Xi Liu, xl3504, midterm, N15017945

“I understand the ground rules and agree to abide by them. I will not share ideas, solutions, or assist

another student during this exam, nor will I seek assistance from another student or attempt to view

their ideas or solutions.”

Name: Xi Liu

NYU Net ID: xl3504

Problem 1

(a)

f(n) = O(g(n))

since polynomial functions are asymptotically greater than logarithm functions

lim\_{n -> infinity} n^5 (log n)^50 / (n^5.1 (log n)) = 0

which means the function at the denominator (n^5.1 (log n)) is asymptotically greater than the function at the numerator (n^5 (log n)^50).

(b)

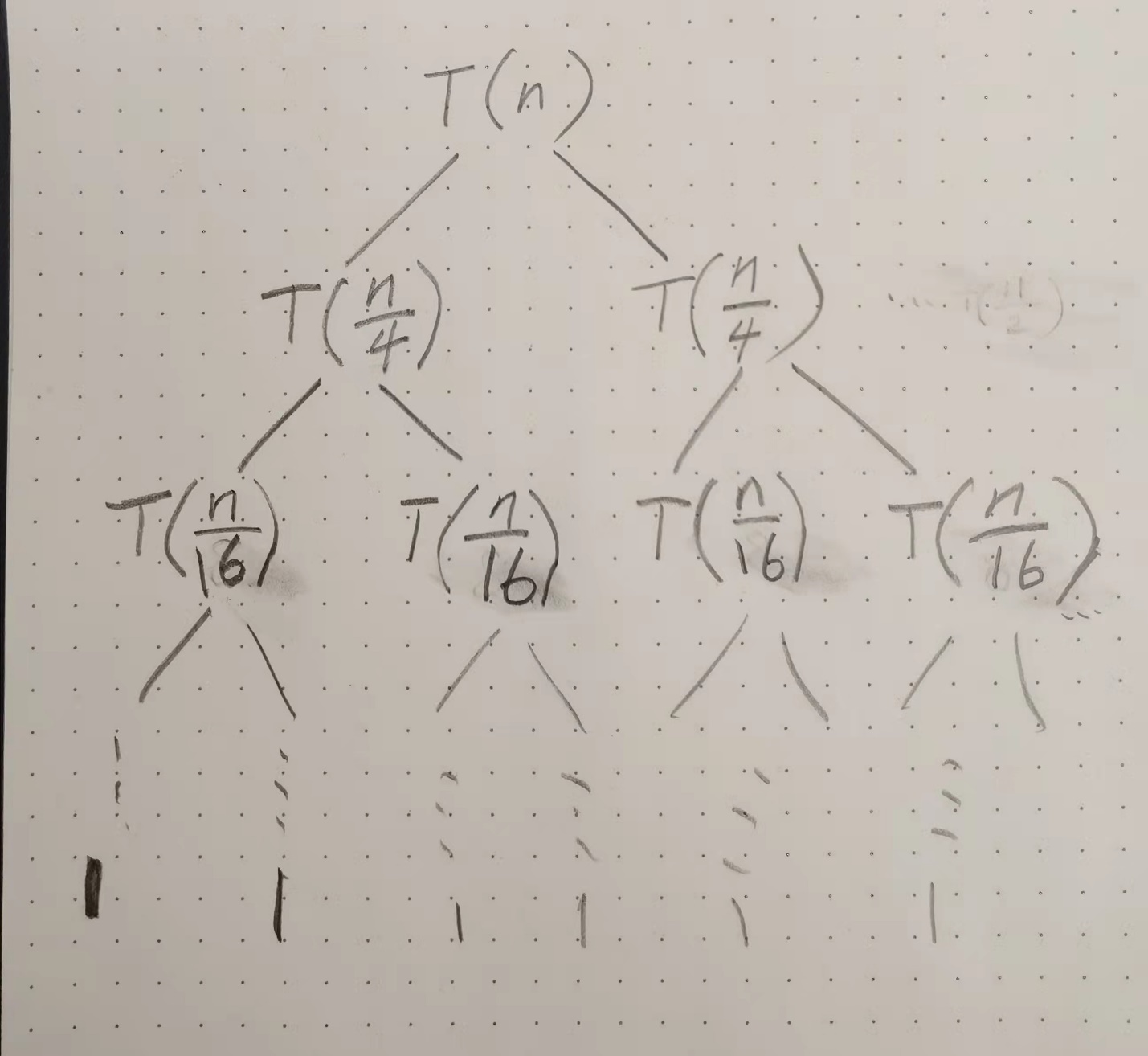
f(n) = Omega(g(n))

since exponential functions (with a bigger exponent) are asymptotically greater than polynomial functions

lim\_{n -> infinity} 5^n / (125^{sqrt{n}} + n^2) = infinity

which means the function at the denominator (125^{sqrt{n}} + n^2) is asymptotically less than the function at the numerator (5^n).

(c)



cost at depth i is equal to n / 2^i

number of levels = log\_2 n

T(n) = sum\_{i = 0}^{log\_2 n - 1} (n / 2^i)

= n \* sum\_{i = 0}^{log\_2 n - 1} (1 / 2^i)

= Theta(n / 2^{log\_2 n})

Problem 2

(a)

initially left = 1, right = A.length

calculate mid = (left + right) / 2

recursively call pos\_after\_neg(A, temp, left, mid) and pos\_after\_neg(A, temp, mid + 1, right), array A is divided into A[left…mid] and A[mid+1…right], then copy the negative elements of A[left…mid] and A[mid+1…right] to array temp, then copy the positive elements of A[left…mid] and A[mid+1…right] to array temp. finally copy the elements in array temp back to A

def merge(a, temp, left, mid, right):

t\_i = left

for i in range(left, mid + 1):

# negative from left

if a[i] < 0:

temp[t\_i] = a[i]

t\_i += 1

for i in range(mid + 1, right + 1):

# negative from right

if a[i] < 0:

temp[t\_i] = a[i]

t\_i += 1

for i in range(left, mid + 1):

# positive from left

if a[i] >= 0:

temp[t\_i] = a[i]

t\_i += 1

for i in range(mid + 1, right + 1):

# positive from right

if a[i] >= 0:

temp[t\_i] = a[i]

t\_i += 1

for i in range(left, right + 1):

a[i] = temp[i]

def pos\_after\_neg(a, temp, left, right):

if left < right:

mid = (left + right) // 2 # integer division

pos\_after\_neg(a, temp, left, mid)

pos\_after\_neg(a, temp, mid + 1, right)

merge(a, temp, left, mid, right)

(b)

use i and j as 2 pointers, i initially holds the index of the beginning of the array, j initially holds the index of the end of the array.

iteratively find the positive numbers at a[i] and negative numbers at a[j],

if a positive number is encountered at a[i], swap a[i] with the element with current a[j], then i = i + 1, j = j – 1

(c)

initially left = 1, right = A.length

calculate mid = (left + right) / 2

recursively call the function with A[left…mid] and A[mid+1…right],

if A[left] > 0, return A[left]

test if A[left] <= 0 and A[mid] <= 0, everything is negative or 0 in between, then exit out of recursion

if A[left] <= 0 and A[mid] > 0, recursively call the function again

if A[mid + 1] <= 0 and A[right] > 0, recursively call the function again

if A[mid + 1] <= 0 and A[right] <= 0, then there is no positive number in the array

Problem 3

(a)

base case: T(0) = 0

number of cookies from 0 boxes is 0

T(n) =

{

max\_{k <= i <= n} (c[i] + T(n - i))

}

boxes must be at least k distance apart, so i >= k, once chosen the box c[i], the remaining amount of box is n – i, so need to add c[i] with the optimal solution of the subproblem with size n – i

(b)

int memo[n];

for(int i = 1; i < n; ++i)

{

memo[i] = c[i];

for(int j = 1; j < i; ++j)

{

if(i – j >= k) /\* boxes must be more than k distance apart \*/

memo[i] = max(memo[i], c[j] + memo[i - j]);

}

}

return memo[n];

(c)

time complexity = Theta(n^2)

2 levels of for loop

operations within the innermost level are constant, values in c and memo are readily available to be fetched