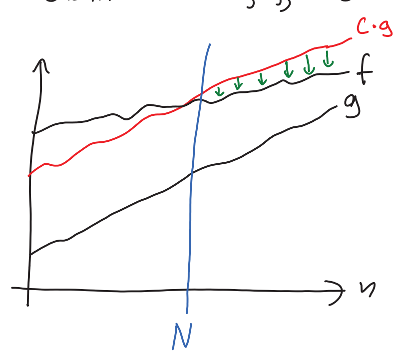
1. IF $f, g : \mathbb{N} \to [0, \infty)$:

(a) $f(n) = O(g(n))$ MEANS $\exists\, C > 0, N$ SUCH THAT $\forall n \geq N$, $\boxed{|f(n)| \leq C \cdot g(n)}$

(I.E., $f(n)$ IS EVENTUALLY NO LARGER THAN SOME CONSTANT MULTIPLE OF $g(n)$.)

(i) THERE MUST BE SOME CONSTANT MULTIPLE OF g $(\exists\, C > 0)$
THAT $f(n)$ IS NO LARGER THAN $(|f(n)| \leq C \cdot g(n))$
WHEN n IS SUFFICIENTLY LARGE $(\exists\, N, \forall n \geq N)$

(ii) IF WE PLOT THE GRAPHS OF f, g, AND $C \cdot g$:



(b) THE POINT OF BIG-O NOTATION IS THAT IT ALLOWS US TO THINK ABOUT THE OVERALL GROWTH RATE OF A FUNCTION WITHOUT WORRYING ABOUT LITTLE DETAILS (SEE PROBLEM #4!).

(c) (i) NOTHING HERE EQUALS ANYTHING (THE = IS USED AS A DESCRIPTION, OR BETTER YET, A $\in$).

(ii) WHAT THIS IS REALLY ABOUT IS THE WHOLE FUNCTION, NOT JUST AN INDIVIDUAL VALUE $f(n)$ VS. $g(n)$.

(ALL IN ALL, THIS SHOULD BE WRITTEN $f \in O(g)$ TO BETTER REFLECT THE ACTUAL CONCEPT)

(d) $f_1(n), f_2(n) = O(g(n)) \Rightarrow f_1(n) \pm f_2(n) = O(g(n))$

(SIMILARLY FOR CONSTANT MULTIPLES: $f(n) = O(g(n)) \Rightarrow A \cdot f(n) \in O(g(n))$)

(e) $f_1(n) = O(g_1(n)) \wedge f_2(n) = O(g_2(n)) \Rightarrow f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$

BOTH BOUNDS ARE USED!

2. (a) $f(n)=\Omega(g(n))$ MEANS $\exists\, C>0, N$ SUCH THAT $\forall n \geq N$, $f(n) \geq C \cdot g(n)$

THE OPPOSITE INEQUALITY!

f IS EVENTUALLY NO SMALLER THAN SOME POSITIVE MULTIPLE OF g

(b) $f(n)=\Theta(g(n))$ MEANS BOTH $f(n)=O(g(n)) \wedge f(n)=\Omega(g(n))$

I.E., f IS EVENTUALLY BETWEEN TWO POSITIVE MULTIPLES OF g, SO f & g ARE ROUGHLY COMPARABLE IN SIZE.

3. WHEN n IS LARGE, A FEW BASIC CATEGORIES GIVE US A STARTING POINT FOR UNDERSTANDING O, Ω, AND Θ:

					(POSITIVE POWERS OF n)	(EXPONENTIALS, BASE >1)		
0 (ZERO)	CONSTANTS	$\log(\log n)$	$\log n$	$\log^k n$	$\ldots n^{1/10} \ldots \sqrt{n} \ldots n \ldots n^2 \ldots n^{10} \ldots$	$\ldots 1.01^n \ldots 2^n \ldots 10^n \ldots 100^n \ldots$	$n!$	n^n
$O(0) \subset$	$O(1) \subset$	$O(\log\log n) \subset$	$O(\log n) \subset$	POLYLOG $\subset$	$O(n^a) \subset$	$O(a^n) \subset$	$O(n!) \subset$	$O(n^n)$

- ANYTHING FARTHER LEFT IS O(ANYTHING FARTHER RIGHT)
- ANYTHING FARTHER RIGHT IS Ω(ANYTHING FARTHER LEFT)
- OTHER THAN NONZERO CONSTANTS, NONE OF THESE ARE Θ(ANY OTHER)

⊛ DUE TO OUR $\pm$ RULE FOR BIG-O, "SMALLER" SUMMANDS (& CONSTANT FACTORS) ARE SIMPLY ABSORBED INTO LARGER ONES!

⊛ DON'T FORGET THAT OUR $\cdot$ RULE MEANS THAT NON-CONSTANT FACTORS AREN'T ABSORBED!

4. (a) $\underbrace{5n^3}_{O(n^3)} + \underbrace{3n}_{O(n^3)} + \underbrace{1000}_{O(n^3)} = O(n^3)$

(b) $\underbrace{6\log(n^3)}_{O(n)} + \underbrace{2n}_{O(n)} + \underbrace{5}_{O(n)} = O(n)$

NOTE THAT $\log(n^3) = 3 \cdot \log n$

(c) $\underbrace{8n^{100}}_{O(2^n)} + \underbrace{2^n}_{O(2^n)} + \underbrace{\log n}_{O(2^n)} = O(2^n)$

(d) $\underbrace{100^n}_{O(n!)} + \underbrace{4n^{50}}_{O(n!)} + \underbrace{n!}_{O(n!)} = O(n!)$

(e) $(\underbrace{100n}_{O(n^4)} + \underbrace{n^4}_{O(n^4)})(\underbrace{n^2}_{O(2^n)} + \underbrace{2^n}_{O(2^n)}) = O(n^4 \cdot 2^n)$

(f) $(\underbrace{\log\log n}_{O(\log\log n)} + \underbrace{10000}_{O(\log\log n)})(\underbrace{\log n}_{O(n)} + \underbrace{n}_{O(n)} + \underbrace{\sqrt{n}}_{O(n)}) = O(n \cdot \log\log n)$

(g) $(\underbrace{n!}_{O(n^n)} + \underbrace{n^n}_{O(n^n)} + \underbrace{1000^n}_{O(n^n)})(\underbrace{10}_{O(n^3)} + \underbrace{n^3}_{O(n^3)} + \underbrace{300n^2}_{O(n^3)}) = O(n^n \cdot n^3) = O(n^{n+3})$

5. (a) CLAIM: $1000 = O(\log n)$, I.E., $\exists\, C_1 > 0, N$ SUCH THAT $\forall n \geq N$, $1000 \leq C_1 \cdot \log n$

PROOF: TAKE $C_1 = 1000 > 0$
& $N = 2$.

LET $n \geq N$ BE GIVEN
↳ SO $n \geq 2$.

THEN $C_1 \cdot \log n \geq 1000 \cdot \log 2 = 1000 \cdot 1 = 1000$. ∎

SCRATCH WORK: (LOTS OF POSSIBLE ANSWERS!)
COULD JUST TAKE $C_1 = 1000$, THEN WE JUST NEED $1 \leq \log n$, SO $n \geq 2 \to N$

(* COULD ALSO TAKE, E.G., $C_1 = 1$ & $N = 2^{1000}$)

↳ IN GENERAL, BE CAREFUL WITH THE BASE OF log'S! WE'LL ALWAYS TAKE $\log_2$ BECAUSE THIS IS CSCI.

(b) CLAIM: $100n = O(n^2)$, I.E., $\exists\, C_1 > 0, N$ SUCH THAT $\forall n \geq N$, $100n \leq C_1 \cdot n^2$

PROOF: TAKE $C_1 = 100 > 0$
& $N = 1$.

LET $n \geq N$ BE GIVEN
↳ SO $n \geq 1$

THEN $n \leq n^2$, SO $100n \leq 100n^2 = C_1 \cdot n^2$. ∎

SCRATCH WORK: (AGAIN, LOTS OF POSSIBLE ANSWERS!)
COULD TAKE $C_1 = 100$, SO THAT WE JUST NEED $n \leq n^2$ — WHICH IS TRUE $\forall n \in \mathbb{N}$.

(* COULD ALSO TAKE, E.G., $C_1 = 1$ & $N = 100$)

(c) CLAIM: $3^n = \Omega(100 \cdot 2^n)$, I.E., $\exists\, C_1 > 0, N$ SUCH THAT $\forall n \geq N$, $3^n \geq C_1 \cdot 100 \cdot 2^n$

PROOF: TAKE $C_1 = \frac{1}{100} > 0$

& $N = 1$.

LET $n \geq N$ BE GIVEN

↳ SO $n \geq 1$

THEN $3^n \geq 2^n = \frac{1}{100} \cdot 100 \cdot 2^n = C_1 \cdot 100 \cdot 2^n$. ∎

SCRATCH WORK: (LOTS OF POSSIBLE ANSWERS)

COULD TAKE $C_1 = \frac{1}{100}$, SO THAT WE JUST NEED $3^n \geq 2^n$ — WHICH IS TRUE $\forall n$

(* COULD ALSO TAKE, E.G., $C = 1$ & $N \geq \log_3 100 = \frac{\log 100}{\log 3}$)

(d) $100n^2 = \Theta(n^2)$ MEANS ① $100n^2 = O(n^2)$ $\wedge$ ② $100n^2 = \Omega(n^2)$,

SO THIS IS TWO LITTLE SUB-PROOFS:

① CLAIM: $100n^2 = O(n^2)$, I.E. $\exists\, C_1 > 0, N$ SO THAT $\forall n \geq N$, $100n^2 \leq C \cdot n^2$

(JUST LET $C_1 = 100$)

PROOF: TAKE $C_1 = 100 > 0$

AND $N = 1$.

LET $n \geq N$ BE GIVEN.

↳ SO $n \geq 1$

THEN $100n^2 = C_1 \cdot n^2$, SO $100n^2 \leq C_1 \cdot n^2$. ∎

② CLAIM: $100n^2 = \Omega(n^2)$, I.E., $\exists\, C > 0, N$ SO THAT $\forall n \geq N$, $100n^2 \geq C_1 \cdot n^2$

PROOF: TAKE $C_1 = 100 > 0$

AND $N = 1$.

LET $n \geq N$ BE GIVEN

↳ SO $n \geq 1$

THEN $100n^2 = C_1 \cdot n^2$, SO $100n^2 \geq C_1 \cdot n^2$ ∎

!! NOTE THAT IN GENERAL, THE "C_1" IN THESE TWO HALVES OF A BIG-Θ PROOF COULD BE DIFFERENT — IN THIS SIMPLE EXAMPLE, THEY ENDED UP THE SAME!

6. (a) COMPARING A TO EACH OF THE 1023 ELEMENTS AND NEVER FINDING IT WILL MAKE 1023 COMPARISONS.

(b) STARTING WITH 1023 ELEMENTS:

- COMPARING IT TO THE MIDDLE ELEMENT a_{511} OF $a_0, \ldots, a_{1022}$ WILL LEAVE US LOOKING AT $\frac{1}{2}(1023-1) = 511$ ELEMENTS, DEPENDING ON WHICH SIDE OF a_{511} A LIES.
- THE NEXT COMPARISON WILL LEAVE US WITH $\frac{1}{2}(511-1) = 255$ ELEMENTS TO LOOK AT;
- THE NEXT LEAVES US WITH $\frac{1}{2}(255-1) = 127$;
- THEN $\frac{1}{2}(127-1) = 63$;
- THEN $\frac{1}{2}(63-1) = 31$;
- THEN $\frac{1}{2}(31-1) = 15$;
- THEN $\frac{1}{2}(15-1) = 7$;
- THEN $\frac{1}{2}(7-1) = 3$;
- THEN $\frac{1}{2}(3-1) = 1$;
- AND ONE FINAL COMPARISON TO THE ONE LEFT.

10 COMPARISONS;

THIS IS EXACTLY $\log(1023+1)$

(i) $\underset{\frac{0+1022}{2}}{a_{511}}$, $\underset{\frac{512+1022}{2}}{a_{767}}$, $\underset{\frac{768+1022}{2}}{a_{895}}$, $\underset{\frac{896+1022}{2}}{a_{959}}$, $\underset{\frac{960+1022}{2}}{a_{991}}$, $\underset{\frac{992+1022}{2}}{a_{1007}}$, $\underset{\frac{1008+1022}{2}}{a_{1015}}$, $\underset{\frac{1016+1022}{2}}{a_{1019}}$, $\underset{\frac{1020+1022}{2}}{a_{1021}}$, a_{1022}

(MIDDLE ELEMENTS OF THE REST, AT EACH STEP)

(ii) $\underset{\frac{0+1022}{2}}{a_{511}}$, $\underset{\frac{0+510}{2}}{a_{255}}$, $\underset{\frac{0+254}{2}}{a_{127}}$, $\underset{\frac{0+126}{2}}{a_{63}}$, $\underset{\frac{0+62}{2}}{a_{31}}$, $\underset{\frac{0+30}{2}}{a_{15}}$, $\underset{\frac{0+14}{2}}{a_{7}}$, $\underset{\frac{0+6}{2}}{a_{3}}$, $\underset{\frac{0+2}{2}}{a_{1}}$, a_0

(c) AS IN PART (a), THE LINEAR SEARCH WILL MAKE $n = 2^k - 1$ COMPARISONS;
AS IN PART (b), THE BINARY SEARCH WILL MAKE $\log(n+1)$ COMPARISONS.

$\therefore$ THE LINEAR SEARCH IS $O(n)$, AND THE BINARY SEARCH IS $O(\log n)$

$\hookrightarrow \log(n+1) \le \log(n+n) = \log(2n) = 1 + \log n$, WHICH IS $O(\log n)$

7. E.G.:

1	3	7	2	4
2	1	0	1	5
5	2	3	1	4
1	6	1	2	0
2	2	3	5	4

GOAL: FIND ↴ PATH FROM TOP-LEFT WITH MAXIMUM SUM OF #'S ENCOUNTERED.

(a) BRUTE FORCE SEARCH (OF ALL SUCH PATHS):

(i) THERE ARE $\binom{8}{4} = 70$ SUCH PATHS TO SEARCH

(ii) EACH PATH ENCOUNTERS 9 NUMBERS, SO THE SUM TAKES 8 ADDITIONS

(iii) IN TOTAL, THIS GIVES $70 \cdot 8 = 560$ ADDITIONS.

(b): DYNAMIC PROGRAMMING: FOR THE EXAMPLE ABOVE:

(i)

1	3	7	2	4
2	1	0	1	5
5	2	3	1	4
1	6	1	2	0
2	2	3	5	4

(2·1 ADDITIONS)

1	4	7	2	4
3	1	0	1	5
5	2	3	1	4
1	6	1	2	0
2	2	3	5	4

(2·2 ADDITIONS)

1	4	11	2	4
3	5	0	1	5
8	2	3	1	4
1	6	1	2	0
2	2	3	5	4

(2·3 ADDITIONS)

1	4	11	13	4
3	5	11	1	5
8	10	3	1	4
9	6	1	2	0
2	2	3	5	4

(2·4 ADDITIONS)

1	4	11	13	17
3	5	11	14	5
8	10	14	1	4
9	16	1	2	0
11	2	3	5	4

(2·4 ADDITIONS)

1	4	11	13	17
3	5	11	14	22
8	10	14	15	4
9	16	17	2	0
11	18	3	5	4

(2·3 ADDITIONS)

1	4	11	13	17
3	5	11	14	22
8	10	14	15	26
9	16	17	19	0
11	18	21	5	4

(2·2 ADDITIONS)

1	4	11	13	17
3	5	11	14	22
8	10	14	15	26
9	16	17	19	25
11	18	21	26	4

(2·1 ADDITIONS)

1	4	11	13	17
3	5	11	14	22
8	10	14	15	26
9	16	17	19	26
11	18	21	26	30

MAXIMUM POSSIBLE SUM IS 30 (CAN BACKTRACK ALONG LARGEST NEIGHBORS ↑/← TO FIND A PATH)

(ii) $4\cdot(1+2+3+4) = 40$ ADDITIONS

(c) THE DYNAMIC-PROGRAMMING ALGORITHM IS QUITE A BIT MORE EFFICIENT! (40 VS 560 ADDITIONS)

WHAT MAKES THIS POSSIBLE IS THAT MANY, MANY OF THE 560 ADDITIONS INVOLVED THE SAME NUMBERS (THERE ARE ONLY 25 #'S IN THE GRID!), AND THAT KNOWING THE MAXIMUM SUM POSSIBLE TO EACH POINT ALONG THE WAY AS WE BUILD DOWNWARD & RIGHTWARD DICTATES THE REST OF THE MAXIMUM SUMS TO OTHER POINTS PAST IT.

(d) FOR A GENERAL $n \times n$ GRID:

- BRUTE-FORCE CHECKS $\binom{2(n-1)}{n-1}$ PATHS, EACH WITH $2(n-1)$ ADDITIONS, SO $(2n-2)\dfrac{(2n-2)!}{(n-1)!^2}$ ADDITIONS
- DYNAMIC PROGRAMMING TAKES $4(1+2+\cdots+n-1) = 4\,\dfrac{(n-1)(n)}{2} = \underbrace{2(n^2-n)}_{O(n^2)}$ ADDITIONS.

(e) WITH A LITTLE COMPUTATIONAL HELP:

- BRUTE-FORCE FOR $n=11$ GIVES < 1 BILLION ADDITIONS, AND FOR $n=12$ GIVES > 1 BILLION $\therefore$ $n=12$.
- DYNAMIC PROGRAMMING FOR $n=22361$ GIVES < 1 BILLION ADDITIONS, AND FOR $n=22362$ GIVES > 1 BILLION $\therefore$ $n=22362$.

!