Conor Sweeney - cjs2201

COMS W3134 Spring 2016

HW 1 Written Section

2/5/2016

1. Weiss, Exercise 2.1: Order the following functions by growth rate: N, √N, N^1.5, N^2, NlogN, NloglogN, Nlog^2N, 2/N, 2^N, 2^(n/2), 37, N^2logN, N^3. Indicate which functions grow at the same rate.
   1. In increasing order: 2/N, 37, √ N, N, N log log N, N log N, Nlog(N2 ), N log^2 N, N^1.5 , N^2 , N^2 log N, N^3 , 2^N/2 , 2^N . N log N and N log(N2) grow at the same rate.
2. Weiss, Exercise 2.6: In a recent court case, a judge cited a city for contempt and ordered a fine of $2 for the first day. Each subsequent day, until the city followed the judge’s order, the fine was squared (that is the fine progressed as follows $2, $4, $16, $256, $65,536…)
   1. What would be the find on day N?
      1. int fine = 2;
      2. for (int i = 0, i < N, i++){
      3. fine = fine\*fine;
      4. return fine;
   2. How many days would it take the fine to reach D dollars? (A Big-Oh answer will do)
      1. int fine = 2;
      2. if (D <= 2){
      3. return 1;
      4. }
      5. else{
      6. for (int i = 2, , i++){
      7. fine = fine\*fine;
      8. if (fine >=D){
      9. return i;
      10. }
3. Give an analysis of the Big-Oh running time for each of the following program fragments:
   1. int sum = 0;
   2. for ( int i = 0; i < 23; i ++)
   3. for ( int j = 0; j < n ; j ++)
   4. sum = sum + 1;
   5. O (N) j can be a large as n and since the outer loop only runs 23 times it the running time is proportional to 23 \* N
   6. int sum = 0;
   7. for ( int i = 0; i < n ; i ++)
   8. for ( int k = i ; k < n ; k ++)
   9. sum = sum + 1;
   10. O(N ^ 2) : k and i can be as large as n – 1, thus since both get larger the running time is equal to N \* N
   11. public int foo(int n, int k) {
   12. if(n<=k)
   13. return 1;
   14. else
   15. return foo(n/k,k) + 1;
   16. }
   17. O(N/2) : N is being returned exponentially smaller each time the recursive method is called. k is always constant as the function recursively calls itself and therefore 2 is the appropriate answer since it is the simplest form.
4. Weiss Exercise 2.11: An algorithm takes 0.5 ms for input size 100. How long will it take for input size 500 if the running time is the following (assume low-order terms are negligible):
   1. Linear
      1. 500/100 = N 0.5 ; N = 2.5 ms
   2. O(NlogN)
      1. (500 log 500) /(100 log 100) = N 0.5; N = 3.3737 ms
   3. Quadratic
      1. 500^2 /100^2 = N 0.5; N = 12.5 ms
   4. Cubic
      1. 500^3 /100^3 = N 0.5; N = 162.5 ms
5. Give an efficient algorithm to determine if there exists an integer I such that Ai = i in an array of integers A1<A2<A3<…<An. What is the running time of your algorithm?
   1. Since A(i) is always less than A(i+1) therefore as i increases so does the value of A(i).
   2. Using a binary search method to find the value as fast as possible since the values are predictably increasing. i at a rate of i = i+1 and A[i] needs to be less than i since it needs to be increasing by more that +1 and have a value less than i or else i will not be able to catch up to its value
   3. The runtime for this algorithm is O(logN)
   4. The algoritm:

//If -1 is returned then there is no such value

//else the answer will be returned

int i = 0; //first value

int max = A.length; //maximum size of the array

if (A[0] > 1){

return -1; // i only increases by 1 so it will not catch up if this is untrue.

}

else if {

for ( , i < max, ){

int mid = (i + max)/2;

if( A[mid] = mid + 1;){ //since the array begins counting at 0 add 1 to get i

return mid + 1;

}

else if (A[mid] >( mid +1)){

max = mid -1;

}

else if (A[mid] <( mid +1)){

i = mid + 1;

}

}

}

return -1; //if the loop ends without returning a value then no solution exists