CSCI 136: Data Structures and Advanced Programming

Lecture 32

Graph and course wrap-up

Instructors: Dan & Bill J

Williams

Announcements

One last week for quiz/activity/feedback

Submit all "soft" labs by May 19 (end of reading period)

Midterm resubmission: also due May 19

Final exam: May 20-25

Evaluation Forms

(all of these are anonymous)

We care a lot about what you say in these forms.
Please take your time and write thoughtful responses.

Your feedback is very valuable to us!

Purpose of Blue Sheets

> Student comments on the blue sheets […] are solely for your benefit. They are not made available to department or program chairs, the Dean of the Faculty, or the CAP for evaluation purposes.
>
> —Office of the Provost, Williams College

Purpose of SCS Forms

> "[T]he SCS provides instructors with feedback regarding their courses and teaching. The faculty legislation governing the SCS provides that SCS results are made available to the appropriate department chair, the Dean of the Faculty, and at appropriate times, to members of the Committee on Appointments and Promotions (CAP). The results are considered in matters of faculty reappointment, tenure, and promotion."
>
> —Office of the Provost, Williams College

<u>Blue sheet prompts:</u>

* What course topic did you enjoy the most?

* What course topic did you least enjoy? Do you think that it was valuable to learn anyway?

* Are there other aspects of the course that you liked or disliked? (E.g., *office hours*, *TAs*, *assignments*, *course structure*, *meeting times*, etc.) Feel free to suggest alternatives.

* Did you look forward to coming to class?

Outline

Graph applications:

- shortest paths
- traveling salesperson

Semester recap

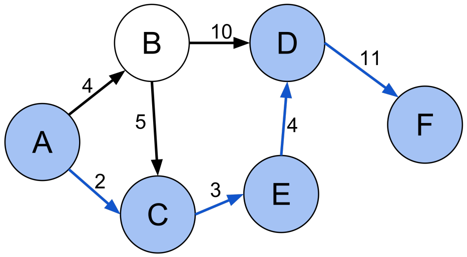
Notes about final exam

Next steps

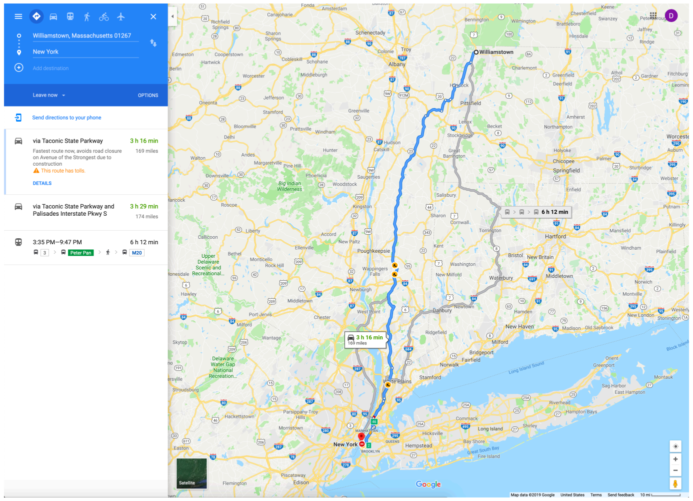
Graphs: shortest paths

Shortest path problem

The **shortest path problem** is the problem of finding a **path between two vertices** in a graph such that **the sum** of the weights of its constituent edges **is minimized**.



Applications


Williamstown, Massachusetts 01267

New York

Add destination

Leave now

OPTIONS

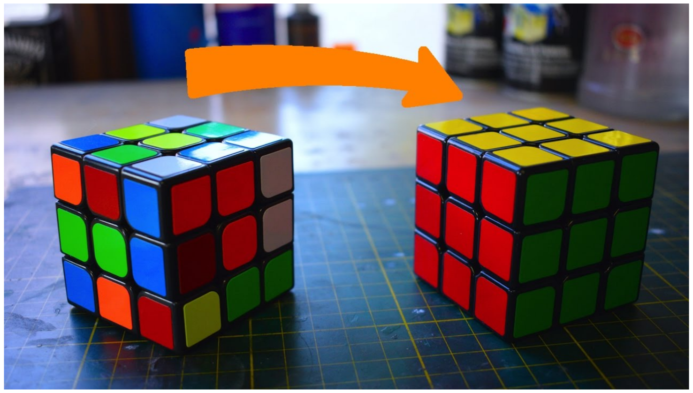
Send directions to your phone

via Taconic State Parkway 3 h 16 min
Fastest route now, avoids road closure on Avenue of the Strongest due to construction 169 miles
This route has tolls.
DETAILS

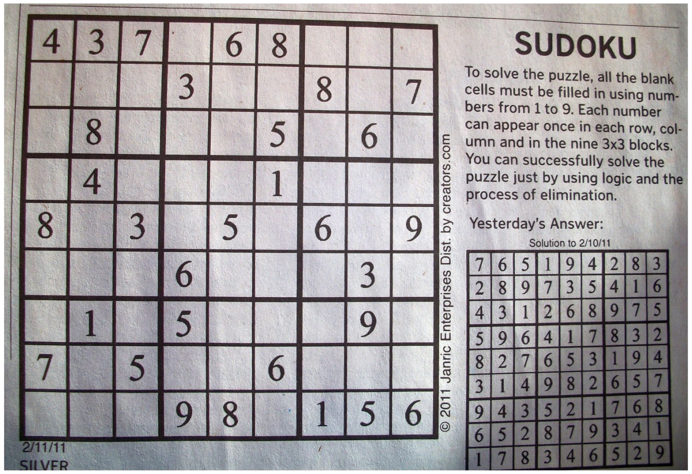
via Taconic State Parkway and Palisades Interstate Pkwy S 3 h 29 min
174 miles

3:35 PM—9:47 PM 6 h 12 min
3 > Peter Pan > > M20

Applications



Applications



SUDOKU

To solve the puzzle, all the blank cells must be filled in using numbers from 1 to 9. Each number can appear once in each row, column and in the nine 3x3 blocks. You can successfully solve the puzzle just by using logic and the process of elimination.

4	3	7		6	8			
			3			8		7
	8				5		6	
	4				1			
8		3		5		6		9
			6				3	
	1		5				9	
7		5			6			
			9	8		1	5	6

2/11/11
SILVER

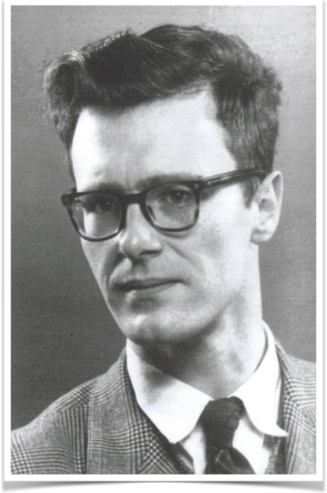
Yesterday's Answer:

Solution to 2/10/11

7	6	5	1	9	4	2	8	3
2	8	9	7	3	5	4	1	6
4	3	1	2	6	8	9	7	5
5	9	6	4	1	7	8	3	2
8	2	7	6	5	3	1	9	4
3	1	4	9	8	2	6	5	7
9	4	3	5	2	1	7	6	8
6	5	2	8	7	9	3	4	1
1	7	8	3	4	6	5	2	9

Applications

Dijkstra's algorithm



- Invented by Edsgar Dijkstra in 1959.
- The original version used a min-priority queue.
- Designed using pencil and paper; algorithm was intended to demonstrate to non-technical people how computers could be useful.

```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

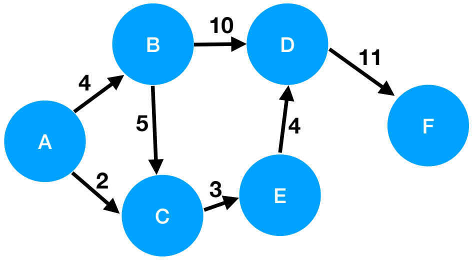
A	∞
B	∞
C	∞
D	∞
E	∞
F	∞
G	∞

prev

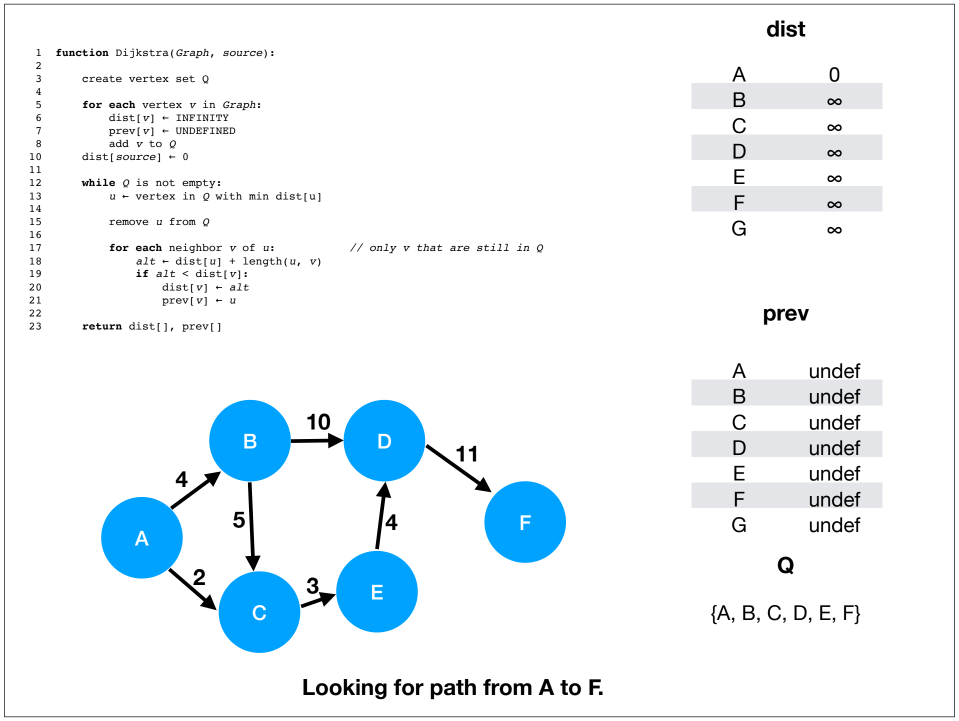
A	undef
B	undef
C	undef
D	undef
E	undef
F	undef
G	undef

Q

{A, B, C, D, E, F}



Looking for path from A to F.



```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

A	0
B	∞
C	∞
D	∞
E	∞
F	∞
G	∞

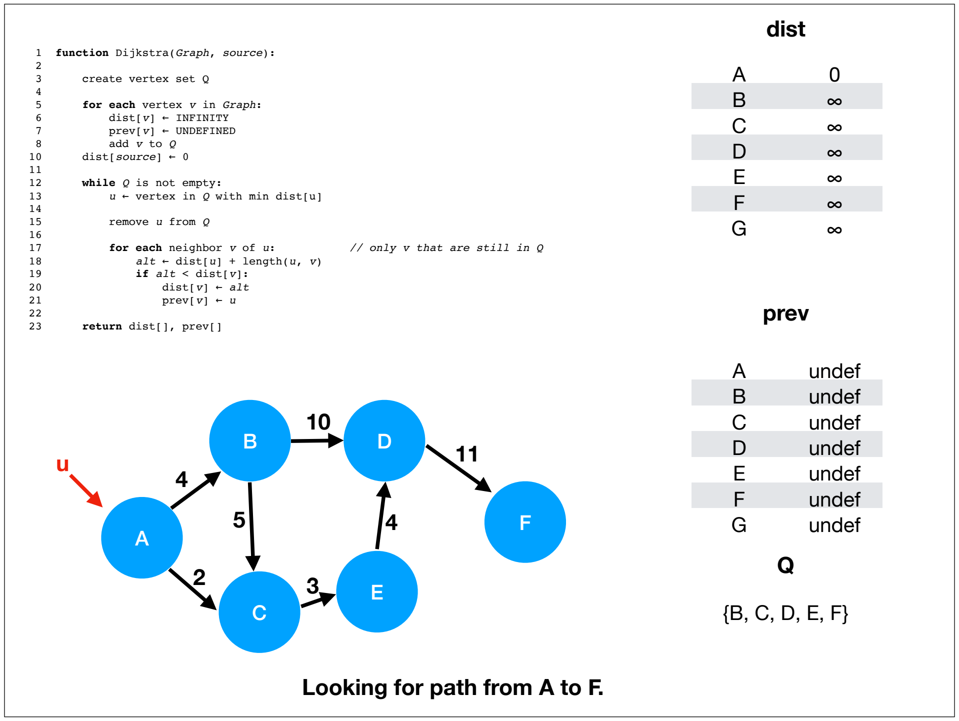
prev

A	undef
B	undef
C	undef
D	undef
E	undef
F	undef
G	undef

Q

{A, B, C, D, E, F}

Looking for path from A to F.



```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

A	0
B	∞
C	∞
D	∞
E	∞
F	∞
G	∞

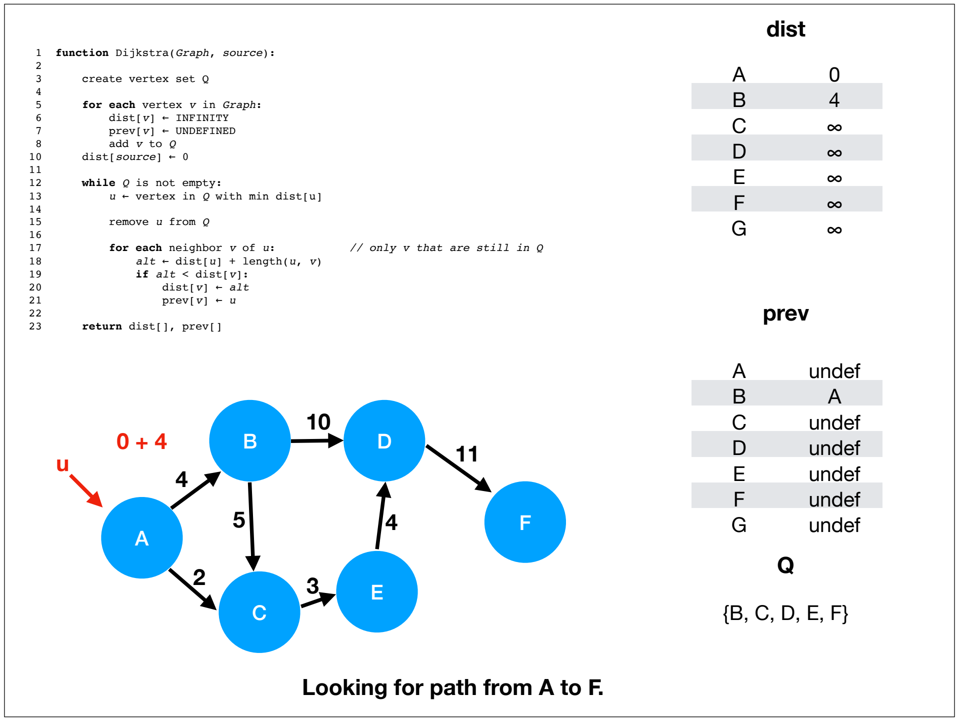
prev

A	undef
B	undef
C	undef
D	undef
E	undef
F	undef
G	undef

Q

{B, C, D, E, F}

Looking for path from A to F.



```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

A	0
B	4
C	∞
D	∞
E	∞
F	∞
G	∞

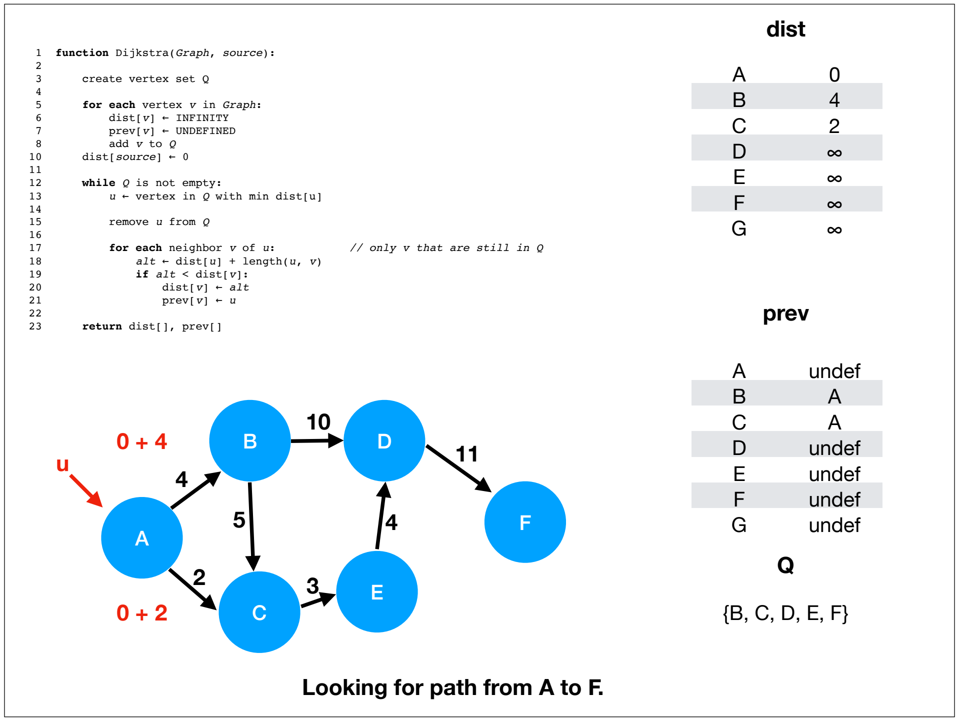
prev

A	undef
B	A
C	undef
D	undef
E	undef
F	undef
G	undef

Q

{B, C, D, E, F}

Looking for path from A to F.



```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

A	0
B	4
C	2
D	∞
E	∞
F	∞
G	∞

prev

A	undef
B	A
C	A
D	undef
E	undef
F	undef
G	undef

Q

{B, C, D, E, F}

Looking for path from A to F.

```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

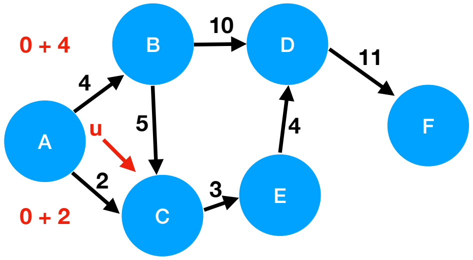
A	0
B	4
C	2
D	∞
E	∞
F	∞
G	∞

prev

A	undef
B	A
C	A
D	undef
E	undef
F	undef
G	undef

Q

{B, D, E, F}



Looking for path from A to F.

```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

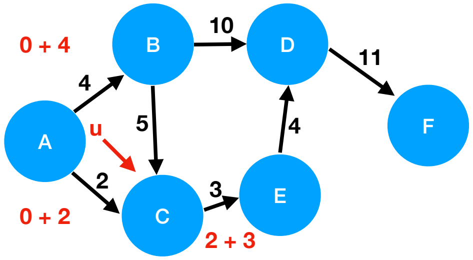
A	0
B	4
C	2
D	∞
E	5
F	∞
G	∞

prev

A	undef
B	A
C	A
D	undef
E	C
F	undef
G	undef

Q

{B, D, E, F}



Looking for path from A to F.

```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

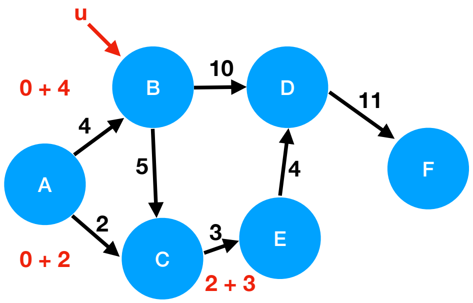
A	0
B	4
C	2
D	∞
E	5
F	∞
G	∞

prev

A	undef
B	A
C	A
D	undef
E	C
F	undef
G	undef

Q

{D, E, F}



Looking for path from A to F.

```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

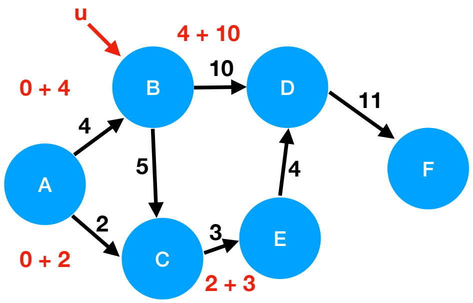
A	0
B	4
C	2
D	14
E	5
F	∞
G	∞

prev

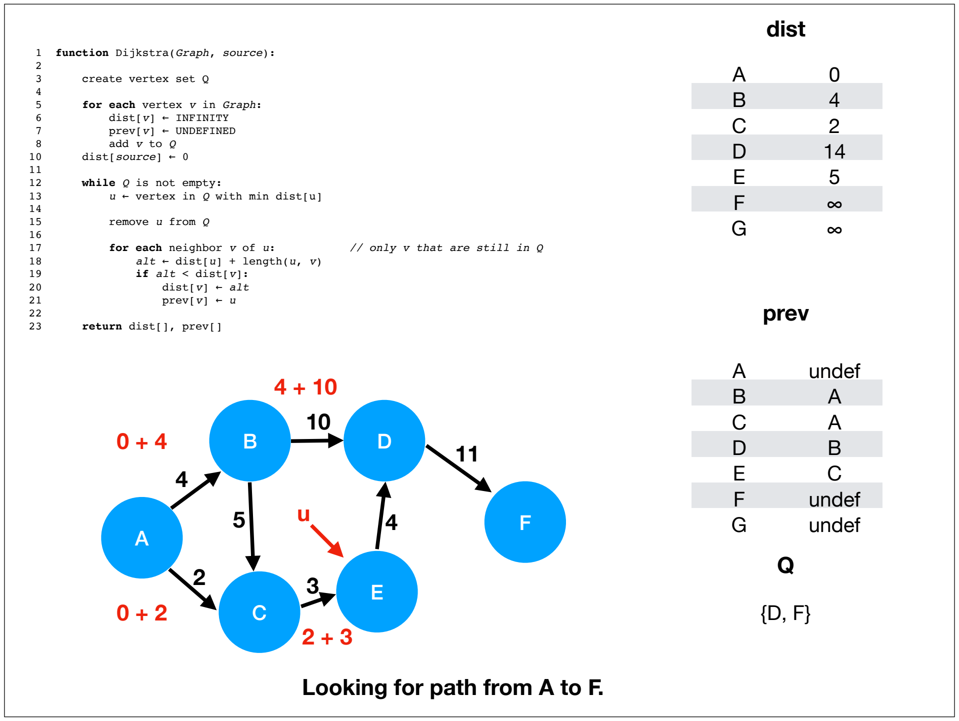
A	undef
B	A
C	A
D	B
E	C
F	undef
G	undef

Q

{D, E, F}



Looking for path from A to F.



```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

A	0
B	4
C	2
D	14
E	5
F	∞
G	∞

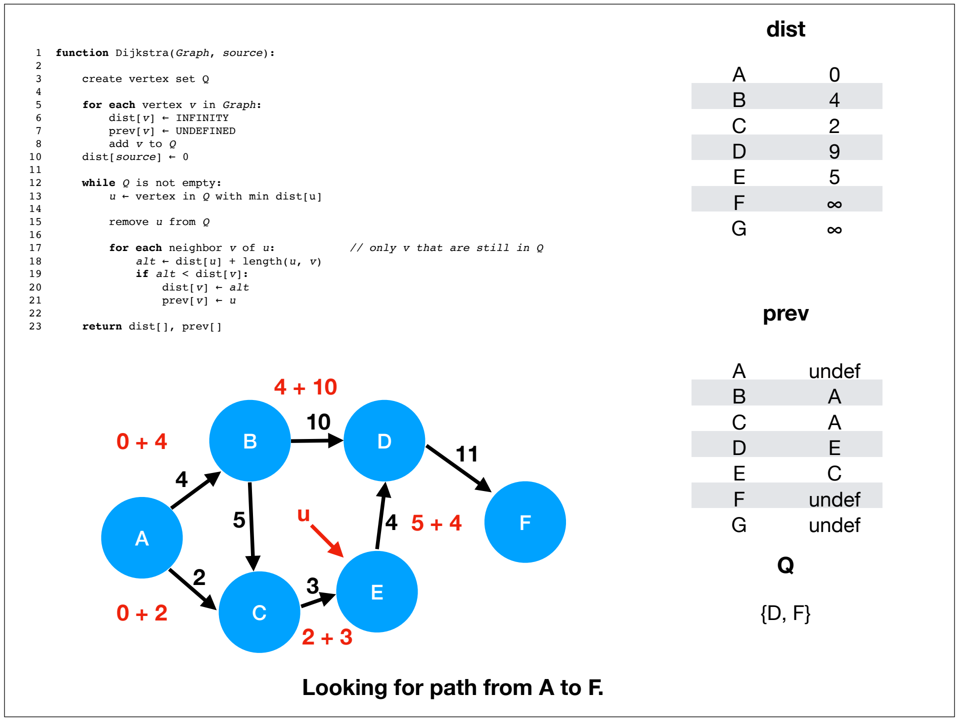
prev

A	undef
B	A
C	A
D	B
E	C
F	undef
G	undef

Q

{D, F}

Looking for path from A to F.



```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

A	0
B	4
C	2
D	9
E	5
F	∞
G	∞

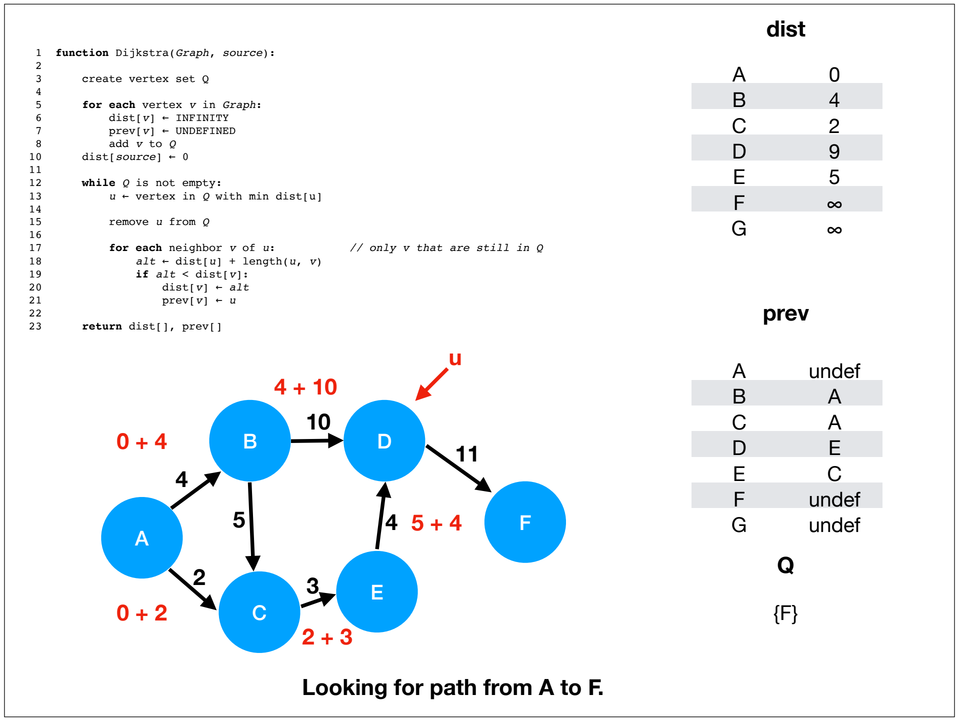
prev

A	undef
B	A
C	A
D	E
E	C
F	undef
G	undef

Q

{D, F}

Looking for path from A to F.



```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

A	0
B	4
C	2
D	9
E	5
F	∞
G	∞

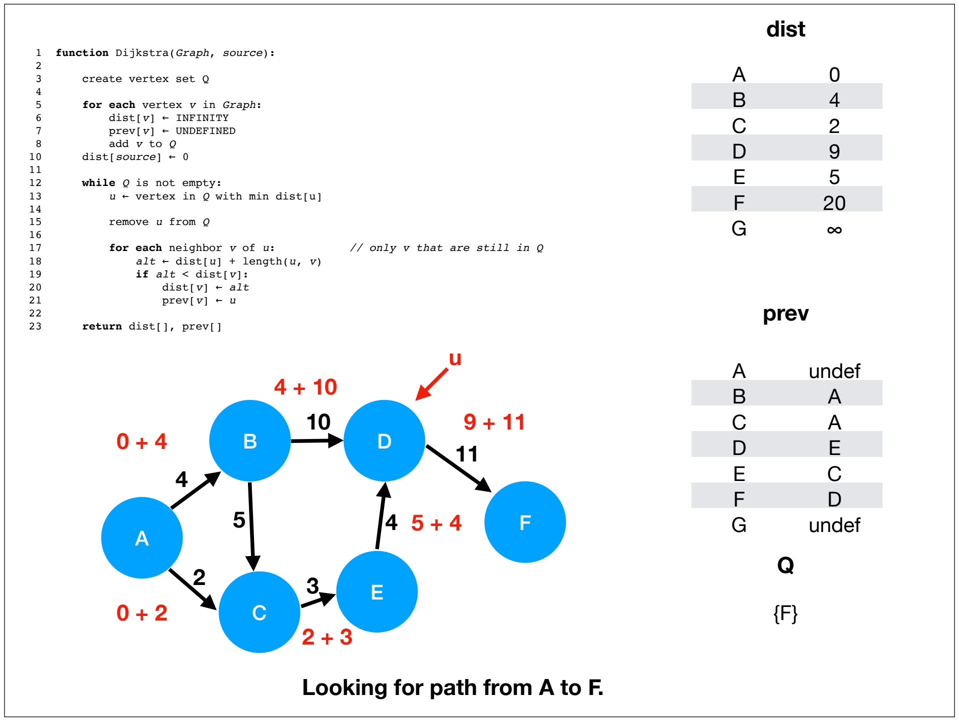
prev

A	undef
B	A
C	A
D	E
E	C
F	undef
G	undef

Q

{F}

Looking for path from A to F.



```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

A	0
B	4
C	2
D	9
E	5
F	20
G	∞

prev

A	undef
B	A
C	A
D	E
E	C
F	D
G	undef

Q

{F}

Looking for path from A to F.

```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

A	0
B	4
C	2
D	9
E	5
F	20
G	∞

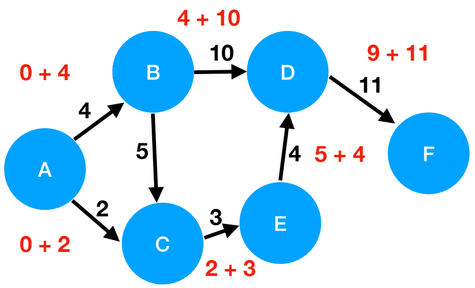
prev

A	undef
B	A
C	A
D	E
E	C
F	D
G	undef

Q

{}

Done!



Looking for path from A to F.

```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

dist

A	0
B	4
C	2
D	9
E	5
F	20
G	∞

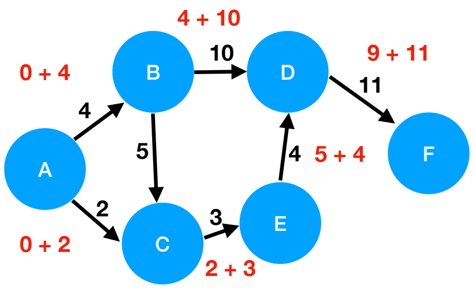
prev

A	undef
B	A
C	A
D	E
E	C
F	D
G	undef

Q

{}

Done!



Read backward from F and reverse.

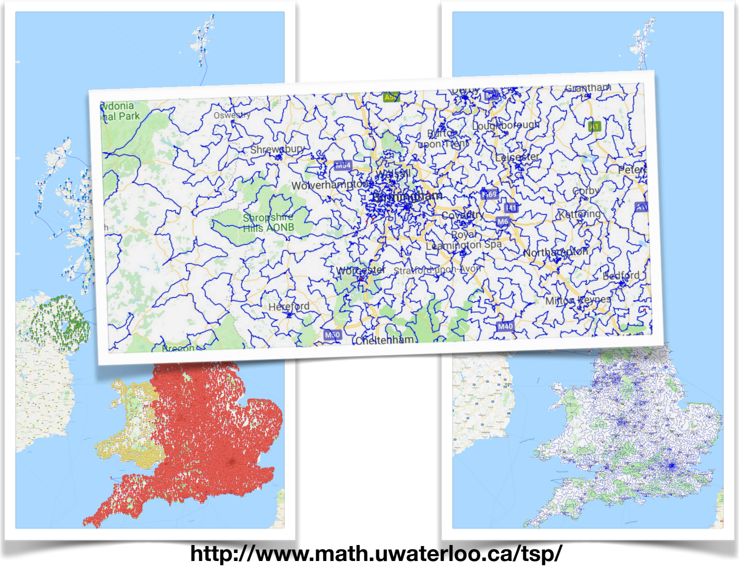
Graphs: traveling salesperson

Applications

Delivery routes.

Applications

Optimal 49,687-stop pub crawl



http://www.math.uwaterloo.ca/tsp/

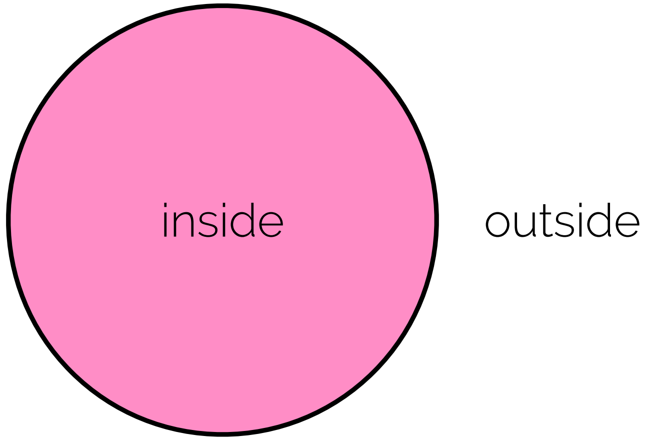
You learned a lot this semester!

(great job!)

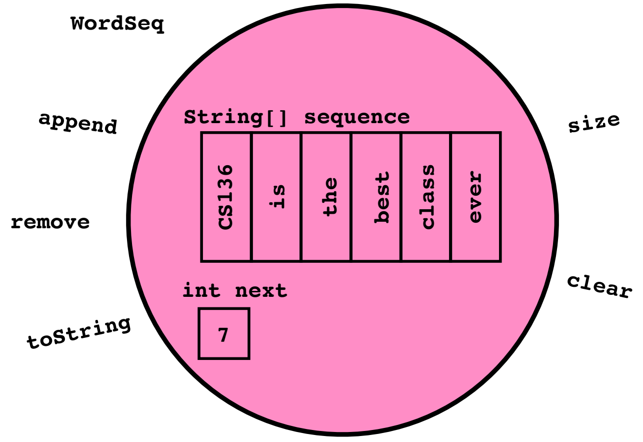
Java

Program design

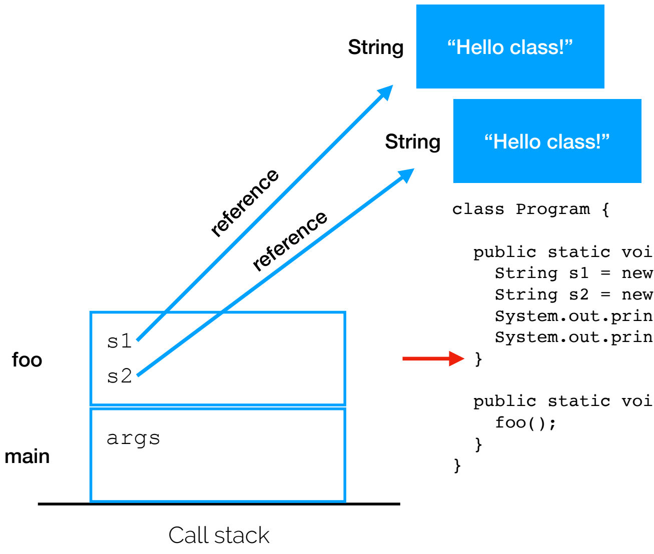
Abstraction



Composition

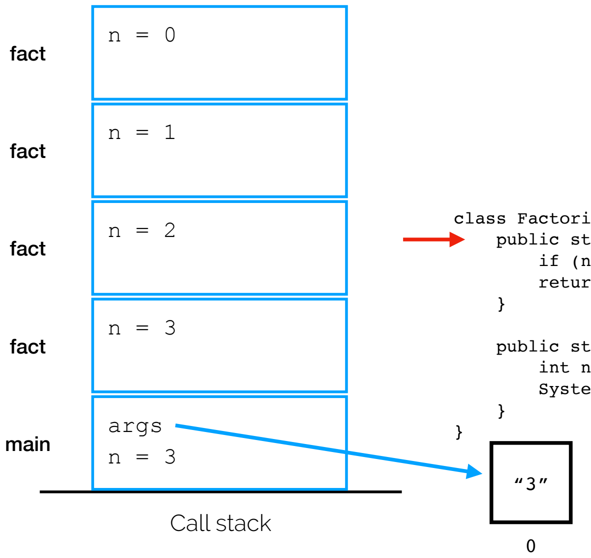


Abstract machine



```
class Program {

  public static void foo() {
    String s1 = new String(“Hello class!”);
    String s2 = new String(“Hello class!”);
    System.out.println(s1 == s2);
    System.out.println(s1.equals(s2));
  }

  public static void main(String[] args) {
    foo();
  }
}
```

Recursion



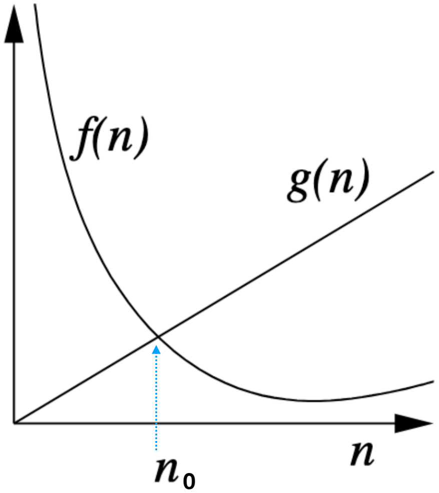
```
class Factorial {
  public static int fact(int n) {
    if (n == 0) { return 1; }
    return n * fact(n - 1);
  }

  public static void main(String[] args) {
    int n = Integer.parseInt(args[0]);
    System.out.println(fact(n));
  }
}
```

Formal methods

Induction

Program performance

Big-O analysis



Algorithm design

# of copies for doubling expansion:

	1	+	2	+	4	+	…	+(n/2)
add()	up to 2nd elem.		up to 4th elem.		up to 8th elem.			up to nth elem.

Neat theorem: $1 + 2 + 4 + \ldots + 2^{k-1} = 2^k - 1$

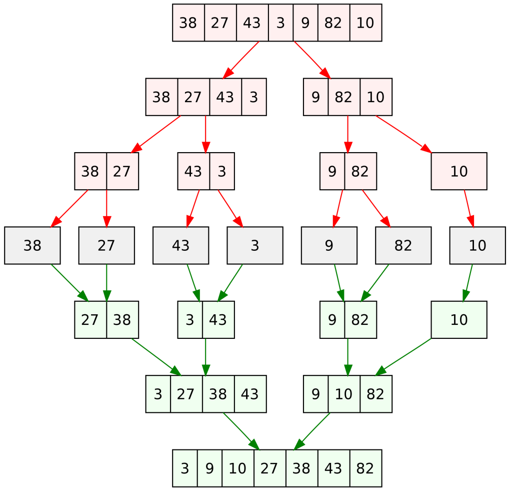
Suppose $n = 2^k$.

Then $1 + \ldots + n/2 = 1 + \ldots + 2^k/2$

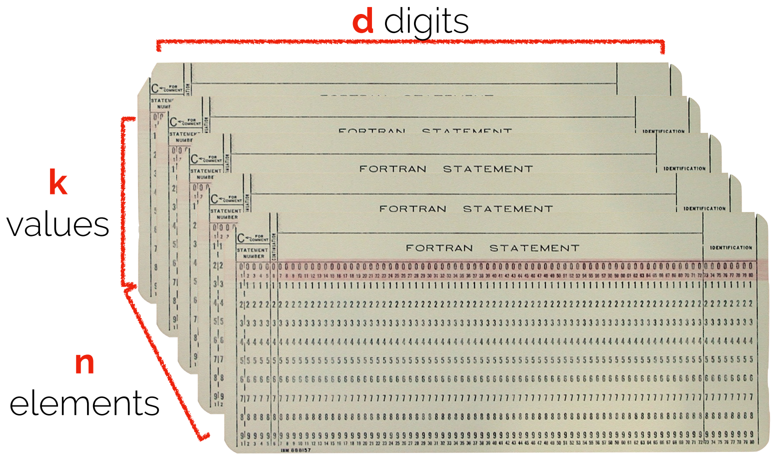
$= 1 + \ldots + 2^{k-1} = 2^k - 1 = n - 1$

Doubling expansion costs ≈ **O(n)**

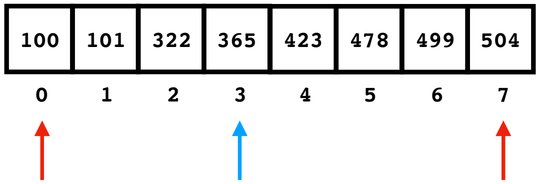
Sorting algorithms



Exotic sorting algorithms



Search algorithms

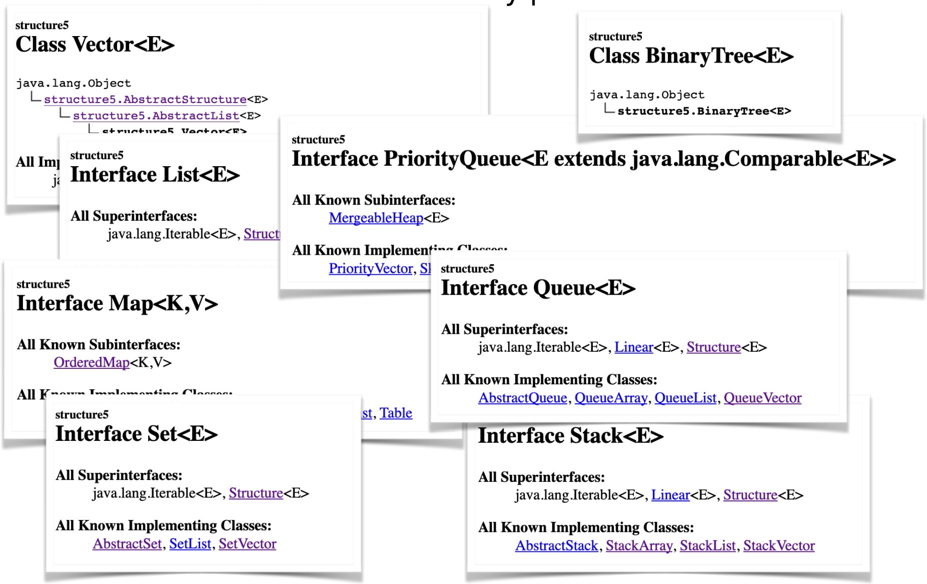


100	101	322	365	423	478	499	504
0	1	2	3	4	5	6	7

322 = 365? **no**

322 < 365? **yes**

Abstract data types (ADTs)



structure5
Class Vector<E>

```
java.lang.Object
  └structure5.AbstractStructure<E>
      └structure5.AbstractList<E>
          └structure5.Vector<E>
```

All Imp
ja

structure5
Class BinaryTree<E>

```
java.lang.Object
  └structure5.BinaryTree<E>
```

structure5
Interface List<E>

All Superinterfaces:
java.lang.Iterable<E>, Struct

structure5
Interface PriorityQueue<E extends java.lang.Comparable<E>>

All Known Subinterfaces:
MergeableHeap<E>

All Known Implementing Classes:
PriorityVector, Sl

structure5
Interface Map<K,V>

All Known Subinterfaces:
OrderedMap<K,V>

All Known Implementing Classes:
st, Table

structure5
Interface Queue<E>

All Superinterfaces:
java.lang.Iterable<E>, Linear<E>, Structure<E>

All Known Implementing Classes:
AbstractQueue, QueueArray, QueueList, QueueVector

structure5
Interface Set<E>

All Superinterfaces:
java.lang.Iterable<E>, Structure<E>

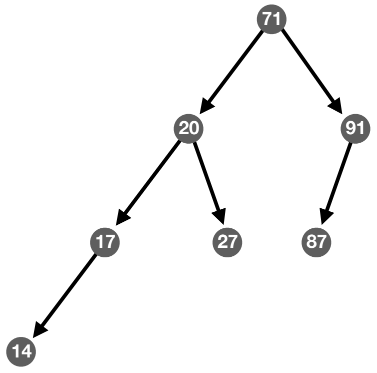
All Known Implementing Classes:
AbstractSet, SetList, SetVector

Interface Stack<E>

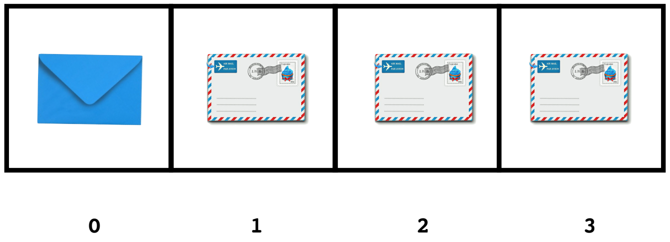
All Superinterfaces:
java.lang.Iterable<E>, Linear<E>, Structure<E>

All Known Implementing Classes:
AbstractStack, StackArray, StackList, StackVector

Ordered structures



Partially-ordered structures

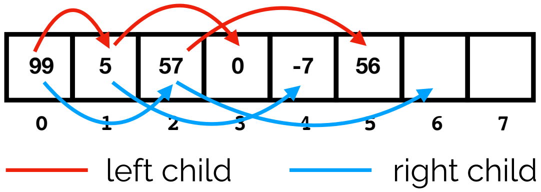


✉ Ordinary letter ✉ Blue letter

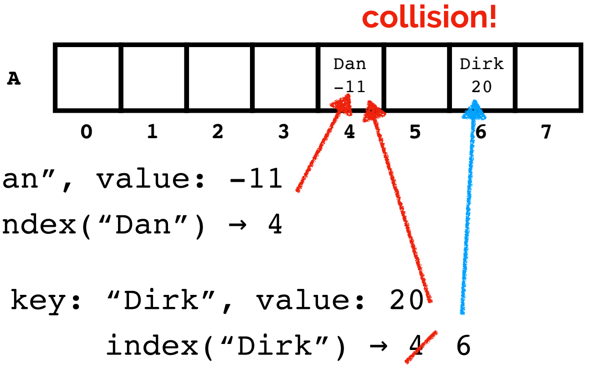
Number representations

0	1	1	1	1	0	0	1
7	6	5	4	3	2	1	0

Efficient encoding of structures

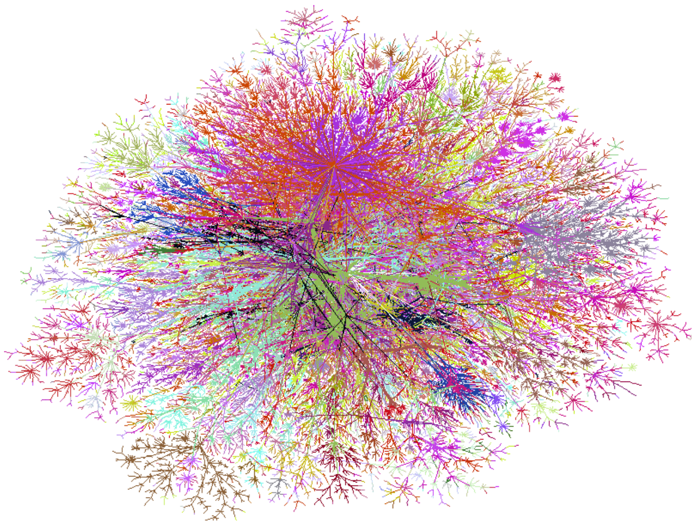


High-performance structures

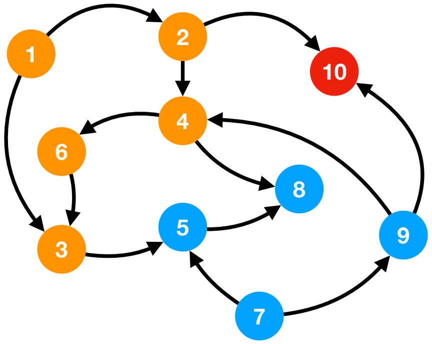


```
key: “Dan”, value: -11
     index(“Dan”) → 4

     key: “Dirk”, value: 20
          index(“Dirk”) → ~~4~~ 6
```

Very general structures: graphs



Graph algorithms



Major declaration

(it'll happen in June or July)

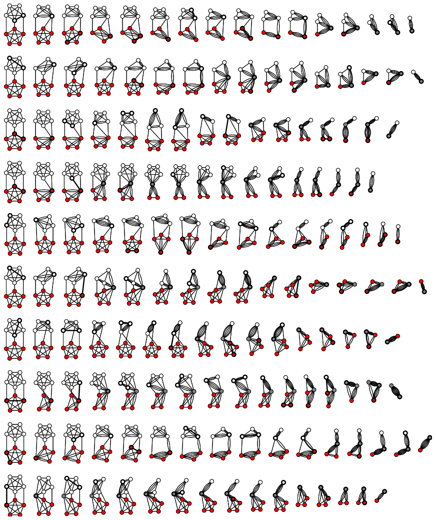
Final exam info

Final exam info

- Posted from May 20-May 25 on GLOW.
- As before: choose a 3-hour window to take the exam.
- Structure: 6-7 questions.
- Open book.
- Covers all material from the semester; more emphasis on material in second half.
- Question form: What is the most appropriate data structure?
  - Justify in terms of ADT guarantees, Big-O, etc.
  - Note that this is an open book exam!
- AFAIK, all of you are doing great so far.
  - If you're worried about not passing, get in touch! We are happy to talk with you privately and offer support.

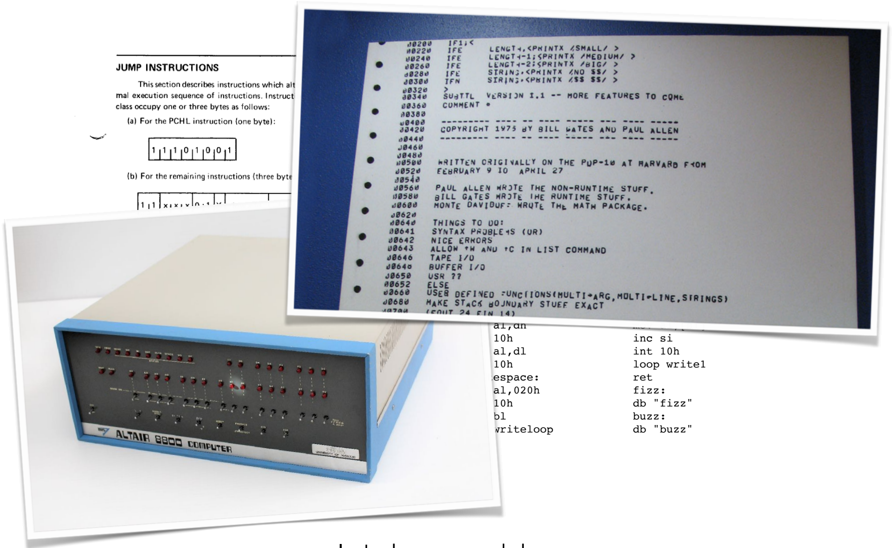
Life after CS136

CS256: Analysis of Algorithms



(10 runs of Karger's randomized min-cut algorithm)

CS237: Computer Organization



JUMP INSTRUCTIONS

This section describes instructions which alt
mal execution sequence of instructions. Instructi
class occupy one or three bytes as follows:

(a) For the PCHL instruction (one byte):

1	1	1	0	1	0	0	1

(b) For the remaining instructions (three byte

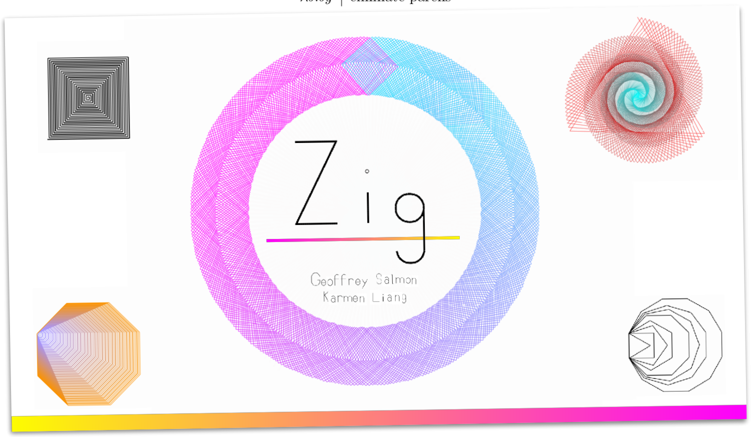
```
00200   IF1;<
00220   IFE      LENGTH,<PRINTX /SMALL/ >
00240   IFE      LENGTH-1;<PRINTX /MEDIUM/ >
00260   IFE      LENGTH-2;<PRINTX /BIG/ >
00280   IFE      STRING,<PRINTX /NO $$/ >
00300   IFN      STRING,<PRINTX /$$ $$/ >
00320   >
00340   SUBTTL  VERSION 1.1 -- MORE FEATURES TO COME
00360   COMMENT *
00380
00400   --------- ---- -- ---- ----- --- ---- -----
00420   COPYRIGHT 1975 BY BILL GATES AND PAUL ALLEN
00440   --------- ---- -- ---- ----- --- ---- -----
00460
00480
00500   WRITTEN ORIGINALLY ON THE PDP-10 AT HARVARD FROM
00520   FEBRUARY 9 TO  APRIL 27
00540
00560   PAUL ALLEN WROTE THE NON-RUNTIME STUFF.
00580   BILL GATES WROTE THE RUNTIME STUFF.
00600   MONTE DAVIDOFF WROTE THE MATH PACKAGE.
00620
00640   THINGS TO DO:
00641   SYNTAX PROBLEMS (OR)
00642   NICE ERRORS
00643   ALLOW ↑W AND ↑C IN LIST COMMAND
00646   TAPE I/O
00648   BUFFER I/O
00650   USR ??
00652   ELSE
00660   USER DEFINED FUNCTIONS(MULTI-ARG,MULTI-LINE,STRINGS)
00680   MAKE STACK BOUNDARY STUFF EXACT
00700   (FOUT 24 FIN 14)
```

```
al,dh
10h             inc si
al,dl           int 10h
10h             loop write1
espace:         ret
al,020h         fizz:
10h             db "fizz"
bl              buzz:
writeloop       db "buzz"
```

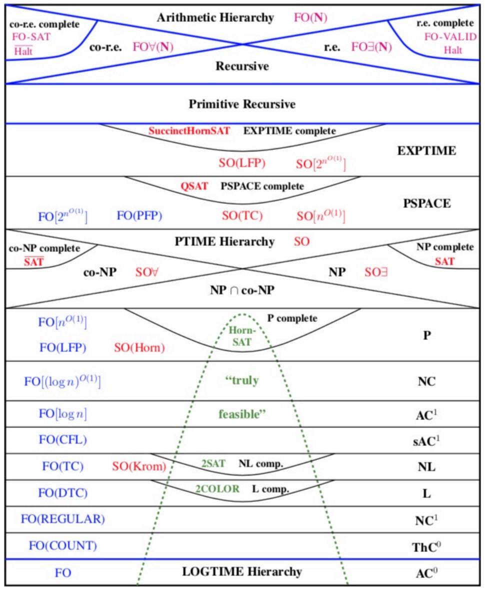
Intel assembly

CS334: Principles of PL

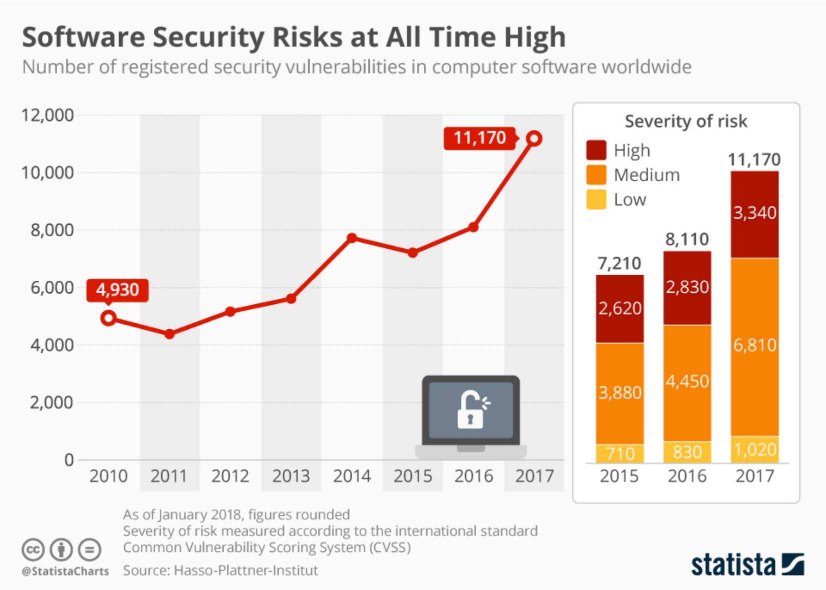
$(\lambda x.\lambda y.xy)(\lambda x.xy)$	given
$(\lambda a.\lambda y.ay)(\lambda x.xy)$	α reduce x with a
$(\lambda a.\lambda b.ab)(\lambda x.xy)$	α reduce y with b
$([(\lambda x.xy)/a]\lambda b.ab)$	β reduce a with $(\lambda x.xy)$
$(\lambda b.(\lambda x.xy)b)$	sub
$(\lambda b.([b/x]xy))$	β reduce x with b
$(\lambda b.(by))$	sub
$\lambda b.by$	elminate parens



CS361: Theory of Computation

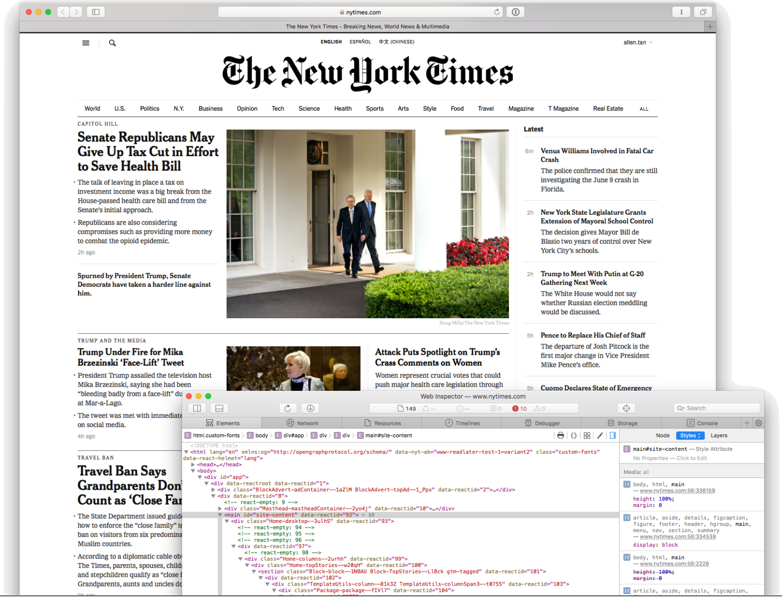


CS331: Intro. to Computer Security



CS338: Parallel Processing

CS343: App. Dev. with Functional Prog.



CAPITOL HILL

Senate Republicans May Give Up Tax Cut in Effort to Save Health Bill

- The talk of leaving in place a tax on investment income was a big break from the House-passed health care bill and from the Senate's initial approach.
- Republicans are also considering compromises such as providing more money to combat the opioid epidemic.

2h ago

Spurned by President Trump, Senate Democrats have taken a harder line against him.

Doug Mills/The New York Times

TRUMP AND THE MEDIA

Trump Under Fire for Mika Brzezinski 'Face-Lift' Tweet

- President Trump assailed the television host Mika Brzezinski, saying she had been "bleeding badly from a face-lift" du
at Mar-a-Lago.
- The tweet was met with immediate
on social media.

4h ago

Attack Puts Spotlight on Trump's Crass Comments on Women

Women represent crucial votes that could push major health care legislation through

TRAVEL BAN

Travel Ban Says Grandparents Don Count as 'Close Fa

- The State Department issued guid how to enforce the "close family" t ban on visitors from six predomina Muslim countries.
- According to a diplomatic cable ob The Times, parents, spouses, child and stepchildren qualify as "close f Grandparents, aunts and uncles d

Latest

6m **Venus Williams Involved in Fatal Car Crash**
The police confirmed that they are still investigating the June 9 crash in Florida.

2h **New York State Legislature Grants Extension of Mayoral School Control**
The decision gives Mayor Bill de Blasio two years of control over New York City's schools.

2h **Trump to Meet With Putin at G-20 Gathering Next Week**
The White House would not say whether Russian election meddling would be discussed.

5h **Pence to Replace His Chief of Staff**
The departure of Josh Pitcock is the first major change in Vice President Mike Pence's office.

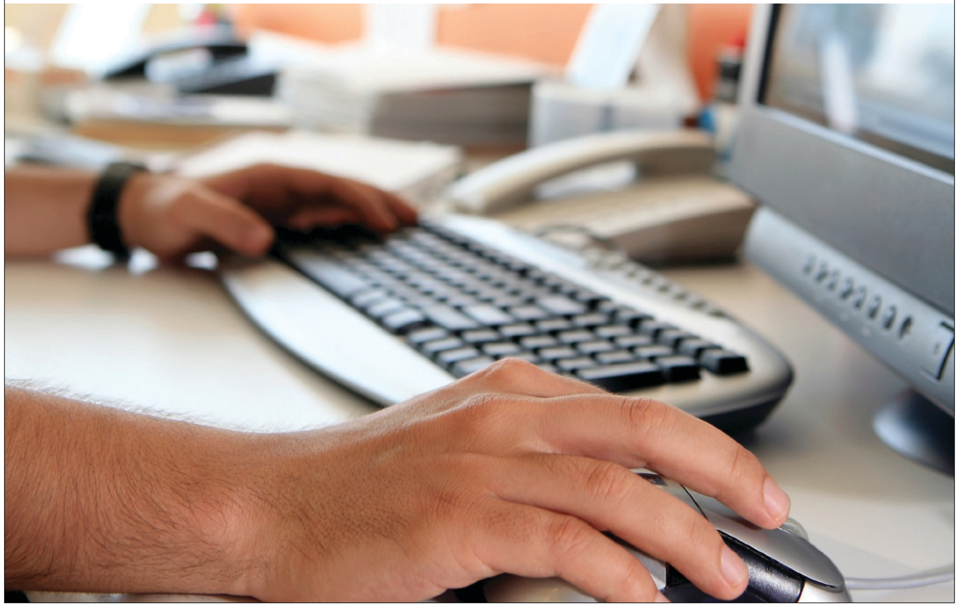
5h **Cuomo Declares State of Emergency**

html.custom-fonts 〉 body 〉 div#app 〉 div 〉 div 〉 main#site-content

```
<!DOCTYPE html>
▼ <html lang="en" xmlns:og="http://opengraphprotocol.org/schema/" data-nyt-ab="www-readlater-test-1=variant2" class="custom-fonts"
data-react-helmet="lang">
  ▶ <head>…</head>
  ▼ <body>
    ▼ <div id="app">
      ▼ <div data-reactroot data-reactid="1">
        ▶ <div class="BlockAdvert-adContainer--1aZlM BlockAdvert-topAd--1_Ppx" data-reactid="2">…</div>
        ▼ <div data-reactid="8">
            <!-- react-empty: 9 -->
          ▶ <div class="Masthead-mastheadContainer--2yo4j" data-reactid="10">…</div>
          ▼ <main id="site-content" data-reactid="92"> = $0
            ▼ <div class="Home-desktop--3ulhS" data-reactid="93">
                <!-- react-empty: 94 -->
                <!-- react-empty: 95 -->
                <!-- react-empty: 96 -->
              ▼ <div data-reactid="97">
                  <!-- react-empty: 98 -->
                ▼ <div class="Home-columns--2urhh" data-reactid="99">
                  ▼ <div class="Home-topStories--w20qH" data-reactid="100">
                    ▼ <section class="Block-block--1NBAU Block-TopStories--LlBck gtm-tagged" data-reactid="101">
                      ▼ <div data-reactid="102">
                        ▼ <div class="TemplateUtils-column--81k3Z TemplateUtils-columnSpan3--t0755" data-reactid="103">
                          ▼ <div class="Package-package--fIVl7" data-reactid="104">
```

```
main#site-content — Style Attribute
No Properties — Click to Edit

Media: all

body, html, main
— www.nytimes.com:56:336159
height: 100%;
margin: 0

article, aside, details, figcaption,
figure, footer, header, hgroup, main,
menu, nav, section, summary
— www.nytimes.com:56:334539
display: block

body, html, main
— www.nytimes.com:56:2228
~~height: 100%;~~
~~margin: 0~~

article, aside, details, figcaption,
```

CS376: Human-Computer Interaction



CS315: Computational Biology

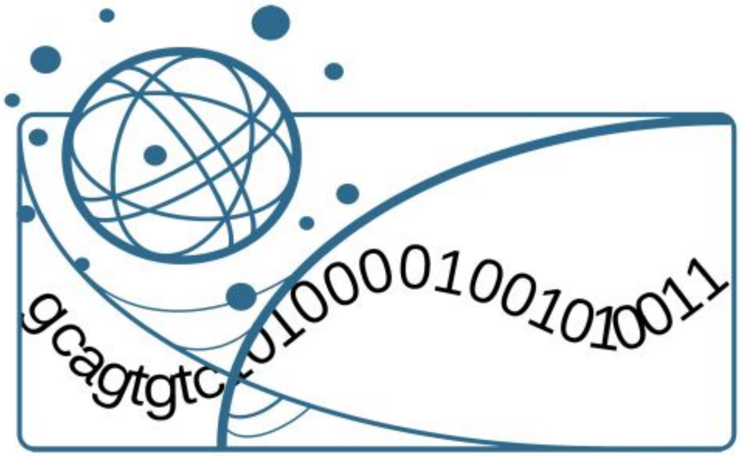
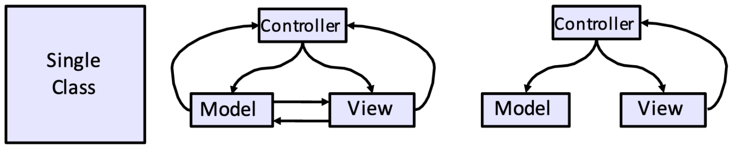


Image credit: Alper Uzun. *PLoS Computational Biology*

CS326: Software Methods

$$\frac{\{P\}S\{Q\} \quad , \quad \{Q\}T\{R\}}{\{P\}S;T\{R\}}$$

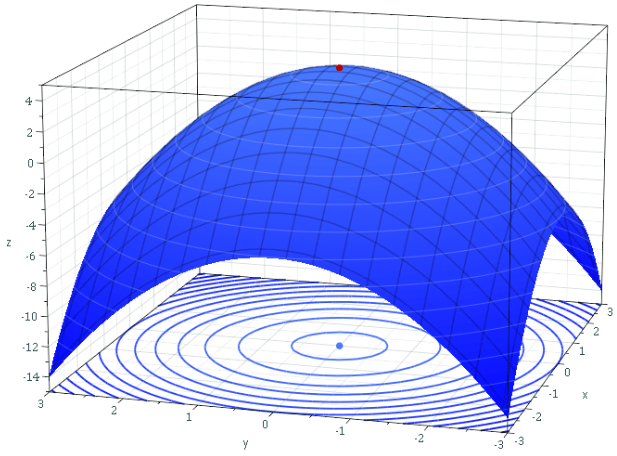


"god class" *Strongly coupled* *Weakly coupled*

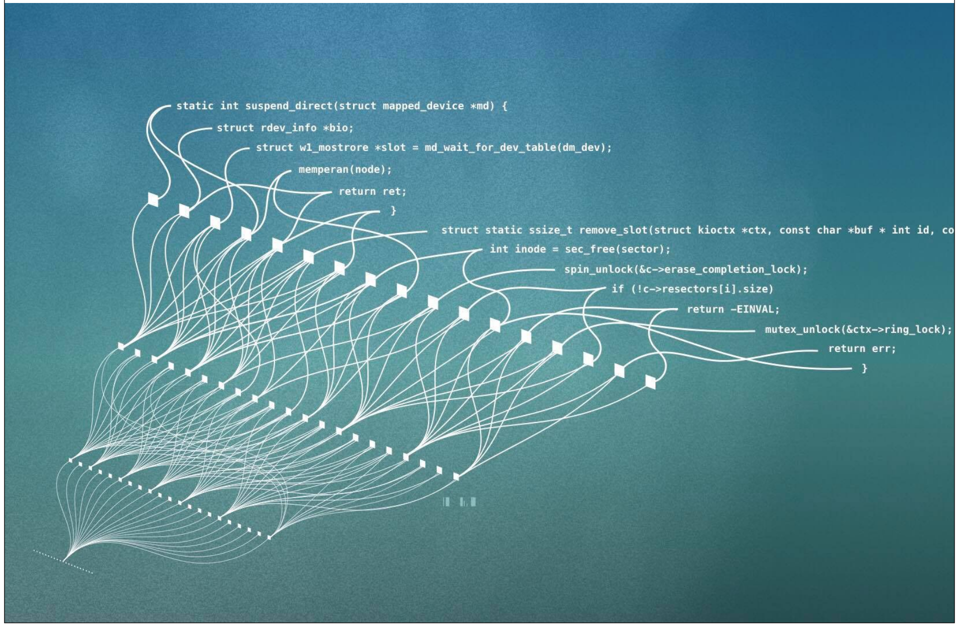
CS333: Storage Systems

CS339: Distributed Systems

CS358: Applied Algorithms



CS374: Machine Learning

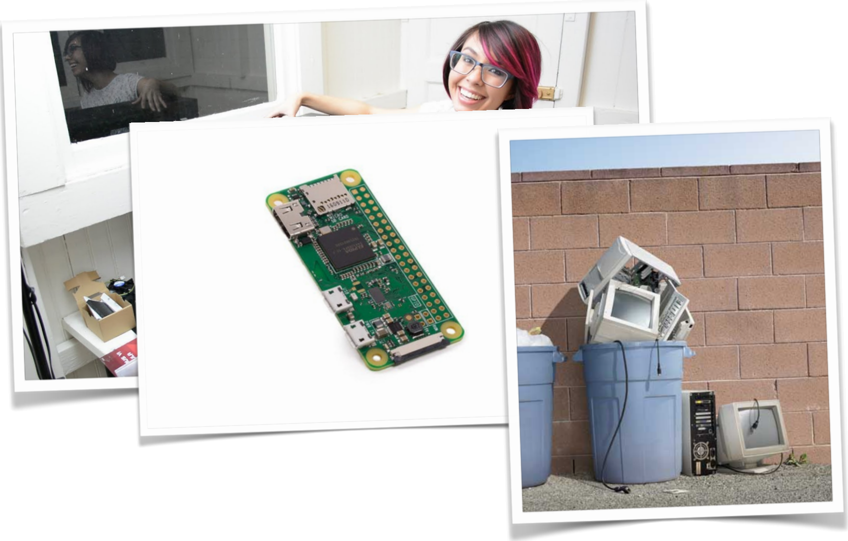


```
static int suspend_direct(struct mapped_device *md) {
struct rdev_info *bio;
struct w1_mostrore *slot = md_wait_for_dev_table(dm_dev);
memperan(node);
return ret;
}
struct static ssize_t remove_slot(struct kioctx *ctx, const char *buf * int id, co
int inode = sec_free(sector);
spin_unlock(&c->erase_completion_lock);
if (!c->resectors[i].size)
return -EINVAL;
mutex_unlock(&ctx->ring_lock);
return err;
}
```

Summer projects

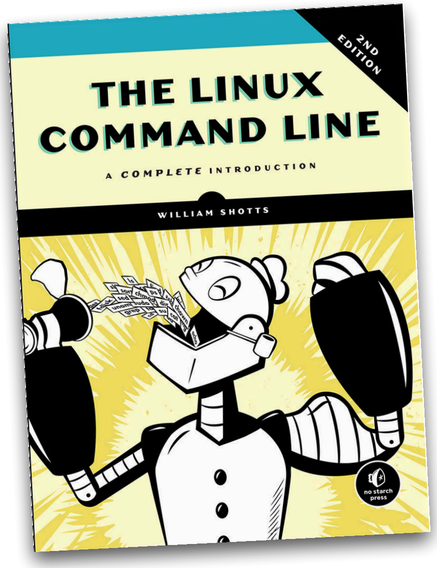
Things that work for me™
be the hero in your own education

Build a computer



https://www.cpu-monkey.com/en/compare_cpu-intel_core_i7_2600k-6-vs-intel_core_i5_8210y-954

Learn Linux



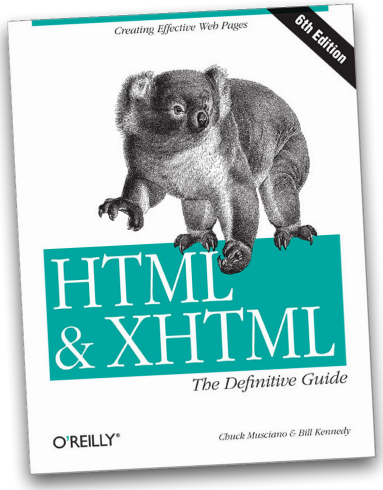
BIZ & IT —

How to run your own e-mail server with your own domain, part 1

Gmail? Apple? The cloud? Forget 'em all—in this series, we take your e-mail back.

LEE HUTCHINSON - 2/16/2014, 9:00 PM

Make your own website



We'll post more ideas soon!

Things that work for me™
physical health = mental health

Recap & Next Class

Today we learned:

- Shortest paths
- Dijkstra's algorithm
- Recap
- Exam info

Next class:

No next class: good luck on the final!