CSCI 141

CSCI 141

Lecture 25

CSCI 141

Lecture 25
Searching and Sorting

Announcements

Announcements

- Reminder: **No A5 submissions accepted past Thursday 12/5 at 10pm**

Announcements

- Reminder: **No A5 submissions accepted past Thursday 12/5 at 10pm**
- Reminder: Extra TA OH today and tomorrow:

Announcements

- Reminder: **No A5 submissions accepted past Thursday 12/5 at 10pm**
- Reminder: Extra TA OH today and tomorrow:
  - Kirsten: 10-12 Monday and Tuesday (CF 163)

Announcements

- Reminder: **No A5 submissions accepted past Thursday 12/5 at 10pm**
- Reminder: Extra TA OH today and tomorrow:
  - Kirsten: 10-12 Monday and Tuesday (CF 163)
  - Rory 12-2 Monday and Tuesday (CF 477)

Announcements

- Reminder: **No A5 submissions accepted past Thursday 12/5 at 10pm**
- Reminder: Extra TA OH today and tomorrow:
  - Kirsten: 10-12 Monday and Tuesday (CF 163)
  - Rory 12-2 Monday and Tuesday (CF 477)
- Monday 12/2 (a week from today) there will be a review quiz.

Announcements

- Reminder: **No A5 submissions accepted past Thursday 12/5 at 10pm**
- Reminder: Extra TA OH today and tomorrow:
  - Kirsten: 10-12 Monday and Tuesday (CF 163)
  - Rory 12-2 Monday and Tuesday (CF 477)
- Monday 12/2 (a week from today) there will be a review quiz.
  - Delivered via Socrative, but counted towards QOTD credit; worth **2 QOTD**s.

Announcements

- Regular QOTDs are done! Suggestion: spend the time doing a bit of studying every day.
- Sample coding questions for the final exam will be out by next Monday (aiming for tomorrow).

Suppose the file rick.txt contains:

```
Never gonna give you up
```

What is the output of the following code?

```
print(open("rick.txt", "r").read(5).split("e"))
```

A. Nvr
B. Never
C. ["N", "e", "v", "e", "r"]
D. ["N", "v", "r"]

QOTD

```
print("_".join("88r4if4r462".split("r4")))
```

QOTD

```
print("_".join("88r4if4r462".split("r4")))
```

```
"88r4if4r462".split("r4")
```

QOTD

```
print("_".join("88r4if4r462".split("r4")))
```

```
"88r4if4r462".split("r4")
```

```
print("_".join(["88", "if4, "62"]))
```

QOTD

```
print("_".join("88r4if4r462".split("r4")))
```

```
"88r4if4r462".split("r4")
```

```
print("_".join(["88", "if4, "62"]))
```

```
print("88_if4_62")
```

QOTD

```
fin = open("fin.txt", 'r')
fot = open("fot.txt", 'w')

for line in fin:
    fl = line.strip().split(': ')

    fot.write(fl[-1])
    fot.write(fl[0])
```

fin.txt:

```
Start: 4pm
End: 7pm
3h
```

```
fot.close()
fin.close()

fot = open("fot.txt", 'r')
a = fot.read(3)
fot.seek(8)
print(a, fot.read())
fot.close()
```

QOTD

```
fin = open("fin.txt", 'r')
fot = open("fot.txt", 'w')

for line in fin:
    fl = line.strip().split(': ')

    fot.write(fl[-1])
    fot.write(fl[0])

fot.close()
fin.close()

fot = open("fot.txt", 'r')
a = fot.read(3)
fot.seek(8)
print(a, fot.read())
fot.close()
```

fin.txt:

```
Start: 4pm
End: 7pm
3h
```

fout.txt:

```
4pmStart7pmEnd3h3h
```

QOTD

```
fin = open("fin.txt", 'r')
fot = open("fot.txt", 'w')

for line in fin:
    fl = line.strip().split(': ')

    fot.write(fl[-1])
    fot.write(fl[0])

fot.close()
fin.close()

fot = open("fot.txt", 'r')
a = fot.read(3)
fot.seek(8)
print(a, fot.read())
fot.close()
```

fin.txt:

```
Start: 4pm
End: 7pm
3h
```

fout.txt:

```
4pmStart7pmEnd3h3h
```

```
0123456789
```

QOTD

```
fin = open("fin.txt", 'r')
fot = open("fot.txt", 'w')

for line in fin:
    fl = line.strip().split(': ')

    fot.write(fl[-1])
    fot.write(fl[0])

fot.close()
fin.close()

fot = open("fot.txt", 'r')
a = fot.read(3)
fot.seek(8)
print(a, fot.read())
fot.close()
```

fin.txt:

```
Start: 4pm
End: 7pm
3h
```

fout.txt:

```
4pmStart7pmEnd3h3h
```

```
0123456789
```

4pm

QOTD

```
fin = open("fin.txt", 'r')
fot = open("fot.txt", 'w')

for line in fin:
    fl = line.strip().split(': ')

    fot.write(fl[-1])
    fot.write(fl[0])

fot.close()
fin.close()

fot = open("fot.txt", 'r')
a = fot.read(3)
fot.seek(8)
print(a, fot.read())
fot.close()
```

fin.txt:

```
Start: 4pm
End: 7pm
3h
```

fout.txt:

```
4pmStart7pmEnd3h3h
```

```
0123456789
```

4pm

7pmEnd3h3h

QOTD

```
fin = open("fin.txt", 'r')
fot = open("fot.txt", 'w')

for line in fin:
    fl = line.strip().split(': ')

    fot.write(fl[-1])
    fot.write(fl[0])

fot.close()
fin.close()

fot = open("fot.txt", 'r')
a = fot.read(3)
fot.seek(8)
print(a, fot.read())
fot.close()
```

fin.txt:

```
Start: 4pm
End: 7pm
3h
```

fout.txt:

```
4pmStart7pmEnd3h3h
0123456789
```

4pm

7pmEnd3h3h

Output: 4pm 7pmEnd3h3h

Computer Science:
An Analogy

CSCI 141 : spelling and grammar

::

Most of computer science : literature

Computer Science:
An Analogy

CSCI 141 : learning to use a telescope

::

computer science : astronomy

What does the rest of computer science look like?

Many, many "subfields":

- Systems: designing computer systems that do useful stuff.
  Thank these people for your operating system, most of the underpinnings of the internet, etc. (CSCI 247, 347, 447, 367, ...)

- Theory: answering questions about what can be done (efficiently) on a computer.
  Thank these people for the ability to do many of the things you take for granted without (literally) having to wait hundreds of years. (CSCI 301, 305, 405, ...)

- Programming languages: studying, designing, and implementing languages.
  Thank these people for the ability to write Python instead of typing out 1's and 0's. (CSCI 301, 410, 450)

What does the rest of computer science look like?

Many other sub-fields, some of which are more "applied":

- computer vision (497P)
- computer graphics (480)
- human-computer interaction (346)
- machine learning (471)
- robotics (372)
- ...

finding a value in a list

```
def find(v, lst):
    """ Return the index of the first
        occurrence of v in lst.
        Return -1 if v is not in the list.
        Precondition: lst is a list. """
```

Task: write **pseudocode** to solve this problem.

Analysis of Algorithms: Searching a list

What's the most **efficient** way to find a value in a list?

In a list of *N* elements, how many comparisons does our algorithm make:

- in the "best" case?
- in the "worst" case?
- in the "average" case?

finding a value in a *sorted* list

```
def find(v, sorted_lst):
    """ Return the index of the first occurrence
    of v in lst.
    Return -1 if v is not in the list.
    Precondition: lst is a list of things that
    can be compared with the < operator, and is
    in sorted order (i.e. lst[i] <= lst[i+1] for
    all i in range(len(lst)-1) """
```

Task: write **pseudocode** to solve this problem.

Analysis of Algorithms: Searching a sorted list

What's the most **efficient** way to find a value in a list?

In a list of *N* elements, how many comparisons does our algorithm make:

- in the "best" case?
- in the "worst" case?
- in the "average" case?

Sorting a List

Sorting a List

Okay, so searching is fast in a sorted list.

Sorting a List

Okay, so searching is fast in a sorted list.

... how can we sort a list?

Sorting a List

```
def sort(lst):
    """ Sort the given list.
    Precondition: lst is a list of things
    that can be compared with the < operator.
    Postcondition: lst[i] <= lst[i+1] for all
    i in range(len(list)-1).
    """
```

Example list: [8, 4, 7, 9, 2, 1, 6, 4]

Task: write pseudocode to solve this problem.

Analysis of Algorithms: Sorting a list

Analysis of Algorithms: Sort

What's the most **efficient** way to sort a list?

In a list of *N* elements, how many comparisons does your algorithm make:

- in the "best" case?
- in the "worst" case?
- in the "average" case?