Ziad Arafat
FEB 10, 2017

CS 111: CS Principles
Homework III

P.139 Q2

a.

1. Set a to 12.
2. Set b to $a + 1$.
3. Set c to $b \times a \div 12$.
4. Print the value of c

b.

78

P.140 Q3

a.

1. Set F to a list [1, 1]
2. Set x to 1 and y to 2
3. Set z to 1
4. While z is less than or equal to 20:
   (a) Append the sum of F[x] and F[y] to the end of list F
   (b) Increment x, y, and z by 1
5. End of loop
6. Print the value of F[20]

b.

6765

c.

I think the the recursive algorithm is more efficient simply because it's easier to implement and because it ends up being less steps than calculating the equation. It is also much simpler to understand and rings more with the fibonacci sequence. However once we approach bigger numbers it becomes clear that the equation is much more efficient because it then involves much less steps so if we were to compute F[100] I would say the equation is better over all.

1

d.

Done in Python

```
F = [1, 1]
x = 0
y = 1
z = 1
while z < 20:
    F.append(F[x]+F[y])
    x += 1
    y += 1
    z += 1
for i in F:
    print(i, end=",")
```

```
Run fibbo
/home/cat/PycharmProjects/CS111H3/venv/bin/python
  /home/cat/PycharmProjects/CS111H3/fibbo.py
1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,
Process finished with exit code 0
```

e.

Here is code using the iterative algorithm

fibbo.py

```
1  F = [1, 1]
2  x = 0
3  y = 1
4  z = 1
5  while z < 50:
6      F.append(F[x]+F[y])
7      x += 1
8      y += 1
9      z += 1
10 # for i in F:
11     # print(i, end=",")
12 print(F[50-1])
13
```

code using the recursive calculation in Python's default math library

fibbocalc.py

```
1  import math
2
3  a = (math.sqrt(5)/5)*((1+math.sqrt(5))/2)**50 \
4      - (math.sqrt(5)/5)*((1-math.sqrt(5))/2)**50
5  print(a)
6
```

P.141 Q7

a.

linear/sequential search takes n comparisons in worst case and 1 comparison in best case. So the average is $\Theta(\frac{n}{2})$. $(32million \div 2) \div 12000 = 1333seconds = 12minutes\ 13seconds$

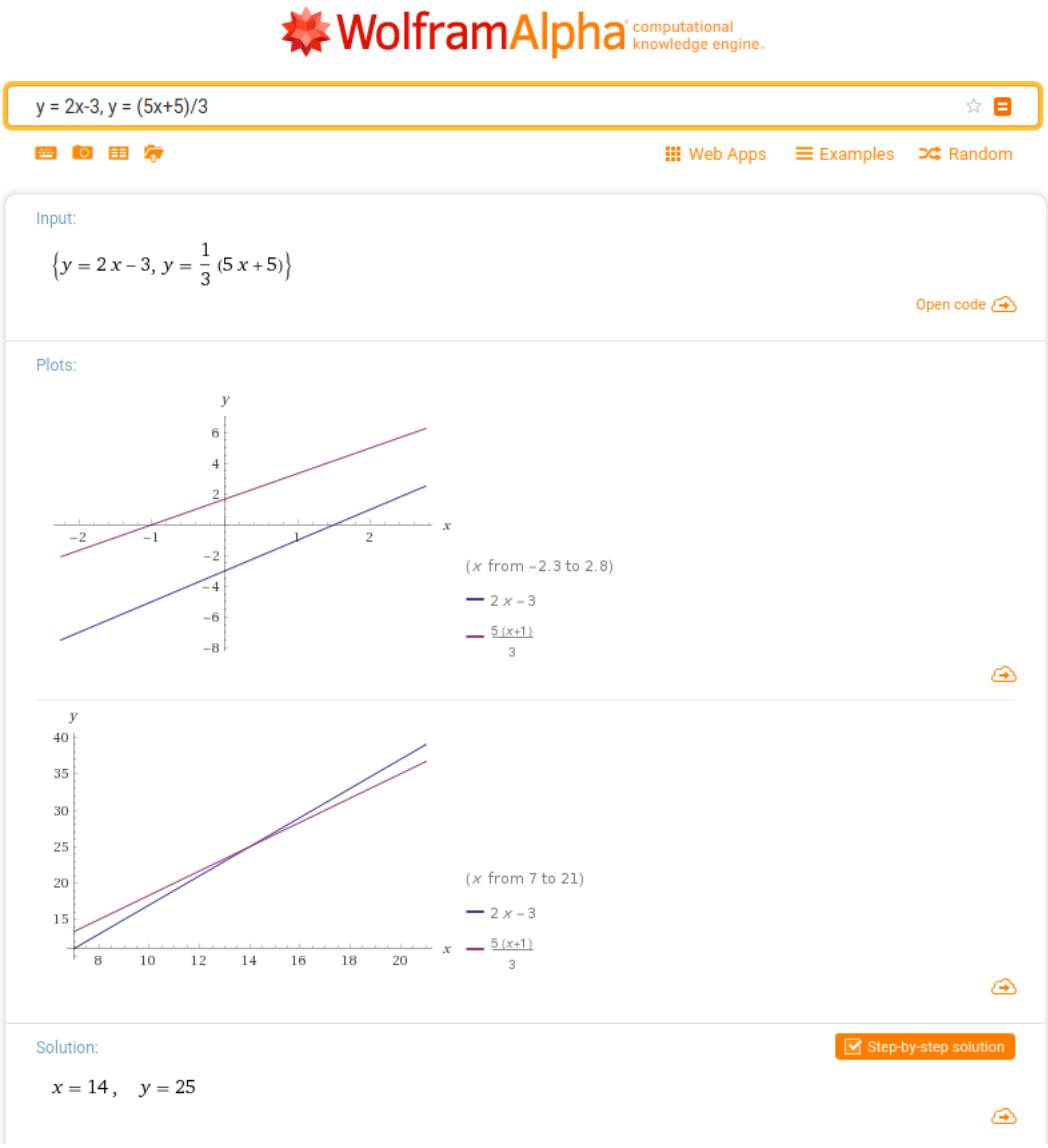
b.

The average case of binary search is $\Theta(\log_2 n)$. So $\log_2 32million \div 12000 \approx 0.002s \approx 2ms$

P.141 Q8

a.

It appears you will need at least or more than 14 pancakes for the algorithm's use to make any sense.



P.141-142 Q11

a.

Bubble Sort

4, 8, 2, 6,
4, 2, 6, 8
2, 4, 6, 8

Selection Sort

4, 8, 2, 6,
2, 8, 4, 6,
2, 4, 8, 6,
2, 4, 6, 8,

b.

Bubble Sort

12, 3, 6, 8, 2, 5, 7,
3, 6, 8, 2, 5, 7, 12,
3, 6, 2, 5, 7, 8, 12,
3, 2, 5, 6, 7, 8, 12,
2, 3, 5, 6, 7, 8, 12,

Selection Sort

12, 3, 6, 8, 2, 5, 7,
2, 3, 6, 8, 12, 5, 7,
2, 3, 5, 8, 12, 6, 7,
2, 3, 5, 6, 12, 8, 7,
2, 3, 5, 6, 7, 8, 12,

c.

Bubble Sort

D, B, G, F, A, C, E, H,
B, D, F, A, C, E, G, H,
B, D, A, C, E, F, G, H,
B, A, C, D, E, F, G, H,
A, B, C, D, E, F, G, H,

Selection Sort

D, B, G, F, A, C, E, H,
A, B, G, F, D, C, E, H,
A, B, C, F, D, G, E, H,
A, B, C, D, F, G, E, H,
A, B, C, D, E, G, F, H,
A, B, C, D, E, F, G, H,

Done in LaTeX