CS 170: Algorithms

Account forms after class!

CS 170: Algorithms

Account forms after class!

Static Course Webpage. (inst.cs.berkeley.edu/c̃s170)

CS 170: Algorithms

Account forms after class!

Static Course Webpage. (inst.cs.berkeley.edu/c̃s170)

Will mostly use piazza. Should have/get invitation soon.

CS 170: Algorithms

Account forms after class!

Static Course Webpage. (inst.cs.berkeley.edu/c̃s170)

Will mostly use piazza. Should have/get invitation soon.

Homeworks will be turned in electronically.

CS 170: Algorithms

Account forms after class!

Static Course Webpage. (inst.cs.berkeley.edu/c̃s170)

Will mostly use piazza. Should have/get invitation soon.

Homeworks will be turned in electronically.
Latex ok, formatting ok, scanning handwritten stuff ok.

CS 170: Algorithms

Account forms after class!

Static Course Webpage. (inst.cs.berkeley.edu/c̃s170)

Will mostly use piazza. Should have/get invitation soon.

Homeworks will be turned in electronically.
Latex ok, formatting ok, scanning handwritten stuff ok.
Instructions on piazza.

Instructor.

Prof. Satish Rao

Instructor.

Prof. Satish Rao

Algorithms, graph algorithms, optimization, approximation.

Instructor.

Prof. Satish Rao

Algorithms, graph algorithms, optimization, approximation.

School: ..a long time ago, far far away..

Instructor.

Prof. Satish Rao

Algorithms, graph algorithms, optimization, approximation.

School: ..a long time ago, far far away.... at MIT..

Instructor.

Prof. Satish Rao

Algorithms, graph algorithms, optimization, approximation.

School: ..a long time ago, far far away.... at MIT..

... then NEC Research

Instructor.

Prof. Satish Rao

Algorithms, graph algorithms, optimization, approximation.

School: ..a long time ago, far far away.... at MIT..

... then NEC Research

... stint at Akamai Technologies.

Instructor.

Prof. Satish Rao

Algorithms, graph algorithms, optimization, approximation.

School: ..a long time ago, far far away.... at MIT..

... then NEC Research

... stint at Akamai Technologies.

... Berkeley!

Instructor.

Prof. Satish Rao

Algorithms, graph algorithms, optimization, approximation.

School: ..a long time ago, far far away.... at MIT..

... then NEC Research

... stint at Akamai Technologies.

... Berkeley!

(also Microsoft, Bell Labs, ICAD,

Instructor.

Prof. Satish Rao

Algorithms, graph algorithms, optimization, approximation.

School: ..a long time ago, far far away.... at MIT..

... then NEC Research

... stint at Akamai Technologies.

... Berkeley!

(also Microsoft, Bell Labs, ICAD, ..., paperboy, sold newspapers door to door.)

GSIs

Rudy Dai
Berkeley Undergrad

GSIs

Rudy Dai
Berkeley Undergrad

GSIs

Rudy Dai
Berkeley Undergrad

Lewin Gan

GSIs

Rudy Dai
Berkeley Undergrad

Lewin Gan
Berkeley Undergrad

GSIs

Rudy Dai
Berkeley Undergrad

Lewin Gan
Berkeley Undergrad

Edwin Liao
Berkeley Undergrad

GSIs

Rudy Dai
Berkeley Undergrad

Lewin Gan
Berkeley Undergrad

Edwin Liao
Berkeley Undergrad

Rohin Shah
Berkeley Undergrad

GSIs

Rudy Dai
Berkeley Undergrad

Lewin Gan
Berkeley Undergrad

Edwin Liao
Berkeley Undergrad

Rohin Shah
Berkeley Undergrad

Yun Park
Berkeley Undergrad

GSIs

Rudy Dai
Berkeley Undergrad

Lewin Gan
Berkeley Undergrad

Edwin Liao
Berkeley Undergrad

Rohin Shah
Berkeley Undergrad

Yun Park
Berkeley Undergrad

Piyush Srivistava

GSIs

Rudy Dai
Berkeley Undergrad

Lewin Gan
Berkeley Undergrad

Edwin Liao
Berkeley Undergrad

Rohin Shah
Berkeley Undergrad

Yun Park
Berkeley Undergrad

Piyush Srivistava
PhD student
Theoretical Computer Science

GSIs

Rudy Dai
Berkeley Undergrad

Lewin Gan
Berkeley Undergrad

Edwin Liao
Berkeley Undergrad

Rohin Shah
Berkeley Undergrad

Yun Park
Berkeley Undergrad

Piyush Srivistava
PhD student
Theoretical Computer Science

Di Wang PhD Student

GSIs

Rudy Dai
Berkeley Undergrad

Lewin Gan
Berkeley Undergrad

Edwin Liao
Berkeley Undergrad

Rohin Shah
Berkeley Undergrad

Yun Park
Berkeley Undergrad

Piyush Srivistava
PhD student
Theoretical Computer Science

Di Wang PhD Student
Theoretical Computer Science

Puzzler.

Given a set of strings, is there a pair which has the same letters permuted?

Puzzler.

Given a set of strings, is there a pair which has the same letters permuted?

ATE, CAT, CATEGORY, GORY, ALOOF,

Puzzler.

Given a set of strings, is there a pair which has the same letters permuted?

ATE, CAT, CATEGORY, GORY, ALOOF, EAT, ...?

Puzzler.

Given a set of strings, is there a pair which has the same letters permuted?

ATE, CAT, CATEGORY, GORY, ALOOF, EAT, ...?

Try all permutations of each word, check for duplicates

Puzzler.

Given a set of strings, is there a pair which has the same letters permuted?

ATE, CAT, CATEGORY, GORY, ALOOF, EAT, ...?

Try all permutations of each word, check for duplicates ?

Puzzler.

Given a set of strings, is there a pair which has the same letters permuted?

ATE, CAT, CATEGORY, GORY, ALOOF, EAT, ...?

Try all permutations of each word, check for duplicates ?

Can you do better?

Puzzler.

Given a set of strings, is there a pair which has the same letters permuted?

ATE, CAT, CATEGORY, GORY, ALOOF, EAT, ...?

Try all permutations of each word, check for duplicates ?

Can you do better?

Idea: make a canonical representation of each word.

Puzzler.

Given a set of strings, is there a pair which has the same letters permuted?

ATE, CAT, CATEGORY, GORY, ALOOF, EAT, ...?

Try all permutations of each word, check for duplicates ?

Can you do better?

Idea: make a canonical representation of each word.

Sort letters of each word, check for duplicates.

Puzzler.

Given a set of strings, is there a pair which has the same letters permuted?

ATE, CAT, CATEGORY, GORY, ALOOF, EAT, ...?

Try all permutations of each word, check for duplicates ?

Can you do better?

Idea: make a canonical representation of each word.

Sort letters of each word, check for duplicates.

AET, ACT, ACEGORTY, GORY,ALFOO, AET...

Puzzler.

Given a set of strings, is there a pair which has the same letters permuted?

ATE, CAT, CATEGORY, GORY, ALOOF, EAT, ...?

Try all permutations of each word, check for duplicates ?

Can you do better?

Idea: make a canonical representation of each word.

Sort letters of each word, check for duplicates.

AET, ACT, ACEGORTY, GORY,ALFOO, AET...

```
def find˙permutations(words):
   stash =
   for w in words:
      s = str(sorted(w))
      if s in stash:
         return [w,stash[s]]
      else:
         stash[s] = w
   return False
```

Another problem.

Does a list have a cyle?

Another problem.

Does a list have a cyle?

Mark first node, see if one comes back to it.

Another problem.

Does a list have a cyle?

Mark first node, see if one comes back to it.

Another problem.

Does a list have a cyle?

Mark first node, see if one comes back to it.

Stupid Pet Quiz: Does this work?

Another problem.

Does a list have a cyle?

Mark first node, see if one comes back to it.

Stupid Pet Quiz: Does this work?

a) Yes. b) No.

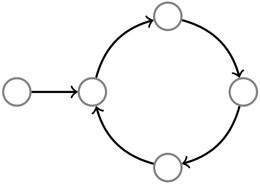
Another problem.

Does a list have a cyle?

Mark first node, see if one comes back to it.

Stupid Pet Quiz: Does this work?

a) Yes. b) No.



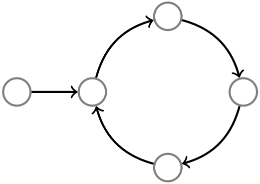
Another problem.

Does a list have a cyle?

Mark first node, see if one comes back to it.

Stupid Pet Quiz: Does this work?

a) Yes. b) No.



First node not in cylce!

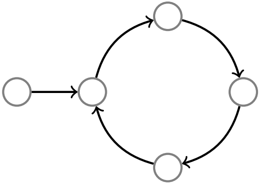
Another problem.

Does a list have a cyle?

Mark first node, see if one comes back to it.

Stupid Pet Quiz: Does this work?

a) Yes. b) No.



First node not in cylce!

Answer is no.

Does a list have a cyle?

Does a list have a cyle?

Two ptrs:

Does a list have a cyle?

Two ptrs: advance ptr 1 twice

Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.

Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.
If ever at the same place, report cycle.

Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.
If ever at the same place, report cycle.

Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.
If ever at the same place, report cycle.

Correctness:

Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.
If ever at the same place, report cycle.

Correctness:
If no cycle, slow pointer never catches fast one.

Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.
If ever at the same place, report cycle.

Correctness:
If no cycle, slow pointer never catches fast one.
If cycle, both pointers will enter cycle at some time.

Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.
If ever at the same place, report cycle.

Correctness:
If no cycle, slow pointer never catches fast one.
If cycle, both pointers will enter cycle at some time.
d - distance from fast ptr to slow ptr.

Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.
If ever at the same place, report cycle.

Correctness:
If no cycle, slow pointer never catches fast one.
If cycle, both pointers will enter cycle at some time.
d - distance from fast ptr to slow ptr.

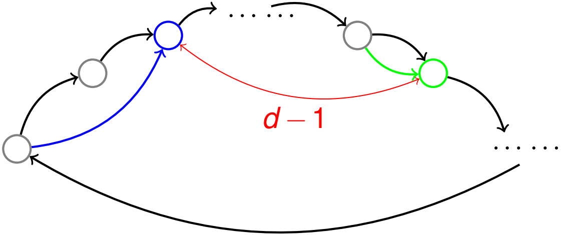
d decreases every step.

Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.
If ever at the same place, report cycle.

Correctness:
If no cycle, slow pointer never catches fast one.
If cycle, both pointers will enter cycle at some time.
d - distance from fast ptr to slow ptr.

d decreases every step.

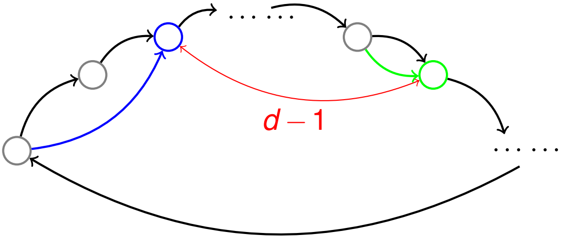


Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.
If ever at the same place, report cycle.

Correctness:
If no cycle, slow pointer never catches fast one.
If cycle, both pointers will enter cycle at some time.
d - distance from fast ptr to slow ptr.

d decreases every step.



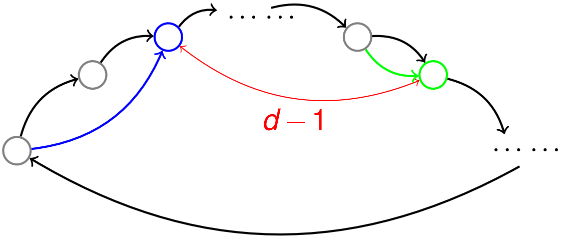
Runtime: n steps to cycle

Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.
If ever at the same place, report cycle.

Correctness:
If no cycle, slow pointer never catches fast one.
If cycle, both pointers will enter cycle at some time.
d - distance from fast ptr to slow ptr.

d decreases every step.



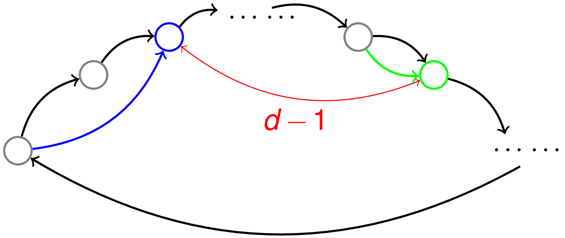
Runtime: *n* steps to cycle *n* steps to catch up.

Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.
If ever at the same place, report cycle.

Correctness:
If no cycle, slow pointer never catches fast one.
If cycle, both pointers will enter cycle at some time.
d - distance from fast ptr to slow ptr.

d decreases every step.



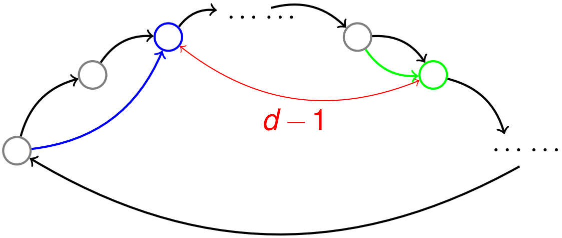
Runtime: n steps to cycle n steps to catch up. $O(n)$

Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.
If ever at the same place, report cycle.

Correctness:
If no cycle, slow pointer never catches fast one.
If cycle, both pointers will enter cycle at some time.
d - distance from fast ptr to slow ptr.

d decreases every step.



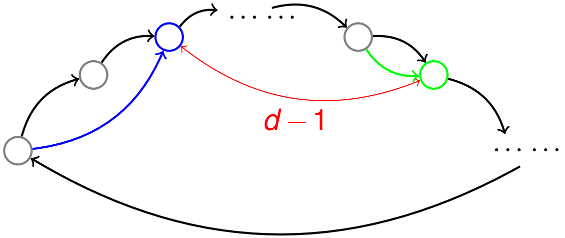
Runtime: n steps to cycle n steps to catch up. $O(n)$

Does a list have a cyle?

Two ptrs: advance ptr 1 twice
advance ptr 2 once.
If ever at the same place, report cycle.

Correctness:
If no cycle, slow pointer never catches fast one.
If cycle, both pointers will enter cycle at some time.
d - distance from fast ptr to slow ptr.

d decreases every step.



Runtime: n steps to cycle n steps to catch up. $O(n)$
Additional storage: two pointers. $O(1)$

Puzzles ..

Solutions

Puzzles ..

Solutions

..are Algorithms...

Puzzles ..

Solutions

..are Algorithms...

which...

Puzzles ..

Solutions

..are Algorithms...

which...

..are correct...

Puzzles ..

Solutions

..are Algorithms...

which...

..are correct...and efficient.

Puzzles ..

Solutions

..are Algorithms...

which...

..are correct...and efficient.

Is this a useful process?

Algorithms for the Human Genome Project

Reconstruct DNA...

Algorithms for the Human Genome Project

Reconstruct DNA...

ACTGAAACTGAGTAGATA....

Algorithms for the Human Genome Project

Reconstruct DNA...

ACTGAAACTGAGTAGATA....

Read first, then next, then next, ...3.1 billion times...
.. slow... error prone...

Algorithms for the Human Genome Project

Reconstruct DNA...

ACTGAAACTGAGTAGATA....

Read first, then next, then next, ...3.1 billion times...
.. slow... error prone...

Parallel sequencing yields chunks of overlapping DNA.

Algorithms for the Human Genome Project

Reconstruct DNA...

ACTGAAACTGAGTAGATA....

Read first, then next, then next, ...3.1 billion times...
.. slow... error prone...

Parallel sequencing yields chunks of overlapping DNA.

AGTAG, AGATA, TGAGT , ACTGAA , CTGAA , AAACTG

Algorithms for the Human Genome Project

Reconstruct DNA...

ACTGAAACTGAGTAGATA....

Read first, then next, then next, ...3.1 billion times...
.. slow... error prone...

Parallel sequencing yields chunks of overlapping DNA.

AGTAG, AGATA, TGAGT , ACTGAA , CTGAA , AAACTG

Assemble into a consistent string?

Algorithms for the Human Genome Project

Reconstruct DNA...

ACTGAAACTGAGTAGATA....

Read first, then next, then next, ...3.1 billion times...
.. slow... error prone...

Parallel sequencing yields chunks of overlapping DNA.

AGTAG, AGATA, TGAGT , ACTGAA , CTGAA , AAACTG

Assemble into a consistent string?

```
ACTGAA
```

Algorithms for the Human Genome Project

Reconstruct DNA...

ACTGAAACTGAGTAGATA....

Read first, then next, then next, ...3.1 billion times...
.. slow... error prone...

Parallel sequencing yields chunks of overlapping DNA.

AGTAG, AGATA, TGAGT , ACTGAA , CTGAA , AAACTG

Assemble into a consistent string?

```
ACTGAA
    AAACTG
```

Algorithms for the Human Genome Project

Reconstruct DNA...

ACTGAAACTGAGTAGATA....

Read first, then next, then next, ...3.1 billion times...
.. slow... error prone...

Parallel sequencing yields chunks of overlapping DNA.

AGTAG, AGATA, TGAGT , ACTGAA , CTGAA , AAACTG

Assemble into a consistent string?

```
ACTGAA
    AAACTG
        TGAGT
```

Algorithms for the Human Genome Project

Reconstruct DNA...

ACTGAAACTGAGTAGATA....

Read first, then next, then next, ...3.1 billion times...
.. slow... error prone...

Parallel sequencing yields chunks of overlapping DNA.

AGTAG, AGATA, TGAGT , ACTGAA , CTGAA , AAACTG

Assemble into a consistent string?

```
ACTGAA
    AAACTG
        TGAGT
          AGTAG
```

Algorithms for the Human Genome Project

Reconstruct DNA...

ACTGAAACTGAGTAGATA....

Read first, then next, then next, ...3.1 billion times...
.. slow... error prone...

Parallel sequencing yields chunks of overlapping DNA.

AGTAG, AGATA, TGAGT , ACTGAA , CTGAA , AAACTG

Assemble into a consistent string?

```
ACTGAA
    AAACTG
        TGAGT
          AGTAG
             AGATA
```

Algorithms for the Human Genome Project

Reconstruct DNA...

ACTGAAACTGAGTAGATA....

Read first, then next, then next, ...3.1 billion times...
.. slow... error prone...

Parallel sequencing yields chunks of overlapping DNA.

AGTAG, AGATA, TGAGT , ACTGAA , CTGAA , AAACTG

Assemble into a consistent string?

```
ACTGAA
    AAACTG
        TGAGT
          AGTAG
             AGATA
-------------------
ACTGAAACTGAGTAGATA
```

Page Rank.

Problem: What is good on the web?

Page Rank.

Problem: What is good on the web?

Website that pays search engine most?

Page Rank.

Problem: What is good on the web?

Website that pays search engine most? Goto.com.

Page Rank.

Problem: What is good on the web?

Website that pays search engine most? Goto.com.

Random Surfer Model (Brin-Page):

Page Rank.

Problem: What is good on the web?

Website that pays search engine most? Goto.com.

Random Surfer Model (Brin-Page): Follow link, follow link,

Page Rank.

Problem: What is good on the web?

Website that pays search engine most? Goto.com.

Random Surfer Model (Brin-Page): Follow link, follow link,
.. occasionally jump to random page (with prob. ε).

Page Rank.

Problem: What is good on the web?

Website that pays search engine most? Goto.com.

Random Surfer Model (Brin-Page): Follow link, follow link,
.. occasionally jump to random page (with prob. ε).

Popular pages are desirable pages.

Page Rank.

Problem: What is good on the web?

Website that pays search engine most? Goto.com.

Random Surfer Model (Brin-Page): Follow link, follow link,
.. occasionally jump to random page (with prob. ε).

Popular pages are desirable pages.

PageRank = popularity for random surfer.

Page Rank.

Problem: What is good on the web?

Website that pays search engine most? Goto.com.

Random Surfer Model (Brin-Page): Follow link, follow link,
.. occasionally jump to random page (with prob. ε).

Popular pages are desirable pages.

PageRank = popularity for random surfer.

Sort search results by PageRank!

Page Rank.

Problem: What is good on the web?

Website that pays search engine most? Goto.com.

Random Surfer Model (Brin-Page): Follow link, follow link,
.. occasionally jump to random page (with prob. ε).

Popular pages are desirable pages.

PageRank = popularity for random surfer.

Sort search results by PageRank!

Made us happy.

Page Rank.

Problem: What is good on the web?

Website that pays search engine most? Goto.com.

Random Surfer Model (Brin-Page): Follow link, follow link,
.. occasionally jump to random page (with prob. ε).

Popular pages are desirable pages.

PageRank = popularity for random surfer.

Sort search results by PageRank!

Made us happy.

Google.

Page Rank.

Problem: What is good on the web?

Website that pays search engine most? Goto.com.

Random Surfer Model (Brin-Page): Follow link, follow link,
.. occasionally jump to random page (with prob. ε).

Popular pages are desirable pages.

PageRank = popularity for random surfer.

Sort search results by PageRank!

Made us happy.

Google. (300 million dollars for license for Stanford!)

Algorithms...

Driving Directions

Algorithms...

Driving Directions
Airline Scheduling

Algorithms...

Driving Directions
Airline Scheduling
Compiling

Algorithms...

Driving Directions
Airline Scheduling
Compiling
Compression

Algorithms...

Driving Directions
Airline Scheduling
Compiling
Compression
Cryptography

Algorithms...

Driving Directions
Airline Scheduling
Compiling
Compression
Cryptography
Optimization

Algorithms...

Driving Directions
Airline Scheduling
Compiling
Compression
Cryptography
Optimization
...(at Akamai, finding fastest, cheapest least loaded caches)

Algorithms...

Driving Directions
Airline Scheduling
Compiling
Compression
Cryptography
Optimization
...(at Akamai, finding fastest, cheapest least loaded caches)
.
.
.

Doing well in this class..

The Bloom 2-sigma effect.

Doing well in this class..

The Bloom 2-sigma effect.

One on one instruction and mastery learning leads to a two sigma improvement in performance.

Doing well in this class..

The Bloom 2-sigma effect.

One on one instruction and mastery learning leads to a two sigma improvement in performance.

50th percentile to the 98th percentile.

Doing well in this class..

The Bloom 2-sigma effect.

One on one instruction and mastery learning leads to a two sigma improvement in performance.

50th percentile to the 98th percentile.

The median Harvard applicant

Doing well in this class..

The Bloom 2-sigma effect.

One on one instruction and mastery learning leads to a two sigma improvement in performance.

50th percentile to the 98th percentile.

The median Harvard applicant would get in!

Doing well in this class..

The Bloom 2-sigma effect.

One on one instruction and mastery learning leads to a two sigma improvement in performance.

50th percentile to the 98th percentile.

The median Harvard applicant would get in!

One on one instruction ....

Doing well in this class..

The Bloom 2-sigma effect.

One on one instruction and mastery learning leads to a two sigma improvement in performance.

50th percentile to the 98th percentile.

The median Harvard applicant would get in!

One on one instruction ....for 300 students

Doing well in this class..

The Bloom 2-sigma effect.

One on one instruction and mastery learning leads to a two sigma improvement in performance.

50th percentile to the 98th percentile.

The median Harvard applicant would get in!

One on one instruction ....for 300 students ???

Doing well in this class..

The Bloom 2-sigma effect.

One on one instruction and mastery learning leads to a two sigma improvement in performance.

50th percentile to the 98th percentile.

The median Harvard applicant would get in!

One on one instruction ....for 300 students ???

Resources: office hours, discussion, piazza,...

Doing well in this class..

The Bloom 2-sigma effect.

One on one instruction and mastery learning leads to a two sigma improvement in performance.

50th percentile to the 98th percentile.

The median Harvard applicant would get in!

One on one instruction ....for 300 students ???

Resources: office hours, discussion, piazza,...

and ..

Doing well in this class..

The Bloom 2-sigma effect.

One on one instruction and mastery learning leads to a two sigma improvement in performance.

50th percentile to the 98th percentile.

The median Harvard applicant would get in!

One on one instruction ....for 300 students ???

Resources: office hours, discussion, piazza,...

and .. you!

Doing well in this class..

The Bloom 2-sigma effect.

One on one instruction and mastery learning leads to a two sigma improvement in performance.

50th percentile to the 98th percentile.

The median Harvard applicant would get in!

One on one instruction ....for 300 students ???

Resources: office hours, discussion, piazza,...

and .. you!

You have the book, homeworks, exams, slides, and the web...

The book.

More e.e. cummings than Danielle Steele.

The book.

More e.e. cummings than Danielle Steele.

Close reading.

The book.

More e.e. cummings than Danielle Steele.

Close reading.

Work through examples.

The book.

More e.e. cummings than Danielle Steele.

Close reading.

Work through examples.

Take care to understand arguments.

Homeworks.

Do them, in groups (up to four.)

Homeworks.

Do them, in groups (up to four.)

Write up solutions, on your own, in your own words.

Homeworks.

Do them, in groups (up to four.)

Write up solutions, on your own, in your own words.

Uh...

Homeworks.

Do them, in groups (up to four.)

Write up solutions, on your own, in your own words.

Uh...helpful for exams.

Homeworks.

Do them, in groups (up to four.)

Write up solutions, on your own, in your own words.

Uh...helpful for exams.

To be clear...

Homeworks.

Do them, in groups (up to four.)

Write up solutions, on your own, in your own words.

Uh...helpful for exams.

To be clear... extremely important for exams

Homeworks.

Do them, in groups (up to four.)

Write up solutions, on your own, in your own words.

Uh...helpful for exams.

To be clear... extremely important for exams ...fun too!!

Quiz.

According to the Bloom effect, with one on one tutoring and mastery learning, a 50th percentile student could reasonably expect to

Quiz.

According to the Bloom effect, with one on one tutoring and mastery learning, a 50th percentile student could reasonably expect to

a) perform at the 50th percentile.

Quiz.

According to the Bloom effect, with one on one tutoring and mastery learning, a 50th percentile student could reasonably expect to

a) perform at the 50th percentile.

b) get into Harvard!

Quiz.

According to the Bloom effect, with one on one tutoring and mastery learning, a 50th percentile student could reasonably expect to

a) perform at the 50th percentile.

b) get into Harvard!

c) be really annoyed by their tutor.

Quiz.

According to the Bloom effect, with one on one tutoring and mastery learning, a 50th percentile student could reasonably expect to

a) perform at the 50th percentile.

b) get into Harvard!

c) be really annoyed by their tutor.

d) perform at the 98th percentile.

Quiz.

According to the Bloom effect, with one on one tutoring and mastery learning, a 50th percentile student could reasonably expect to

a) perform at the 50th percentile.

b) get into Harvard!

c) be really annoyed by their tutor.

d) perform at the 98th percentile.

The best answer is

Quiz.

According to the Bloom effect, with one on one tutoring and mastery learning, a 50th percentile student could reasonably expect to

a) perform at the 50th percentile.

b) get into Harvard!

c) be really annoyed by their tutor.

d) perform at the 98th percentile.

The best answer is ... (d).

Calculating: 300 BC through Middle Ages in Europe.

I – one

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five
X – ten

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five
X – ten
C – one hundred

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five
X – ten
C – one hundred
D – five hundred

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five
X – ten
C – one hundred
D – five hundred
M – a thousand

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five
X – ten
C – one hundred
D – five hundred
M – a thousand

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five
X – ten
C – one hundred
D – five hundred
M – a thousand

VIII – eight

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five
X – ten
C – one hundred
D – five hundred
M – a thousand

VIII – eight

DCLXV – five hundred plus a hundred plus fifty plus ten plus five

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five
X – ten
C – one hundred
D – five hundred
M – a thousand

VIII – eight

DCLXV – five hundred plus a hundred plus fifty plus ten plus five

MCDLXVIII – one thousand five hundred minus one hundred ....

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five
X – ten
C – one hundred
D – five hundred
M – a thousand

VIII – eight

DCLXV – five hundred plus a hundred plus fifty plus ten plus five

MCDLXVIII – one thousand five hundred minus one hundred ....

Add them?

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five
X – ten
C – one hundred
D – five hundred
M – a thousand

VIII – eight

DCLXV – five hundred plus a hundred plus fifty plus ten plus five

MCDLXVIII – one thousand five hundred minus one hundred ....

Add them?

1448 + 665 =

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five
X – ten
C – one hundred
D – five hundred
M – a thousand

VIII – eight

DCLXV – five hundred plus a hundred plus fifty plus ten plus five

MCDLXVIII – one thousand five hundred minus one hundred ....

Add them?

1448 + 665 = 2013

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five
X – ten
C – one hundred
D – five hundred
M – a thousand

VIII – eight

DCLXV – five hundred plus a hundred plus fifty plus ten plus five

MCDLXVIII – one thousand five hundred minus one hundred ....

Add them?

1448 + 665 = 2013

665 years since the printing press.

Calculating: 300 BC through Middle Ages in Europe.

I – one
V – five
X – ten
C – one hundred
D – five hundred
M – a thousand

VIII – eight

DCLXV – five hundred plus a hundred plus fifty plus ten plus five

MCDLXVIII – one thousand five hundred minus one hundred ....

Add them?

1448 + 665 = 2013

665 years since the printing press.

Multiply roman numbers?

Modern system.

From India, via Al Khwarizmi.

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Note:

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Note:

Mayans (base 20):

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Note:

Mayans (base 20): dots (ones) and underlines (fives).

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Note:

Mayans (base 20): dots (ones) and underlines (fives).

13 is “$\underline{\underline{\cdot\cdot\cdot}}$”

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Note:

Mayans (base 20): dots (ones) and underlines (fives).

13 is “$\underline{\underline{\cdot\cdot\cdot}}$”

Babylonions (base 60):

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Note:

Mayans (base 20): dots (ones) and underlines (fives).
 13 is “$\underline{\underline{\cdot\cdot\cdot}}$”

Babylonions (base 60): clusters of 10 instead of digits.

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Note:

Mayans (base 20): dots (ones) and underlines (fives).

13 is “$\underline{\underline{\cdot\cdot\cdot}}$”

Babylonions (base 60): clusters of 10 instead of digits.

Abacus

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Note:

Mayans (base 20): dots (ones) and underlines (fives).
13 is “$\underline{\underline{\cdots}}$”

Babylonions (base 60): clusters of 10 instead of digits.

Abacus successive rows,

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Note:

Mayans (base 20): dots (ones) and underlines (fives).
13 is "$\underline{\underline{\cdot\cdot\cdot}}$"

Babylonions (base 60): clusters of 10 instead of digits.

Abacus successive rows, successive places..

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Note:

Mayans (base 20): dots (ones) and underlines (fives).
    13 is “$\underline{\underline{\cdot\cdot\cdot}}$”

Babylonions (base 60): clusters of 10 instead of digits.

Abacus successive rows, successive places..

India: “invented” 0!

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Note:

Mayans (base 20): dots (ones) and underlines (fives).
    13 is “$\underline{\underline{\cdots}}$”

Babylonions (base 60): clusters of 10 instead of digits.

Abacus successive rows, successive places..

India: “invented” 0! ... and decimal symbols.

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Note:

Mayans (base 20): dots (ones) and underlines (fives).
13 is “$\underline{\underline{\cdots}}$”

Babylonions (base 60): clusters of 10 instead of digits.

Abacus successive rows, successive places..

India: “invented” 0! ... and decimal symbols.

20th century. Base 2!

Modern system.

From India, via Al Khwarizmi.

He also described recipes for adding, multiplying, computing digits of π..

Algorithms!

Note:

Mayans (base 20): dots (ones) and underlines (fives).

13 is "$\underline{\underline{\cdot\cdot\cdot}}$"

Babylonions (base 60): clusters of 10 instead of digits.

Abacus successive rows, successive places..

India: "invented" 0! ... and decimal symbols.

20th century. Base 2!

The input representation for modern computers and communication.

Writing to propagating..

Al Khwarizmi:

Writing to propagating..

Al Khwarizmi: Go west! Young decimal system!

Writing to propagating..

Al Khwarizmi: Go west! Young decimal system!

..but Fibonacci popularized its use.

Writing to propagating..

Al Khwarizmi: Go west! Young decimal system!

..but Fibonacci popularized its use.

So some homage...

Fibonacci numbers.

$F_0 = 0$, $F_1 = 1$.

Fibonacci numbers.

$F_0 = 0$, $F_1 = 1$.

$F_n = F_{n-1} + F_{n-2}$.

Fibonacci numbers.

$F_0 = 0$, $F_1 = 1$.

$F_n = F_{n-1} + F_{n-2}$.

```
def fib(n):
  if n <= 1:
     return n
  else:
     return fib(n-1) + fib(n-2)
```

Fibonacci numbers.

$F_0 = 0$, $F_1 = 1$.

$F_n = F_{n-1} + F_{n-2}$.

```
def fib(n):
  if n <= 1:
     return n
  else:
     return fib(n-1) + fib(n-2)
```

Correct?

Fibonacci numbers.

$F_0 = 0$, $F_1 = 1$.

$F_n = F_{n-1} + F_{n-2}$.

```
def fib(n):
  if n <= 1:
     return n
  else:
     return fib(n-1) + fib(n-2)
```

Correct? Implements definition!

Fibonacci numbers.

$F_0 = 0$, $F_1 = 1$.

$F_n = F_{n-1} + F_{n-2}$.

```
def fib(n):
  if n <= 1:
     return n
  else:
     return fib(n-1) + fib(n-2)
```

Correct? Implements definition!

Run time.

Fibonacci numbers.

$F_0 = 0$, $F_1 = 1$.

$F_n = F_{n-1} + F_{n-2}$.

```
def fib(n):
  if n <= 1:
     return n
  else:
     return fib(n-1) + fib(n-2)
```

Correct? Implements definition!

Run time.

$T(n) = T(n-1) + T(n-2) + 2$

Fibonacci numbers.

$F_0 = 0$, $F_1 = 1$.

$F_n = F_{n-1} + F_{n-2}$.

```
def fib(n):
  if n <= 1:
     return n
  else:
     return fib(n-1) + fib(n-2)
```

Correct? Implements definition!

Run time.

$T(n) = T(n-1) + T(n-2) + 2$

$T(n) \geq F_n$

Fibonacci algorithm and numbers!

$$F_n = F_{n-1} + F_{n-2}$$

Fibonacci algorithm and numbers!

$$F_n = F_{n-1} + F_{n-2} = F_{n-2} + F_{n-3} + F_{n-2}$$

Fibonacci algorithm and numbers!

$$F_n = F_{n-1} + F_{n-2} = F_{n-2} + F_{n-3} + F_{n-2} \geq 2F_{n-2}$$

Fibonacci algorithm and numbers!

$$F_n = F_{n-1} + F_{n-2} = F_{n-2} + F_{n-3} + F_{n-2} \geq 2F_{n-2}$$

By induction, we get $F_n \geq 2^{n/2}$.

Fibonacci algorithm and numbers!

$$F_n = F_{n-1} + F_{n-2} = F_{n-2} + F_{n-3} + F_{n-2} \geq 2F_{n-2}$$

By induction, we get $F_n \geq 2^{n/2}$.

From book.. $F_n \approx 2^{0.694n}$.

Fibonacci algorithm and numbers!

$$F_n = F_{n-1} + F_{n-2} = F_{n-2} + F_{n-3} + F_{n-2} \geq 2F_{n-2}$$

By induction, we get $F_n \geq 2^{n/2}$.

From book.. $F_n \approx 2^{0.694n}$.

$T(n) \geq 2^{n/2}$

Fibonacci algorithm and numbers!

$$F_n = F_{n-1} + F_{n-2} = F_{n-2} + F_{n-3} + F_{n-2} \geq 2F_{n-2}$$

By induction, we get $F_n \geq 2^{n/2}$.

From book.. $F_n \approx 2^{0.694n}$.

$T(n) \geq 2^{n/2}$

From book $T(n) \geq 2^{0.694n}$

Fibonacci algorithm and numbers!

$F_n = F_{n-1} + F_{n-2} = F_{n-2} + F_{n-3} + F_{n-2} \geq 2F_{n-2}$

By induction, we get $F_n \geq 2^{n/2}$.

From book.. $F_n \approx 2^{0.694n}$.

$T(n) \geq 2^{n/2}$

From book $T(n) \geq 2^{0.694n}$

For $n = 100$, this is around 2^{64} operations, (much more than thousand years or so on a fast computer.)

Fibonacci algorithm and numbers!

$$F_n = F_{n-1} + F_{n-2} = F_{n-2} + F_{n-3} + F_{n-2} \geq 2F_{n-2}$$

By induction, we get $F_n \geq 2^{n/2}$.

From book.. $F_n \approx 2^{0.694n}$.

$T(n) \geq 2^{n/2}$

From book $T(n) \geq 2^{0.694n}$

For $n = 100$, this is around 2^{64} operations, (much more than thousand years or so on a fast computer.)

Exponential algorithm. Bad.

Fibonacci algorithm and numbers!

$F_n = F_{n-1} + F_{n-2} = F_{n-2} + F_{n-3} + F_{n-2} \geq 2F_{n-2}$

By induction, we get $F_n \geq 2^{n/2}$.

From book.. $F_n \approx 2^{0.694n}$.

$T(n) \geq 2^{n/2}$

From book $T(n) \geq 2^{0.694n}$

For $n = 100$, this is around 2^{64} operations, (much more than thousand years or so on a fast computer.)

Exponential algorithm. Bad. Grows very fast.

Fibonacci algorithm and numbers!

$F_n = F_{n-1} + F_{n-2} = F_{n-2} + F_{n-3} + F_{n-2} \geq 2F_{n-2}$

By induction, we get $F_n \geq 2^{n/2}$.

From book.. $F_n \approx 2^{0.694n}$.

$T(n) \geq 2^{n/2}$

From book $T(n) \geq 2^{0.694n}$

For $n = 100$, this is around 2^{64} operations, (much more than thousand years or so on a fast computer.)

Exponential algorithm. Bad. Grows very fast.

Can we do better?

Sure..much better.

Sure..much better.

```
def fib(n):
  if n <= 1:
     return n
  else:
    a = [0,1]
    for i in xrange(2,n+1):
       a.append(a[i-1]+a[i-2])
    return a[n]
```

Sure..much better.

```
def fib(n):
  if n <= 1:
     return n
  else:
    a = [0,1]
    for i in xrange(2,n+1):    O(n)
       a.append(a[i-1]+a[i-2])
    return a[n]
```

$O(n)$ operations!

Sure..much better.

```
def fib(n):
  if n <= 1:
     return n
  else:
    a = [0,1]
    for i in xrange(2,n+1):    O(n)
       a.append(a[i-1]+a[i-2])
    return a[n]
```

$O(n)$ operations! Maybe.

Sure..much better.

```
def fib(n):
  if n <= 1:
     return n
  else:
    a = [0,1]
    for i in xrange(2,n+1):     O(n)
       a.append(a[i-1]+a[i-2])
    return a[n]
```

$O(n)$ operations! Maybe.

Let’s try it.

From demo: Size matters.

How many bits in F_n?

From demo: Size matters.

How many bits in F_n?

Remember $F_n \approx 2^{0.694n}$.

From demo: Size matters.

How many bits in F_n?

Remember $F_n \approx 2^{0.694n}$.

About how many bits in F_n?

From demo: Size matters.

How many bits in F_n?

Remember $F_n \approx 2^{0.694n}$.

About how many bits in F_n?

$\log_2 F_n$

From demo: Size matters.

How many bits in F_n?

Remember $F_n \approx 2^{0.694n}$.

About how many bits in F_n?

$\log_2 F_n \approx 0.6294n$

From demo: Size matters.

How many bits in F_n?

Remember $F_n \approx 2^{0.694n}$.

About how many bits in F_n?

$\log_2 F_n \approx 0.6294n$

How long does it take to compute $F_{n-1} + F_{n-2}$?

From demo: Size matters.

How many bits in F_n?

Remember $F_n \approx 2^{0.694n}$.

About how many bits in F_n?

$\log_2 F_n \approx 0.6294n$

How long does it take to compute $F_{n-1} + F_{n-2}$?

$O(n)$.

From demo: Size matters.

How many bits in F_n?

Remember $F_n \approx 2^{0.694n}$.

About how many bits in F_n?

$\log_2 F_n \approx 0.6294n$

How long does it take to compute $F_{n-1} + F_{n-2}$?

$O(n)$.

How long does Fib take?

From demo: Size matters.

How many bits in F_n?

Remember $F_n \approx 2^{0.694n}$.

About how many bits in F_n?

$\log_2 F_n \approx 0.6294n$

How long does it take to compute $F_{n-1} + F_{n-2}$?

$O(n)$.

How long does Fib take?

n additions.

From demo: Size matters.

How many bits in F_n?

Remember $F_n \approx 2^{0.694n}$.

About how many bits in F_n?

$\log_2 F_n \approx 0.6294n$

How long does it take to compute $F_{n-1} + F_{n-2}$?

$O(n)$.

How long does Fib take?

n additions.

At most $O(n^2)$.

Demo: polynomial.

Doubling size, made time grow by factor of four.

Demo: polynomial.

Doubling size, made time grow by factor of four.

cn^2 runtime.

Demo: polynomial.

Doubling size, made time grow by factor of four.

cn^2 runtime.

$c(2n)^2$

Demo: polynomial.

Doubling size, made time grow by factor of four.

cn^2 runtime.

$c(2n)^2 = 4cn^2$.

Demo: polynomial.

Doubling size, made time grow by factor of four.

cn^2 runtime.

$c(2n)^2 = 4cn^2$.

Asymptotic bounds give good prediction of scaling behavior.

Demo: polynomial.

Doubling size, made time grow by factor of four.

cn^2 runtime.

$c(2n)^2 = 4cn^2$.

Asymptotic bounds give good prediction of scaling behavior.

Doubling size, scales runtime by a constant for any polynomial time algorithm.

Demo: polynomial.

Doubling size, made time grow by factor of four.

cn^2 runtime.

$c(2n)^2 = 4cn^2$.

Asymptotic bounds give good prediction of scaling behavior.

Doubling size, scales runtime by a constant for any polynomial time algorithm.

Not true for exponential algorithms.

Demo: polynomial.

Doubling size, made time grow by factor of four.

cn^2 runtime.

$c(2n)^2 = 4cn^2$.

Asymptotic bounds give good prediction of scaling behavior.

Doubling size, scales runtime by a constant for any polynomial time algorithm.

Not true for exponential algorithms. Could square runtime!

Asymptotic Analysis.

Used $O(n)$ for number of additions, rather than $n-2$.

Asymptotic Analysis.

Used $O(n)$ for number of additions, rather than $n-2$.

Why?

Asymptotic Analysis.

Used $O(n)$ for number of additions, rather than $n-2$.

Why?

61a, 61b..

Asymptotic Analysis.

Used $O(n)$ for number of additions, rather than $n-2$.

Why?

61a, 61b..

Recursive fib has faster inner loop than iterative fib.

Asymptotic Analysis.

Used $O(n)$ for number of additions, rather than $n-2$.

Why?

61a, 61b..

Recursive fib has faster inner loop than iterative fib.

Does it matter?

Asymptotic Analysis.

Used $O(n)$ for number of additions, rather than $n-2$.

Why?

61a, 61b..

Recursive fib has faster inner loop than iterative fib.

Does it matter?

$2^{.694n}$ versus $O(n^2)$.

Asymptotic Analysis.

Used $O(n)$ for number of additions, rather than $n-2$.

Why?

61a, 61b..

Recursive fib has faster inner loop than iterative fib.

Does it matter?

$2^{.694n}$ versus $O(n^2)$.

For $2^{.694n}$, doubling n, squares run time.

Asymptotic Analysis.

Used $O(n)$ for number of additions, rather than $n-2$.

Why?

61a, 61b..

Recursive fib has faster inner loop than iterative fib.

Does it matter?

$2^{.694n}$ versus $O(n^2)$.

For $2^{.694n}$, doubling n, squares run time.

For $O(n^2)$, doubling n, multiplies run time by four.

Asymptotic Notation.

Ignore constant factors.

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$

both are $O(n^2)$

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$

both are $O(n^2)$

$4\log n$ asymptotically same as $100\log n$

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$
both are $O(n^2)$

$4\log n$ asymptotically same as $100\log n$
both are $O(\log n)$

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$
both are $O(n^2)$

$4\log n$ asymptotically same as $100\log n$
both are $O(\log n)$

Ignore smaller order terms.

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$
both are $O(n^2)$

$4 \log n$ asymptotically same as $100 \log n$
both are $O(\log n)$

Ignore smaller order terms.

$2n^2 + 100$

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$
both are $O(n^2)$

$4\log n$ asymptotically same as $100\log n$
both are $O(\log n)$

Ignore smaller order terms.

$2n^2 + 100$ is $O(n^2)$

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$
both are $O(n^2)$

$4\log n$ asymptotically same as $100\log n$
both are $O(\log n)$

Ignore smaller order terms.

$2n^2 + 100$ is $O(n^2)$
$2n^2 + 1000\log n$

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$
both are $O(n^2)$

$4 \log n$ asymptotically same as $100 \log n$
both are $O(\log n)$

Ignore smaller order terms.

$2n^2 + 100$ is $O(n^2)$
$2n^2 + 1000 \log n$ is $O(n^2)$

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$
both are $O(n^2)$

$4\log n$ asymptotically same as $100\log n$
both are $O(\log n)$

Ignore smaller order terms.

$2n^2 + 100$ is $O(n^2)$
$2n^2 + 1000\log n$ is $O(n^2)$

Upper bound.
n^2

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$
both are $O(n^2)$

$4 \log n$ asymptotically same as $100 \log n$
both are $O(\log n)$

Ignore smaller order terms.

$2n^2 + 100$ is $O(n^2)$
$2n^2 + 1000 \log n$ is $O(n^2)$

Upper bound.
n^2 is $O(n^3)$.

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$
both are $O(n^2)$

$4\log n$ asymptotically same as $100\log n$
both are $O(\log n)$

Ignore smaller order terms.

$2n^2 + 100$ is $O(n^2)$
$2n^2 + 1000\log n$ is $O(n^2)$

Upper bound.
n^2 is $O(n^3)$.
$\log n$

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$
both are $O(n^2)$

$4 \log n$ asymptotically same as $100 \log n$
both are $O(\log n)$

Ignore smaller order terms.

$2n^2 + 100$ is $O(n^2)$
$2n^2 + 1000 \log n$ is $O(n^2)$

Upper bound.
n^2 is $O(n^3)$.
$\log n$ is $O(n)$.

Asymptotic Notation.

Ignore constant factors.

$2n^2$ asymptotically same as $4n^2$
both are $O(n^2)$

$4 \log n$ asymptotically same as $100 \log n$
both are $O(\log n)$

Ignore smaller order terms.

$2n^2 + 100$ is $O(n^2)$
$2n^2 + 1000 \log n$ is $O(n^2)$

Upper bound.
n^2 is $O(n^3)$.
$\log n$ is $O(n)$.

Formally, for positive functions g, f from integers to reals,
$g(n) = O(f(n)$, if there is a constant c where $g(n) \leq cf(n)$

More asymptotic notation.

Ω notation.

More asymptotic notation.

Ω notation.

A “lower bound”.

More asymptotic notation.

Ω notation.

A "lower bound".

$2n^2$ is $\Omega(n^2)$ ...

More asymptotic notation.

Ω notation.

A “lower bound”.

$2n^2$ is $\Omega(n^2)$ ...and $\Omega(n)$...

More asymptotic notation.

Ω notation.

A "lower bound".

$2n^2$ is $\Omega(n^2)$ ...and $\Omega(n)$...

More asymptotic notation.

Ω notation.

A "lower bound".

$2n^2$ is $\Omega(n^2)$ ...and $\Omega(n)$...

Formally, for positive functions g, f from integers to reals,
$g(n) = \Omega(f(n))$, if there is a constant c where $g(n) \geq cf(n)$

More asymptotic notation.

Ω notation.

A "lower bound".

$2n^2$ is $\Omega(n^2)$ ...and $\Omega(n)$...

Formally, for positive functions g, f from integers to reals,
$g(n) = \Omega(f(n))$, if there is a constant c where $g(n) \geq cf(n)$

$g(n) = \Theta(f(n))$ if $g(n) = O(f(n))$ and $g(n) = \Omega(f(n))$.

...see you on Wednesday.