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):        n *O(1)
          sum+= j * inputList[i]    n*O(1) = O(n)
       for k in range(0, n):        n*O(n)
          sum -= inputList[k]       O(1)
```

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

T(n) = O(1) + O(1) + O(n) + O(1) + n*O(1) + O(n) + n*O(n) + O(1)

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

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

T(n) = O(1) + O(1) + O(1)

```
► num = 10                                          O(1)

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

    return testList                                 O(1)
```

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

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

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

T(n) = O(1) + O(n) + n*O(n) + n*n*O(1) + n*n*O(1) + O(1)

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

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

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

The asymptotic complexity of this algorithm is: O (n^3)

T(n) = O(n) + n*O(n) + n*n*O(n) + n*n*O(n) + O(1) + O(1)+O(1)

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

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

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

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

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

►
```
  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 (_____)