CS21: INTRODUCTION TO COMPUTER SCIENCE

Prof. Mathieson

Fall 2018

Swarthmore College

Outline Nov 19:

- Recap binary search and sorting runtime
- Begin: writing our own classes
- Pie class example
- Student class example

Notes

•Lab 9 due **Monday** after Thanksgiving
•There is lab this week! (Tues/Wed)
•**Ninja session**: Sunday after Thanksgiving 7-10pm

Lab 8 worksheet (binary search)

lst = [bear, bird, bug, cat, cow, dog, fish, lion]

0 1 2 3 4 5 6 7

elephant

query = "bear"

low	mid	high
0	3	7
0	2 (mid-1)	
0	0 (mid-1)	

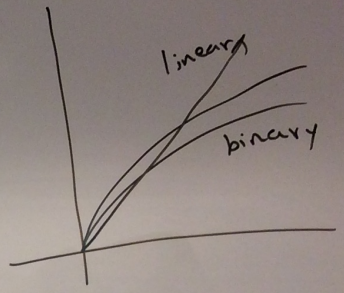
$\frac{0+7}{2} \to 3$

bear < ? cat

bear < ? bird

bear == ? bear

return 0



query = elephant

low	mid	high
0	3	7
(mid+1) 4	5	7
(mid+1) 6	6	7
6 empty	5 (mid-1)	

$\frac{4+7}{2} = 5.5 \to 5$

low > high => STOP

Selection Sort Runtime

$\boxed{O(n^2)}$

Selection Sort Runtime

n steps

n-1

```
for i in range(n):
    m = i    # min index
    for j in range(i+1, n):
        if lst[j] < lst[m]:
            m = j
    swap(i, m, lst)
```

average # steps: $\frac{n}{2}$

i is where min should go

where min is now

i=0	7	3	9	2 ← m	4
i=1	2	3 ← m	9	7	4
i=2	2	3	9	7	4 ← m
i=3	2	3	4	7	9
	2	3	4	7	9

Sorted!

i \ j				
0	1	2	3	4
1		2	3	4
2			3	4
3				4

$\boxed{n-1}$

$\boxed{\frac{n(n-1)}{2}}$ exact runtime

Sorting: didn’t get to in class (good practice!)

- **is_sorted(lst)** function to determine if a list is sorted or not
- **sort_runtime.py** to see runtimes in action

Classes

So far with classes…

- We have already seen existing classes (**Point**, **Circle**, **Line**, etc)

So far with classes…

- We have already seen existing classes (**Point**, **Circle**, **Line**, etc)
- Classes allow us to encapsulate common data structures and actions so we don’t have to define them over and over again

So far with classes…

- We have already seen existing classes (**Point**, **Circle**, **Line**, etc)
- Classes allow us to encapsulate common data structures and actions so we don't have to define them over and over again
- We can create a new instance of a class using the *constructor*

```
dot = Circle(Point(x,y),r)
```

So far with classes…

- We have already seen existing classes (**Point**, **Circle**, **Line**, etc)
- Classes allow us to encapsulate common data structures and actions so we don't have to define them over and over again
- We can create a new instance of a class using the *constructor*

```
dot = Circle(Point(x,y),r)
```

- We can access the instance's data using *methods*

```
r = dot.getRadius()
```

So far with classes…

- We have already seen existing classes (**Point**, **Circle**, **Line**, etc)
- Classes allow us to encapsulate common data structures and actions so we don't have to define them over and over again
- We can create a new instance of a class using the *constructor*

```
dot = Circle(Point(x,y),r)
```

- We can access the instance's data using *methods*

```
r = dot.getRadius()
```

- We can use/modify class instances using *methods*

```
dot.move(dx,dy)
```

Motivation for classes: LOLs

- List-of-lists let us keep track of things that should be "together", but they get cumbersome to modify:

```
>>> pie_lst = [["apple",8], ["cherry",8], ["chocolate",8]]
>>>
>>> pie_lst[2][1] -= 1
>>>
>>> pie_lst
[['apple', 8], ['cherry', 8], ['chocolate', 7]]
```