3$

f = $\Omega(g)$

- polynomials grow faster than logarithms.

l. $f(n) = n^{\frac{1}{2}} \; g(n) = 5^{log_2 N}$

f = O(g)

- exponentials grows faster than polynomials.

m. $f(n) = n2^n \;\; g(n) = 3^n$

f(n) = O(g)

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

n. $f(n) = 2^n, \; g(n) = 2^{n+1}$

f = $\Theta(g)$

- $g(n) = 2^n * 2^1 \rightarrow g(n) = 2f(n)$
- f and g grow at the same rate since multiplicative constants are not considered.

Problem 3

a. $T(n) = 2T\left(\frac{n}{3}\right) + 1$

- a = 2, b = 3, c = 0
- $log_3 2 > 0$, T(n) = $O\left(n^{log3(2)}\right)$

b. $T(n) = 5T\left(\frac{n}{4}\right) + n$

- a = 5, b = 4, c = 1
- $log_4 5 > 1$, T(n) = $O\left(n^{log4(5)}\right)$

c. $T(n) = 7T\left(\frac{n}{7}\right) + n$

- a = 7, b = 7, c = 1
- $log_7 7 = 1$, $T(n) = O(nlogn)$

d. $T(n) = 9T\left(\frac{n}{3}\right) + n^2$

- a = 9, b = 3, c = 2
- $\log_3 9$ = 2, $T(n) = O(n^2 logn)$

e. $T(n) = 8T\left(\frac{n}{2}\right) + n^3$

- a = 8, b = 2, c = 3
- $\log_2 8$ = 3, T(n) = O(n^3 (logn))

g. $T(n) = T(n-1) + 2$

T(n)
/ \
2 T(n-1)
/ \
2 T(n-2)
/ \
2 T(n-3)
...
/ \
2 T(1)
\
T(0)

- There is a total of n levels, on each level the work done is contant (2 units), the total work done is 2n -> T(n) = O(n)

h. $T(n) = T(n-1) + n^c$, where $c \geq 1$ is a constant

T(n)
/ \
n^c T(n-1)
/ \
$(n-1)^c$ T(n-2)
/ \
$(n-2)^c$ T(n-3)
....
1 T(1)
\
T(0)

$n^c + (n-1)^c + (n-2)^c + \ ... \ +1$

- According to the Faulhaber's formula, this sum will be a polynomial function with degree (c+1), therefore T(n) = $O(n^{c+1})$

i. $T(n) = T(n-1) + c^n$, where $c > 1$ is some constant

T(n)
/ \
c^n T(n-1)
/ \
c^{n-1} T(n-2)
/ \
c^{n-2} T(n-3)

```
                                   …
                               1  T(1)
                                     \
                                     T(0)
```

$c^n + c^{n-1} + c^{n-2} + \ ... \ + c + 1$

- Geometric sequence where the ratio is equal to c -> $T(n) = O\left(c^{n+1}\right)$.

j) $T\ (n) = 2T(n-1) + 1$

```
                T(n)
             /          \      work done: 1
         T(n-1)        T(n-1)
          /  \           /   \  work done: 2
    T(n-2)  T(n-2)  T(n-2)  T(n-2)
       … … … …                      work done: 4
 T(1)   T(1)                   T(1)      …
```

- sum of work done:
  - $2^0 + 2^1 + 2^2 + \ ... \ + 2^n = 2^{n+1} - 1$
- Geometric sequence where the ratio is 2, sum will be exponential to 2, therefore $T(n) = O(2^n)$.

References

2.1.1 Recurrence Relation (T(n)= T(n-1) + 1) #1
https://en.wikipedia.org/wiki/List_of_logarithmic_identities
https://math.stackexchange.com/questions/2605764/lognlogn-nlog10logn-why
https://www.ck12.org/book/cbse_maths_book_class_11/section/15.10/