These practice problems are different from the problems for previous topics because they aren't programming problems. All you need to do is look at functions and state their complexity.

For each of these functions, you need to decide on the worst-case complexity of the function. Your choices are \(O(1), O(\log N), O(N), O(N \log N), O(N^2), O(N^3), O(2^N)\).

Most of these functions came from old 121 quizzes and final exams.

I am not going to post solutions for these. I know that's unpopular, but I really believe we all learn better if we aren't tempted to peek at the answers before we think hard about a question.

However, please feel free to talk about these problems on the practice problem solution forum. If you think you know the answer for one of these problems, make a post about it. It will keep things easier for everyone if we use a separate thread for each problem, so if you have answers for more than one problem make a separate post for each of them.

If someone posts an answer and you don't agree, reply to say so and give your reasons. Start a dialog online. No prizes on the forum for right answers; the goal is to have a discussion and learn something. Sometimes we learn more from starting out with the wrong answer and understanding where we went wrong than we do from getting things right at the start.

I will be monitoring the forum and if there is discussion about a problem that doesn't resolve itself, or if students are agreeing on the wrong answer I will chime in with a comment. But I'll let you discuss for a while first before I join in.

def complexA(numbers):
    N = len(numbers)  # N is the problem size
    result = 0
    for i in range(N):
        j = 1
        while j < N:
            result += numbers[i] * numbers[j]
            j = j * 2
    return result

def complexB(numbers):
    N = len(numbers)  # problem size
    done = False
    result = 0
    i = 1
    while i < N and not done:
        if numbers[i] < 1000:
            result += numbers[i]
        else:
            done = True
        i += 1
    return result
def complexC(mylist):
    # problem size (n) is len(mylist)
    for i in mylist:
        for j in range(100):
            print i, j
        if i < 25:
            for k in range(len(mylist)):
                print mylist[k], j
        else:
            for k in range(len(mylist)):
                print k

def complexD(numbers):
    N = len(numbers)  # problem size
    answer = 0
    for i in range(N):
        term = 0
        j = N-1
        while j > i:
            term = term * numbers[j]
            j = j - 1
        answer += term
    return answer

def complexE(numbers):
    N = len(numbers)  # problem size
    result = ""
    if N > 2:
        i = 1
        while i < N:
            result += "X"
            i = i * 3
    else:
        i = 0
        while i < N:
            result += "Y"
            i = i + 1
    return result
def complexF(numbers):
    N = len(numbers) # problem size
    result = 0
    for i in range(1000):
        for j in range(N):
            for k in range(N):
                index = (i*j*k) % N
                result += numbers[index]
    return result

def complexG(numbers):
    n = len(numbers) # problem size
    answer = 0
    i = n
    while i >= 0:
        temp = 5
        j = 1
        while j < n:
            temp *= numbers[j]
            j = j * 2
        answer += temp
        i = i - 3
    return answer

def complexH(numbers):
    n = len(numbers) # problem size
    for i in range(n):
        if numbers[i] < 10:
            sum = 0
            for j in range(n):
                k = 0
                while k < n:
                    sum += numbers[j]*numbers[k]
                    k += 50
        else:
            sum = 47
    print i, sum
def complexI(numbers):
    n = len(numbers)  # problem size
    answer = 1
    if n > 100:
        for i in range(100):
            answer *= numbers[i]
    else:
        for i in range(n):
            for j in range(n//2):
                answer += (numbers[i]*numbers[j])
    return answer

def complexJ(lis):
    n = len(lis)  # n is the problem size
    answer = 1
    i = 0
    done = False
    while not done:
        if lis[i] == 0:
            answer += 5
        elif lis[i] > 0:
            answer *= 2
            lis[i+1] = -2*lis[i]
        else:
            answer -= 2
            done = True
        i += 1
    answer += lis[n-1]
    return answer