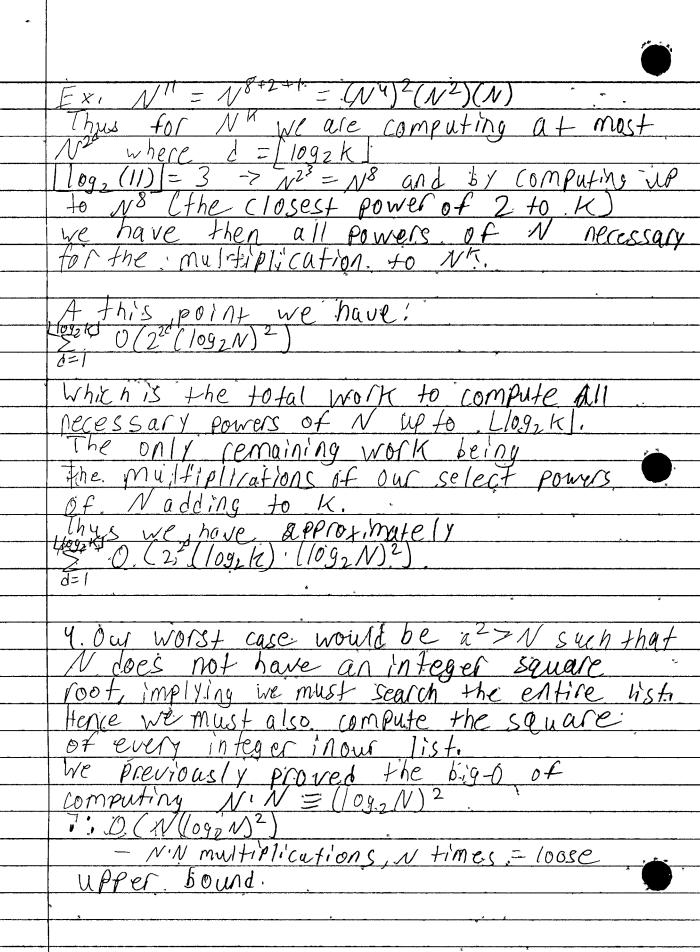
Austin Bennett Final Question #2! Determining the number of bits needed to represent a number N is much like betermining the number of digits

-7/10910(N) 7 where I 7 is rounding up. To determine the number of bits we would take [log2(N)]. The Big-6 would then be O(log2(N)). 2. The worst case of multiplying two integers between I and N would be N.N as they have the highest number of bits. The process of binary multiplication consists of multiplying each bit of one number by all of the bits of the other number + Some addition al constant time for carrying bits. So Multiplying N'N Would require each bit (total of ligz(N)) by all of the bits of N. (1092(N)). Thus O(1092(N)²) 3. Fast exponentation for NK would be !ike NEN NZ =(N)2 N = (NZ)2 N8 = (N4)2 where we are simply repeatedly squaring powers of 2 until we have exponent values adding up to K.



Austin Bennett Final Question #2: The complexity of an ordinary binary Search is O(log N). Our binary search is inique in the cense that we are really searching for Val but we do not know an algorithm for finding square roots so we must compute at and compare to N.

Thus, in addition to computing log(N) to Shrink our list size we must also compute at =7 worst case N2,

i. D((log_2N)^2 · (log_2N)) = D((log_2N)^3) 6. Again the worst case is when $a^{K} > N$ Such that lito &var, we do not have $a^{K} = N$, so we must compute a^{K} , sint times.

Assuming the worst possible case we are computing N^{K} , N times.

No can be expressed as $N \cdot N$, K times or $K(log_{2}(N)^{2})$ i. $O(N)(log_{2}(N)^{2})$ 7. Ordinary binary search: O(log N) For each divide and conquer we are instead (om puting NK => O(K(1092(N)2) O(1092N)O(K1092(N)2) = O(K(1092(N)3))

Note: Theoretically our best choice of largest K would be [109KN] but Kis originally un known and thus we must use 'Llogz NJ as to ensure we do not miss possible Perfect 100ts. The smallest value of K that may need to be considered is 1 because N/o => undefined The largest value of K necessary is 10,92 N Since we are attempting Perfect Kth root numbers. Taking the Kth root > 109, N is futile as Strictly smaller than , . O(1092 N boolean Kth Root (int 100pi=1 to N 100Pi=1 to Llog_N Print"; is a perfect ith power CHUM + HUEL return false; de4 binary-search (list x, int N); ix 11:st | = 0 : return - 1 let midpoint be the middle value inthe 11st if KthRoot (midpoint) == 1 return Position of mid Polyt. if midpoint < N : return binary search lupper half Of the list, N) if midpoint >N: return binary search (lower half of list, N)

Austin Bennett Final Question #2 9, (ontinued., Ordinary binary search + computing
all Kth powers for each N = total work $\frac{1092 \times 10(1092 N) - binary search}{\geq 0(2^{2d}(1092 N)^2) - computing N^{K}, doing fhis}$ $\frac{Multiple times, of K many times.}{\geq 1092 \times 10(2^{2d} \times (1092 N)^2)}$ We compute the above, N many times for I to N to find all perfect 1th roots. total work = 2 0 (22 NK (10.92 N)3)