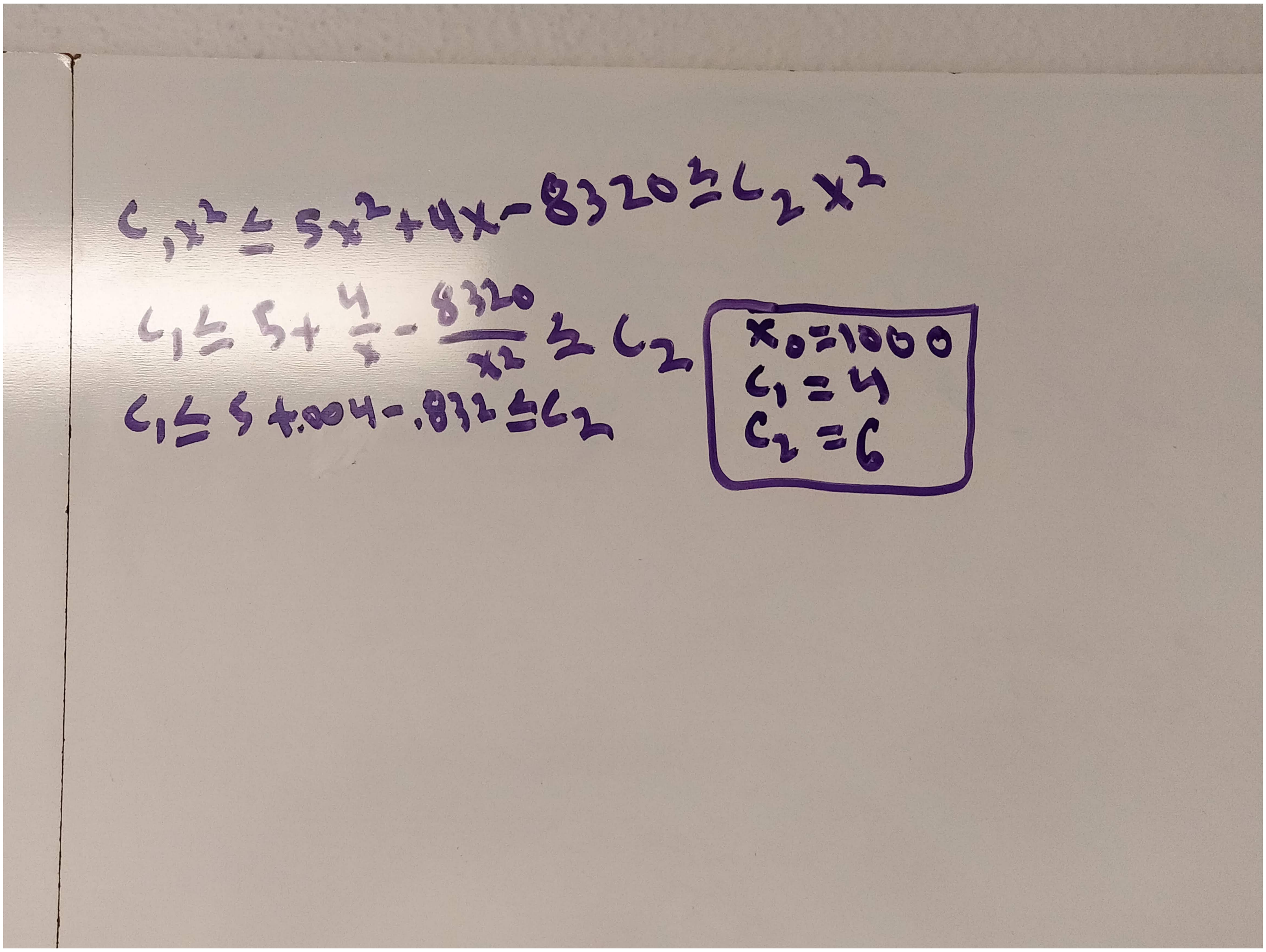
CSE5350 Homework 1: Tyler Giallanza

Problem 1



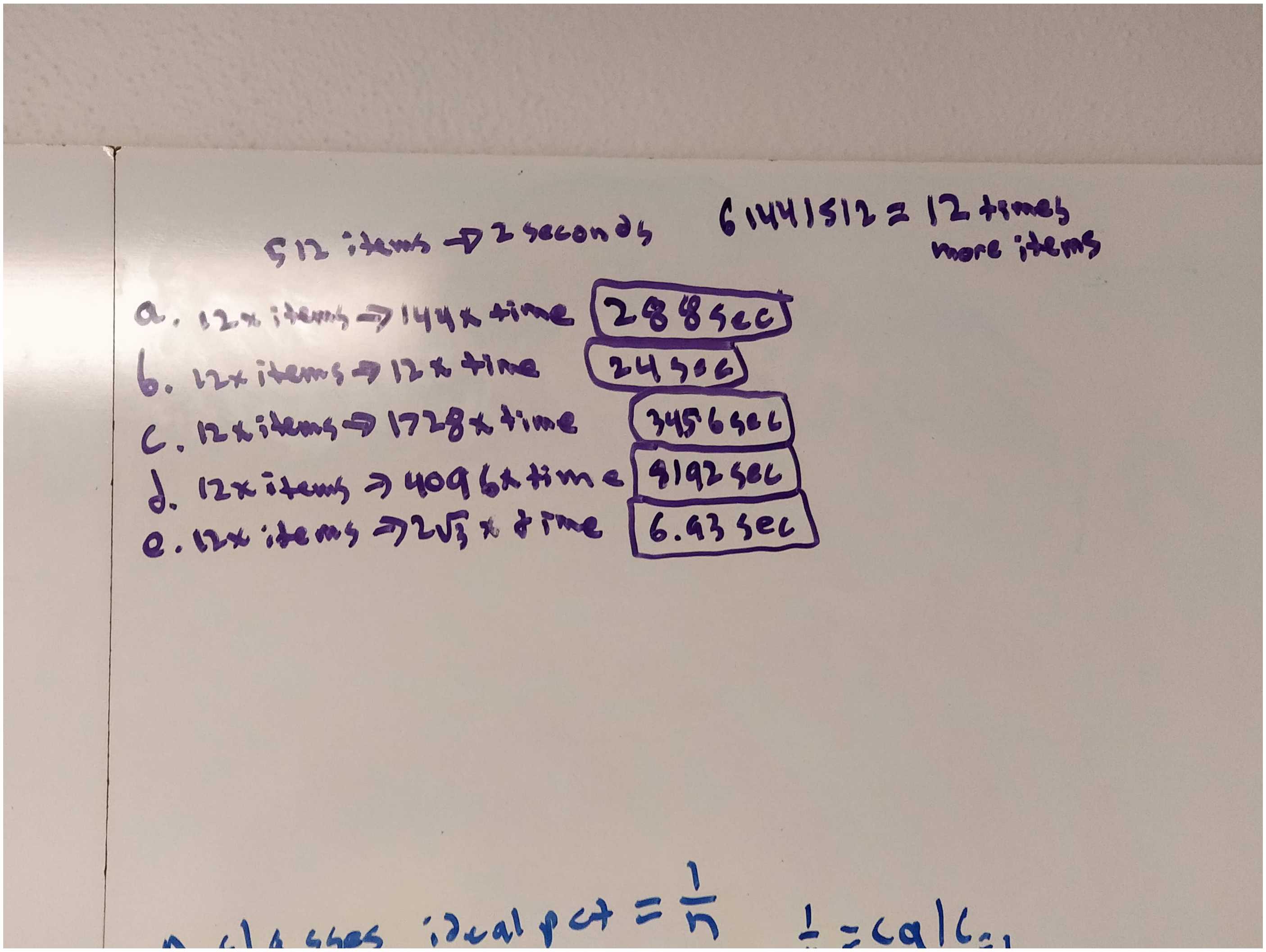
$$c_1 x^2 \leq 5x^2 + 4x - 8320 \leq c_2 x^2$$

$$c_1 \leq 5 + \frac{4}{x} - \frac{8320}{x^2} \leq c_2$$

$$c_1 \leq 5 + .004 - .832 \leq c_2$$

$$\boxed{\begin{array}{l} x_0 = 10000 \\ c_1 = 4 \\ c_2 = 6 \end{array}}$$

Problem 2



512 items → 2 seconds

6144/512 = 12 times more items

a. 12x items → 144x time $\boxed{288 \text{ sec}}$

b. 12x items → 12x time $\boxed{24 \text{ sec}}$

c. 12x items → 1728x time $\boxed{3456 \text{ sec}}$

d. 12x items → 4096x time $\boxed{8192 \text{ sec}}$

e. 12x items → $2\sqrt{3}$x time $\boxed{6.93 \text{ sec}}$

n classes ideal pct = $\frac{1}{n}$ 1 = calc

Problem 3

In [2]:

```
#!/usr/bin/env python

import random
import time

def random_array_count(n):
    arr = [random.randint(1,10) for _ in range(n)]
    count_arr = [0]*10
    for num in arr:
        count_arr[num-1] += 1
    return count_arr

def random_array_sort(n):
    arr = [random.randint(1,10) for _ in range(n)]
    for j in range(2,len(arr)):
        for i in range(j):
            if arr[j] < arr[i]:
                temp = arr[j]
                arr[j] = arr[i]
                arr[i] = temp

def random_array_fast_sort(n):
    arr = [random.randint(1,10) for _ in range(n)]
    arr = sorted(arr)

print('Testing array count for varying n...')
for n in [10,100,1000,10000,100000]:
    start_t = time.time()
    print(n,random_array_count(n),time.time()-start_t)

print('number of n that can be counted in three days: 4e13\n')
```

```
Testing array count for varying n...
10 [0, 2, 1, 1, 0, 3, 3, 0, 0, 0] 2.193450927734375e-05
100 [12, 11, 7, 10, 8, 12, 8, 13, 8, 11] 9.751319885253906e-05
1000 [112, 87, 95, 98, 102, 97, 112, 85, 91, 121] 0.00125694274902343
75
10000 [975, 966, 1055, 967, 1059, 1037, 954, 959, 999, 1029] 0.009650
468826293945
100000 [10077, 9980, 10044, 9935, 10005, 10063, 10005, 9877, 9995, 10
019] 0.09430122375488281
number of n that can be counted in three days: 4e13
```

Problem 4

In [3]:

```
print('Testing insertion sort for varying n...')
for n in [10,100,1000,10000]:
    start_t = time.time()
    random_array_sort(n)
    start_t = time.time()-start_t
    print(n,start_t)

print('number of n that can be insertion sorted in three days: 372,79
3\n')
```

```
Testing insertion sort for varying n...
10 2.4557113647460938e-05
100 0.0004425048828125
1000 0.024599075317382812
10000 2.0658860206604004
number of n that can be insertion sorted in three days: 372,793
```

Problem 5

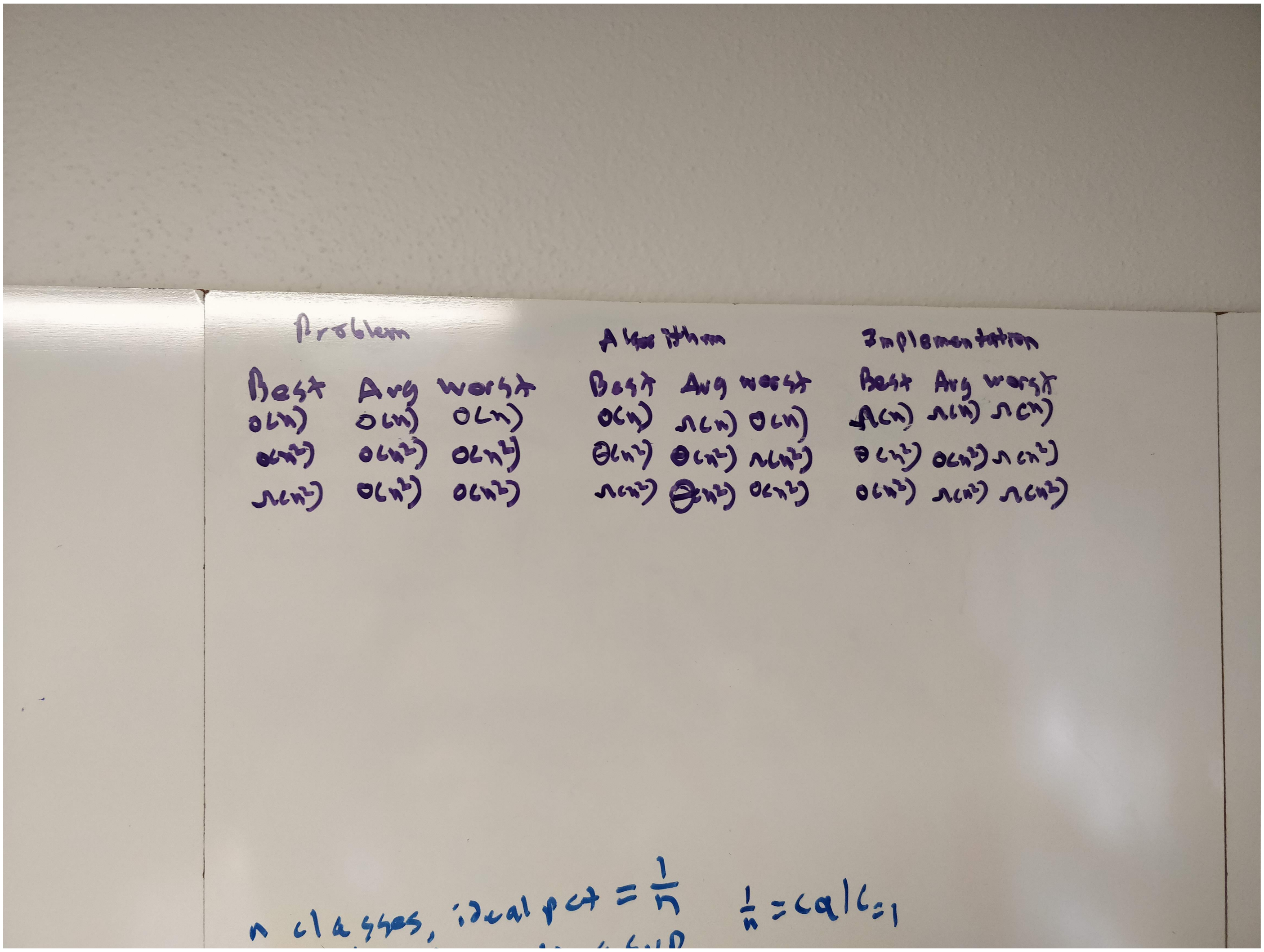
In [5]:

```
print('Testing python built-in sort for varying n...')
for n in [10,100,1000,10000]:
    start_t = time.time()
    random_array_fast_sort(n)
    start_t = time.time()-start_t
    print(n,start_t)

print('number of n that can be Python sorted in three days: 1e13')
```

```
Testing python built-in sort for varying n...
10 1.8596649169921875e-05
100 0.00010371208190917969
1000 0.0009565353393554688
10000 0.014125823974609375
number of n that can be Python sorted in three days: 1e13
```

Problem 6



Problem			Algorithm			Implementation		
Best	Avg	worst	Best	Avg	worst	Best	Avg	worst
$O(n)$	$O(n)$	$O(n)$	$O(n)$	$\Omega(n)$	$O(n)$	$\Omega(n)$	$\Omega(n)$	$\Omega(n)$
$O(n^2)$	$O(n^2)$	$O(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$\Omega(n^2)$
$\Omega(n^2)$	$O(n^2)$	$O(n^2)$	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(n^2)$	$\Omega(n^2)$	$\Omega(n^2)$

n classes, ideal pct $= \frac{1}{n}$ $\frac{1}{n} = calc = 1$