O-notation–O(g(n))={f(n):exists positive constants c and n0 such thatv0≤f(n)≤cg(n) for all n≥n0.



g(n) is an asymptotic upper bound for f(n). Omega notation Omega f(n): exists pos const’s c and n0 such that 0≤cg(n)≤f(n) for all n≥n0. g(n) is an asymptotic lower bound for f(n). Theta notation Theta f(n): exists pos const’s c1, c2 and n0 such that 0≤c1g(n)≤f(n)≤ c2g(n) for all n≥n0. g(n) is an asymptotic tight bound for f(n).



**If** f1(n)=O(g1(n)) and f2(n)=O(g2(n)) then f1(n)\*f2(n) = O(g1(n)g2(n) == since f1(n)=O(g1(n)), there exists c1 & n1 such that0≤ f1(n)≤c1g1(n) for n≥0. since f2(n)=O(g2(n)) there exists c1 & n1 such that 0≤ f2(n)≤c2g2(n) for n≥0. Now… 0≤ f1(n)\*f2(n)≤c1g1(n)\* c2g2(n). Let c3 = c1\*c2, then 0≤ f1(n)\*f2(n)≤c3g1(n)\*g2(n) for all n≥n3, where n3=max(n0, n2). Therefore f1(n)f2(n)=O(g1(n)g2(n)). logn=O(sq.n) and logn=O(nb) and 2log2n = n.

to find the boundary of 2 functions do lim(infinity)[ t1/t2 ]. If t2 >t1, then t1 is bounded above by t2(meaning it’s O). if t1>t2, t2 bounds t1 below (BIG OMEGA).

Make sure to get the Fibonacci algorithm down on your sheet!!

Both the recursive and DP