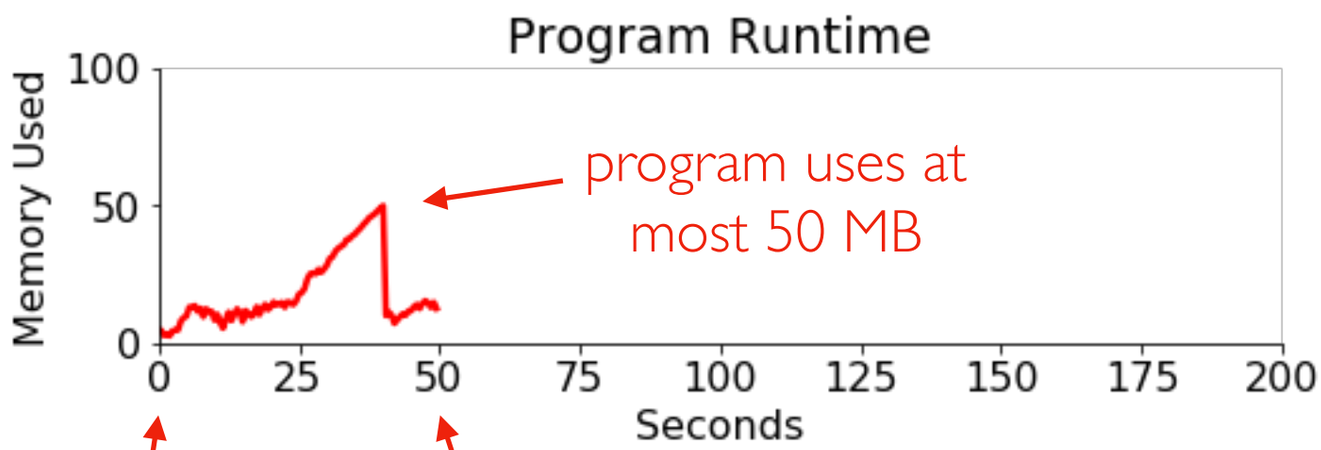
[320] Regular Expressions

Tyler Caraza-Harter

Review Space Complexity



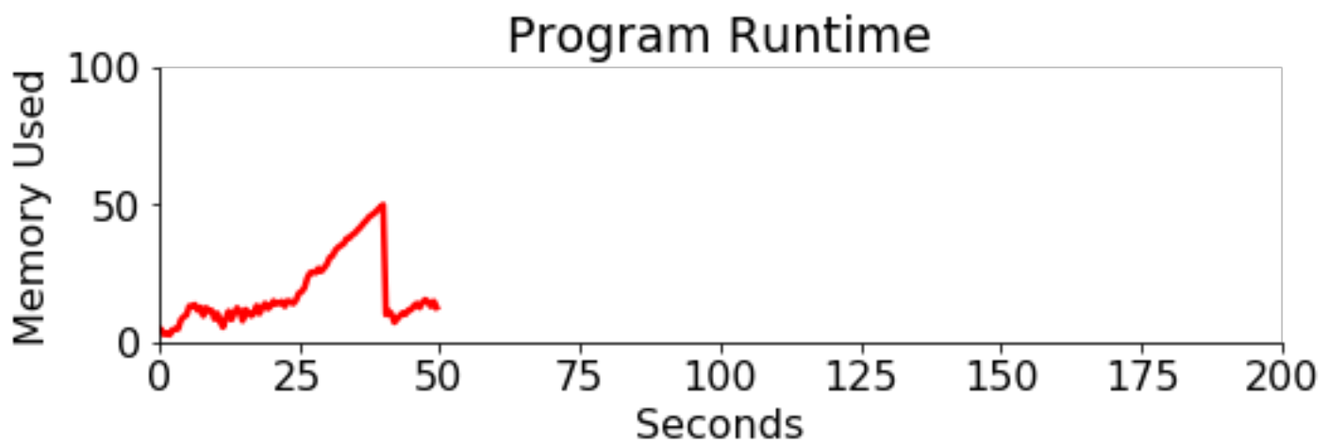
program uses at most 50 MB

assume input size N was large

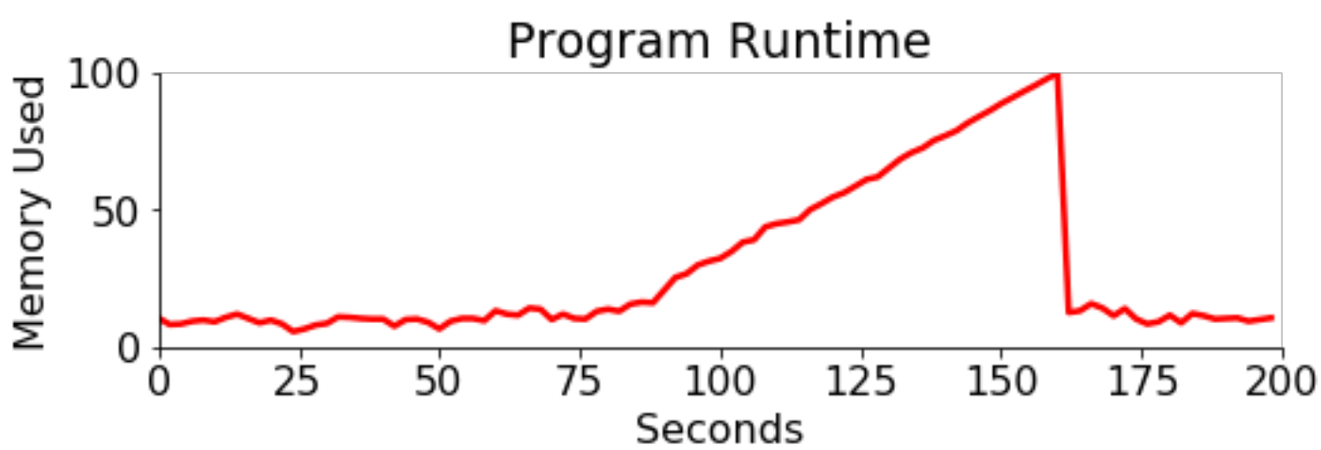
program starts

program ends after 50 seconds

Review Space Complexity

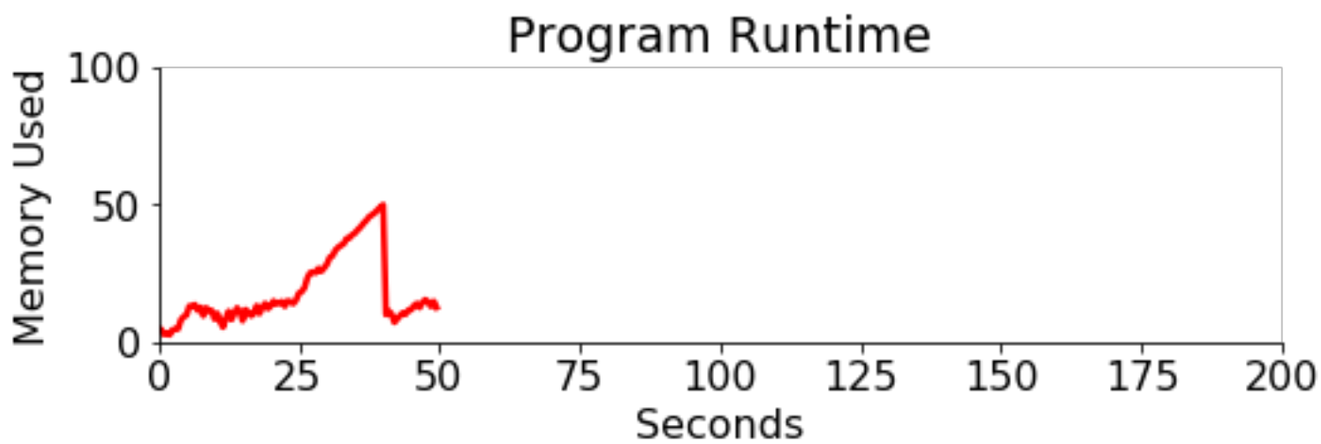


double input size N

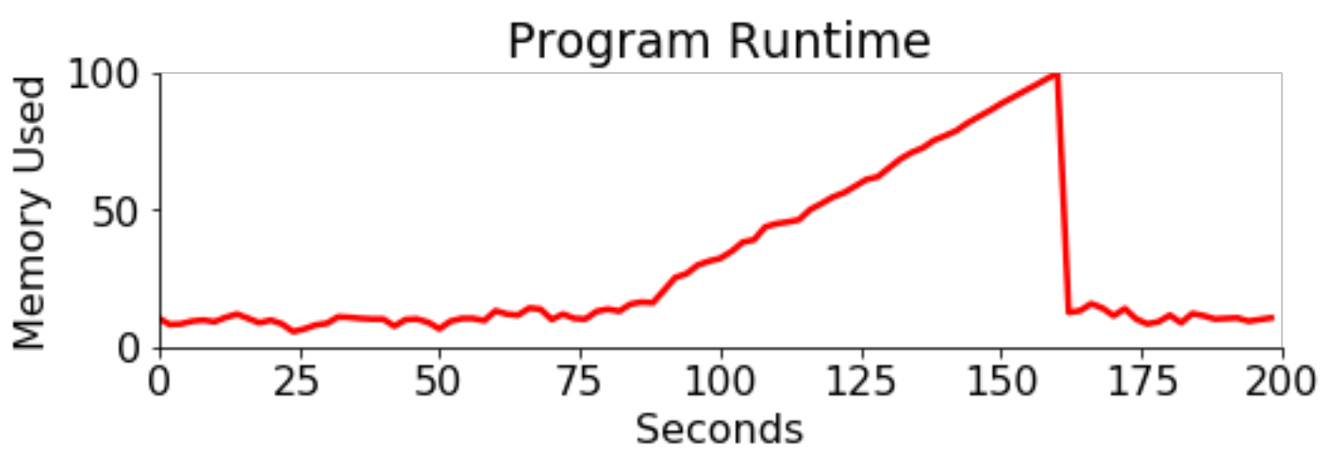


What is the likely order of growth for **time**? For **space**?

Review Space Complexity



double input size N



What is the likely order of growth for **time**? For **space**?

$O(N^2)$ $O(N)$

Reading

New text: *Principles and Techniques of Data Science*
by Sam Lau, Joey Gonzalez, and Deb Nolan

Used for Berkeley's DS100 Course.

Read Chapter 8: https://www.textbook.ds100.org/ch/08/text_intro.html

```
# HIDDEN
def show_regex_match(text, regex):
    """
    Prints the string with the regex match highlighted.
    """
    print(re.sub(f'({regex})', r'\033[1;30;43m\1\033[m', text))
```

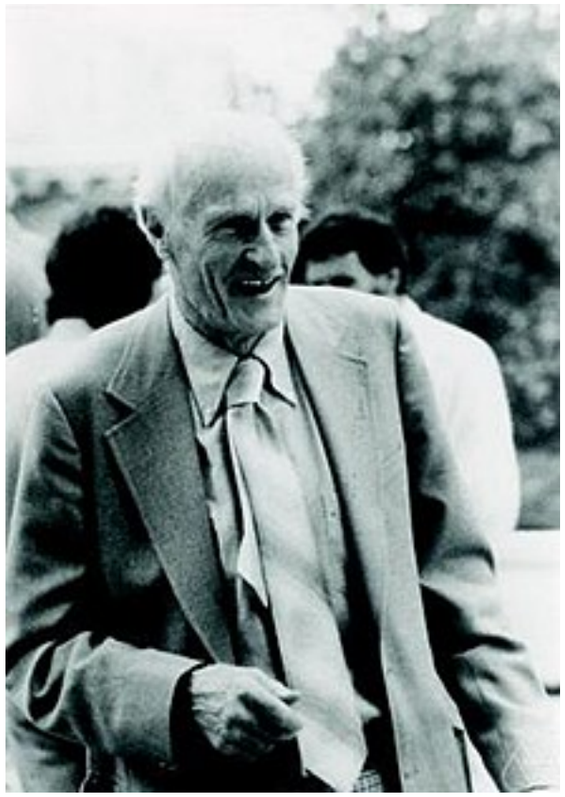
be sure to expand the hidden cells!

```
# The show_regex_match method highlights all regex matches in the
regex = r"green"
show_regex_match("Say! I like green eggs and ham!", regex)
```

```
Say! I like green eggs and ham!
```

Regular Expressions

Regex:
- a small language for describing patterns to search for
- regex patterns are used in many different programming languages (like how many different languages might use SQL queries)
- https://blog.teamtreehouse.com/regular-expressions-10-languages



Stephen Cole Kleene
(mathematician at UW-Madison)

```
msg = "In CS 320, there are 2 exams, 6 projects, 41 lectures,
and 1000 things to learn.  CS 320 is awesome!"

# does the string contain "320"?
has_320 = msg.find("320") >= 0
```

`str.find` is VERY limited -- what if we want to:
- find all occurrences of "320"
- find any 3-digit numbers?
- find any numbers at all?
- find a number before the word "projects"?
- substitute a number for something else?

Regexes can do all these things!

In Python, regular expressions usually use "raw" strings

what character(s) does `print("A\tB")` print between "A" and "B"?

In Python, regular expressions usually use "raw" strings

what character(s) does `print("A\tB")` print between "A" and "B"?

TAB, because backslash is the escape character

what if we actually want a backslash and a "t"?

In Python, regular expressions usually use "raw" strings

what character(s) does `print("A\tB")` print between "A" and "B"?

TAB, because backslash is the escape character

what if we actually want a backslash and a "t"?

`print("A\\tB")`

`print(r"A\tB")`

this is a raw string,
so "\" isn't an escape character

Python regex functions do their own escaping, so this is very handy!

Notebook Demos (copy/paste to start)...

```
import re

# from DS100
def reg(text, regex):
    """
    Prints the string with the regex match highlighted.
    """
    print(re.sub(f'({regex})', r'\033[1;30;43m\1\033[m', text))

s1 = " ".join([
  "A DAG is a directed graph without cycles.",
  "A tree is a DAG where every node has one parent (except the root, which has none).",
  "To learn more, visit www.example.com or call 1-608-123-4567. :)"])

s2 = """1-608-123-4567
a-bcd-efg-hijg (not a phone number)
1-608-123-456 (not a phone number)
608-123-4567
123-4567
"""

s3 = "In CS 320, there are 2 exams, 6 projects, 41 lectures, and 1000 things to learn.
CS 320 is awesome!"
```

Learn Regex Features!

Good overview here: https://www.textbook.ds100.org/ch/08/text_regex.html#Reference-Tables

(screenshots here for convenience)

non-greedy equivalents: `*?` `+?`

Char	Description	Example	Matches	Doesn't Match
.	Any character except \n	...	abc	ab abcd
[]	Any character inside brackets	[cb.]ar	car .ar	jar
[^]	Any character *not* inside brackets	[^b]ar	car par	bar ar
*	≥ 0 or more of last symbol	[pb]*ark	bbark ark	dark
+	≥ 1 or more of last symbol	[pb]+ark	bbpark bark	dark ark
?	0 or 1 of last symbol	s?he	she he	the
{*n*}	Exactly *n* of last symbol	hello{3}	hellooo	hello
|	Pattern before or after bar	we|[ui]s	we us is	e s
\	Escapes next character	\[hi\]	[hi]	hi
^	Beginning of line	^ark	ark two	dark
$	End of line	ark$	noahs ark	noahs arks

Description	Bracket Form	Shorthand
Alphanumeric character	[a-zA-Z0-9]	\w
Not an alphanumeric character	[^a-zA-Z0-9]	\W
Digit	[0-9]	\d
Not a digit	[^0-9]	\D
Whitespace	[\t\n\f\r\p{Z}]	\s
Not whitespace	[^\t\n\f\r\p{z}]	\S

Python **re** Module: **findall** and **sub**

```
import re

s = 'In CS 320, there are 2 exams, 6 projects, 41
lectures, and 1000 things to learn.  CS 320 is
awesome!'
```

```
re.findall(r"\d+", s)
```

pattern input str

```
re.sub(r"\d+", "###", s)
```

pattern replacement input str

Python **re** Module: **findall** and **sub**

```
import re

s = 'In CS 320, there are 2 exams, 6 projects, 41
lectures, and 1000 things to learn.  CS 320 is
awesome!'
```

```
re.findall(r"\d+", s)
```

pattern input str

```
['320', '2', '6',
'41', '1000', '320']
```

```
re.sub(r"\d+", "###", s)
```

pattern replacement input str

```
'In CS ###, there are ### exams, ###
projects, ### lectures, and ### things
to learn.  CS ### is awesome!'
```

Groups

```
import re

s = 'In CS 320, there are 2 exams, 6 projects, 41
lectures, and 1000 things to learn.  CS 320 is
awesome!'
```

```
re.findall(r"(\d+) (\w+)", s)
```

group 1 group 2

Groups

```
import re

s = 'In CS 320, there are 2 exams, 6 projects, 41
lectures, and 1000 things to learn.  CS 320 is
awesome!'
```

```
re.findall(r"(\d+) (\w+)", s)
```

group 1 group 2

```
[('2', 'exams'), ('6', 'projects'), ('41', 'lectures'),
 ('1000', 'things'), ('320', 'is')]
```

Python **re** Module: **findall** and **sub**

```
import re
```

2 spaces tab newline

```
s = """In CS 320,  there are 2 exams,    6 projects,
41 lectures, and 1000 things to learn.  CS 320 is
awesome!"""
```

```
re.findall(r"\d+", s)
```

pattern input str

```
re.sub(r"\d+", "###", s)
```

pattern replacement input str

Python **re** Module: **findall** and **sub**

```
import re
```

2 spaces tab newline

```
s = """In CS 320,  there are 2 exams,    6 projects,
41 lectures, and 1000 things to learn.  CS 320 is
awesome!"""
```

```
re.sub(r"\s+", " ", s)
```

pattern replacement input str

Python **re** Module: **findall** and **sub**

```
import re
```

2 spaces tab newline

```
s = """In CS 320,  there are 2 exams,    6 projects,
41 lectures, and 1000 things to learn.  CS 320 is
awesome!"""
```

```
re.sub(r"\s+", " ", s)
```

pattern replacement input str

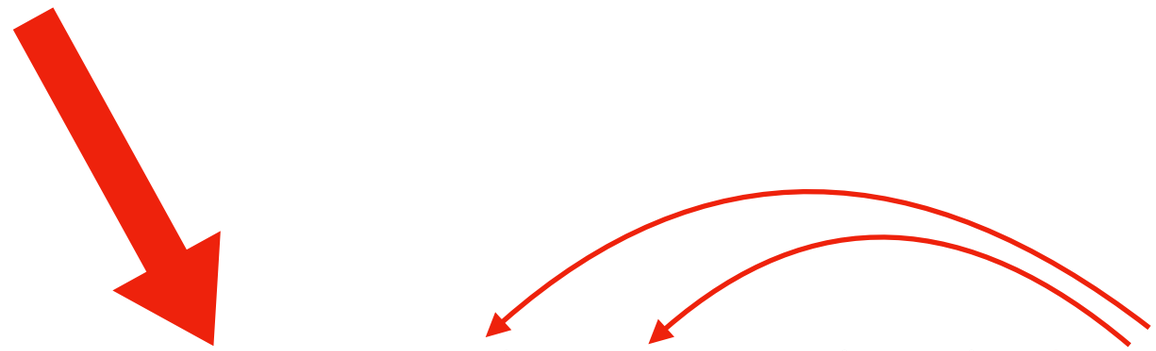
single space is only separator!

```
'In CS 320, there are 2 exams, 6 projects, 41
lectures, and 1000 things to learn. CS 320 is
awesome!'
```

Python **re** Module: **findall** and **sub**

```
import re

s = 'In CS 320, there are 2 exams, 6 projects, 41
lectures, and 1000 things to learn.  CS 320 is
awesome!'
```



```
re.sub(r"(\s+)", "\<b\>\g<1>\</b\>", s)
```

use \g<N> to refer to group N

Python **re** Module: **findall** and **sub**

```
import re

s = 'In CS 320, there are 2 exams, 6 projects, 41
lectures, and 1000 things to learn.  CS 320 is
awesome!'
```

```
re.sub(r"(\d+)", r"<b>\g<1></b>", s)
```

```
In CS <b>320</b>, there are <b>2</b> exams, <b>6</b>
b> projects, <b>41</b> lectures, and <b>1000</b>
things to learn.  CS <b>320</b> is awesome!
```

In CS **320**, there are **2** exams, **6** projects, **41** lectures, and **1000** things to learn. CS **320** is awesome!