UVA DATA SCIENCE

CS 5012: Foundations of Computer Science

Asymptotic Complexity Exercise

Given the following code snippets, provide the worst case time complexity in the form of Big-O notation. Justify your response and state any assumptions made. Treat these functions as constant runtime: print(), append()

```
▶ def measure(inputList):
      int n = len(inputList)   O(1)
      int sum = 0;  O(1)
      for i in range(0, n): O(n)
         for j in range(0, 5): O(1)
            sum+= j * inputList[i] O(1)
         for k in range(0, n): O(n)
            sum -= inputList[k] O(n)
```

2O(1)+O(n)

nx(O(1)+O(1)+O(n)+O(n))
=2O(n)+2O(n^2)

2O(1)+3O(n)+2O(n^2)

The asymptotic complexity of this algorithm is: O (n^2)

```
▶ def addElement(ele):
      myList =[] O(1)
      myList.append(666) O(1)
      print myList O(1)
```

3O(1)

The asymptotic complexity of this algorithm is: O (1)

```
▶ num = 10  O(1)

  def   addOnesToTestList(num):
    testList = []  O(1)                          2O(1)
    for i in range(0,num): O(1)
        testList.append(1)O(1)                   3O(1)
        print(testList) O(1)

                                                 6O(1)
    return testList O(1)
```

The asymptotic complexity of this algorithm is: O (1)

```
▶ testList = [1, 43, 31, 21, 6, 96, 48, 13, 25, 5]  O(1)

  def someMethod(testList):
    for i in range(len(testList)):O(1)
        for j in range(i+1, len(testList)): O(1)
           if testList[j] < testList[i]: O(1)
             testList[j], testList[i] = testList[i], testList[j] O(1)
           print(testList) O(1)
```

The asymptotic complexity of this algorithm is: O (1)

```
▶ def searchTarget(target_word):
  # Assume range variables are unrelated to size of aList

    for (i in range1): O(1)
        for (j in range2): O(1)
             for (k in range3): O(1)
               if (aList[k] == target_word):O(1)
                   return 1  O(1)

         return -1 O(1)
    return -1 O(1)
```

The asymptotic complexity of this algorithm is: O (1)

```
▶ def someSearch(sortedList, target):
      left = 0                                  O(1)
      right = len(sortedList)- 1                O(n)

      while (left <= right):                    O(n)
         mid = (left + right)/2                 O(1)
         if (sortedList(mid)==target):          O(n)
             return mid                         O(1)
         elif(sortedList(mid) < target):        O(n)
             left = mid + 1                     O(1)
         else:
               right = mid - 1                  O(1)
      return -1                                 O(1)
```

O(1)+O(n)

→ O(n)+n*(O(1)+O(n)+O(1))
=O(n)+O(n)+O(n^2)+O(n)

O(1)+4O(n)+O(n^2)

The asymptotic complexity of this algorithm is: O (n^2)

```
▶ #Assume data is a list of size n
    total = 0                   O(1)
    for j in range(n):          O(n)
        total += data[j]        O(1)
    big = data[0]               O(1)
    for k in range(1,n):        O(n)
        big = max(big,
    data[k])                    O(1)
```

O(1)+O(n)+n*O(1)+O(1)+O(n)+n*O(1)

The asymptotic complexity of this algorithm is: O (n^2)

```
▶   powers = 0
    k = 1
    while k < n:
        k = 2*k
        powers += 1
```

The asymptotic complexity of this algorithm is: O (_____)

```
▶   k = 1
    while k < n:
        for j in range(k):
            steps += 1
        k = 2*k
```

The asymptotic complexity of this algorithm is: O (_____)

►
```
for k in range(1,n):
    j = 1
    while j < k:
        total += 1
        j = 2 * j
```

The asymptotic complexity of this algorithm is: O (_____)