

Name _____ Section ____ Andrew Id: _____

15-112 Spring 2016 Quiz 5x practice

Short Answer: Answer each of the following very briefly

1. State and informally prove the worst case big-Oh of selection sort and merge sort.

2. If the runtime of a function increases by 12x when its input size is increased by a factor of 4, what is its probable big-Oh?

3. Consider two functions, f and g . f is $O(n)$, and g is $O(2^{**}n)$. Which of the following statements is true? Why?

- f is faster than g for all inputs
- g is faster than f for all inputs
- Neither of the above

4. Why can't we add lists to sets?

5. Suppose we have an unsorted list L and one element k . We want to determine if k is in L . What is the big-Oh runtime of the following two strategies?

- Sort L , then binary search through L for k .

- Linear search through L for k.

Now, suppose we have an unsorted list L (with length n) and n elements k_1, \dots, k_n . What are the runtimes of the following strategies to determine which of k_1, \dots, k_n are in L?

- Sort L, then binary search for each k.
- Linear search for each k.

6. Below each algorithm (as we discussed in class), write its Big-Oh runtime in terms of the length of the input:

- a) selection sort
- b) mergesort
- c) linear search
- d) binary search
- e) isPrime
- f) fasterIsPrime

7. Write a function that returns a list of all the unique values in a dictionary.

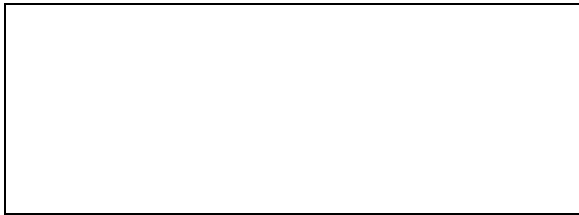
8. I claim that there are certain hash functions that make hashtables equivalent to lists. What is one such hash function?

Code tracing: Indicate what the following will print

```
def g(L, s):
    d = []
    for i in range(len(L)):
        d0 = dict()
        for j in range(len(L)):
            d0[L[j]] = chr(ord(s[j]) + i + j)
        d += [d0]
    return d

def ct1(d1, L):
    for d in L:
        d1.update(d)
    return d1

D = ct1({1: "p", 2:"i", 3:"k", 4:"a"}, g([0,1,2,3,4], "argon"))
for key in sorted(D):
    print(key, D[key])
```



Big-Oh: What is the the big-Oh of the following functions?

<pre>def bigOh0(L): n = len(L) for x in range(0, 2**n, 2**n/n**2): print("This may print a lot!")</pre>	O(_____)
<pre>def bigOh1(L): n = len(L) x = y = 0 while (x < n): while (y < n): print("y", end = " ") y += 3 print("x", end = " ") x += 4</pre>	O(_____)
<pre>def bigOh2(L): n = len(L) total = 0 z = n+n*n for x in range(z//17, z//3, 123): y = 1 while (y**2 < n): y += 1 total += x*y return math.log(total**3)</pre>	O(_____)

Free Response: oddOneOut(L)

You are given a list of letters that are shuffled around, with most of the letters occurring twice in the list. However, exactly one of the letters does not have a pair! Return which letter is the odd one out. You should write three different versions, one that runs in $O(n^2)$, $O(n \log n)$ and $O(n)$.