Lecture 17: Wrapup

CS 106L, Fall '20

Today's Agenda

- Assn2
- Cool new features in C++17, C++20
- Future directions in CS

Assignment 2

Milestone 2

- Due tonight!
- Let us know if you need extra time :)

Const-correctness

If you call a function on a **const** object, that function must be **const**

```
void global_func(const Obj& a, const Obj& b) {
    a.foo();
    b.foo();
}

Obj::foo() const {          // needs to be const, or compilation error
    ...
}
```

Variadic templates

(C++11)

Variadic templates

Allow for templates with a **variable** number of arguments!

```
template<typename T>
T adder(T v) {
  return v;
}

template<typename T, typename... Args>
T adder(T first, Args... args) {
  return first + adder(args...);
}

adder(5, 6, 7, 8)           // 26
```

How does it work?

Overload resolution!

```
template<typename T>
T adder(T v) {                                  // this one is called when there is
  return v;                                     // 1 argument
}

template<typename T, typename... Args>
T adder(T first, Args... args) {                // this one is called when there are
  return first + adder(args...);                // >=2 arguments
}
```

Spaceship operator

(C++17)

Spaceship operator

Writing this boilerplate code is annoying:

```
struct IntWrapper {
  int value;
  IntWrapper(int value): value{value} { }
  bool operator==(const IntWrapper& rhs) const { return value == rhs.value; }
  bool operator!=(const IntWrapper& rhs) const { return !(*this == rhs);    }
  bool operator<(const IntWrapper& rhs)  const { return value < rhs.value;  }
  bool operator<=(const IntWrapper& rhs) const { return !(rhs < *this);     }
  bool operator>(const IntWrapper& rhs)  const { return rhs < *this;        }
  bool operator>=(const IntWrapper& rhs) const { return !(*this < rhs);     }
};
```

Spaceship operator

If you write a single ⇔ operator, everything will be autogenerated for you

```
struct IntWrapper {
  int value;
  IntWrapper(int value): value(value) { }
  auto operator<=>(const int& rhs) auto {
      return value <=> rhs;
  }
};

IntWrapper(5) < IntWrapper(7)       // returns true
```

Spaceship operator

Basically, return -1, 0, or 1 as appropriate:

```
struct IntWrapper {
  int value;
  IntWrapper(int value): value(value) { }
  auto operator<=>(const int& rhs) auto {
      if (value < rhs) return -1;
      else if (value == rhs) return 0;
      else return 1;
  }
};

IntWrapper(5) < IntWrapper(7)       // returns true
```

Designated initializers

(C++20)

Better struct initialization syntax!

Non-specified values → default initialization

```
struct A {
  int x;
  int y;
  int z = 123;
};

A a {.x = 1, .z = 2}; // a.x == 1, a.y == 0, a.z == 2
```

[[likely]]

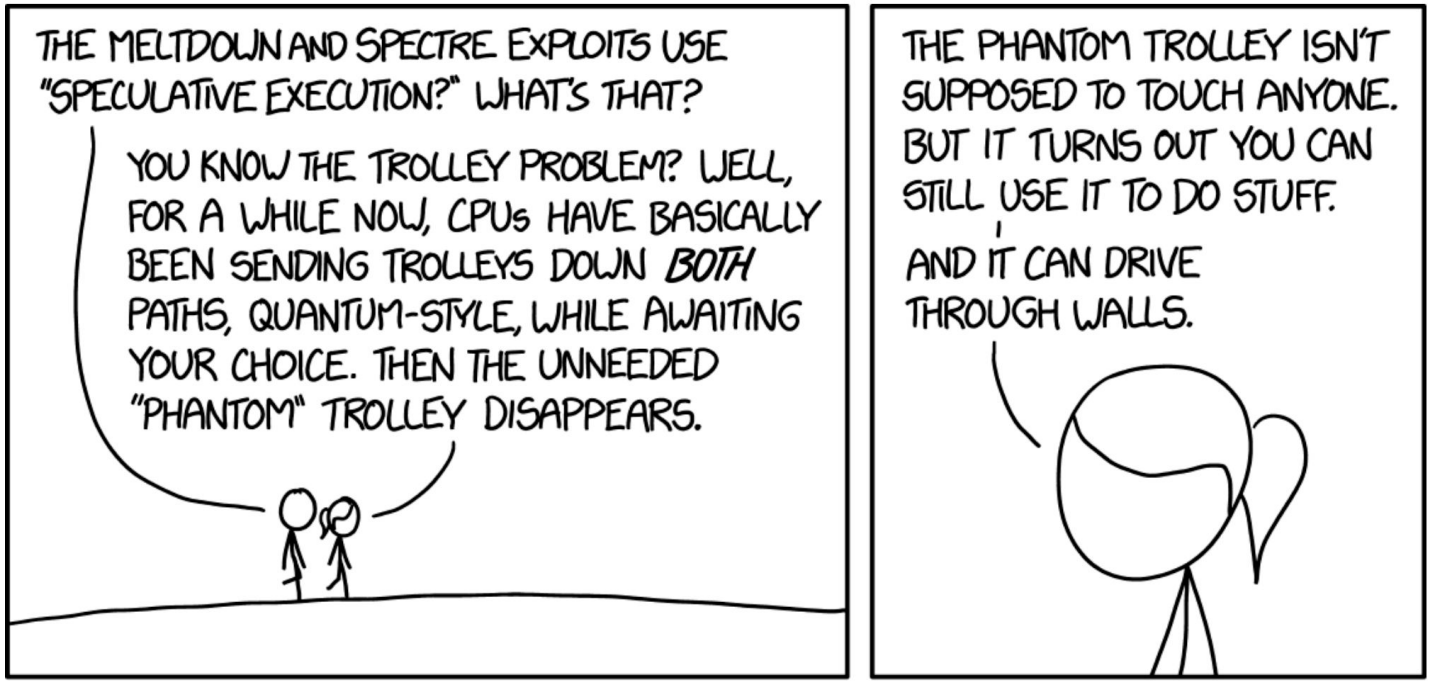
(C++20)

“Compiler, we have a problem...”

Use the **[[likely]]** operator to mark things that probably will run...

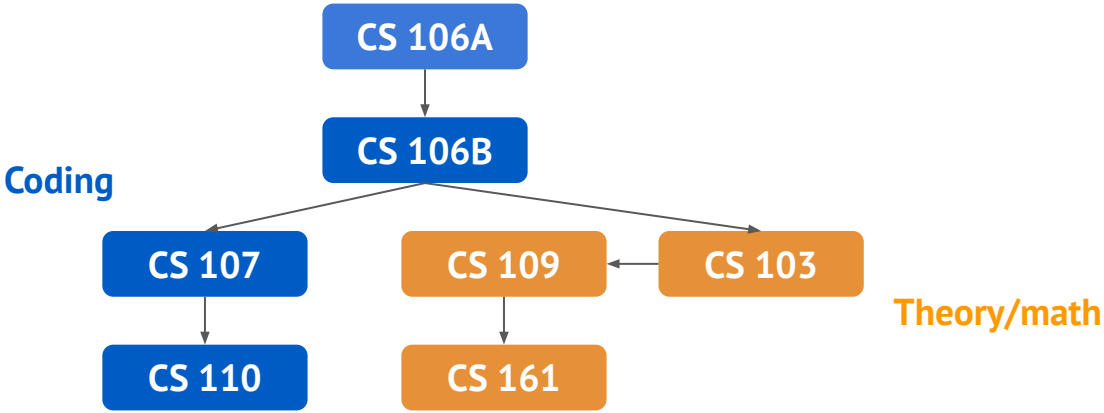
```
int random = get_random_number_between_x_and_y(0, 100);
[[likely]] if (random > 0) {
  // body of if statement; efficiency will be prioritized
}

[[unlikely]] if (random == 0) {
  // body of if statement; efficiency will not be prioritized
}
```

How does this work?

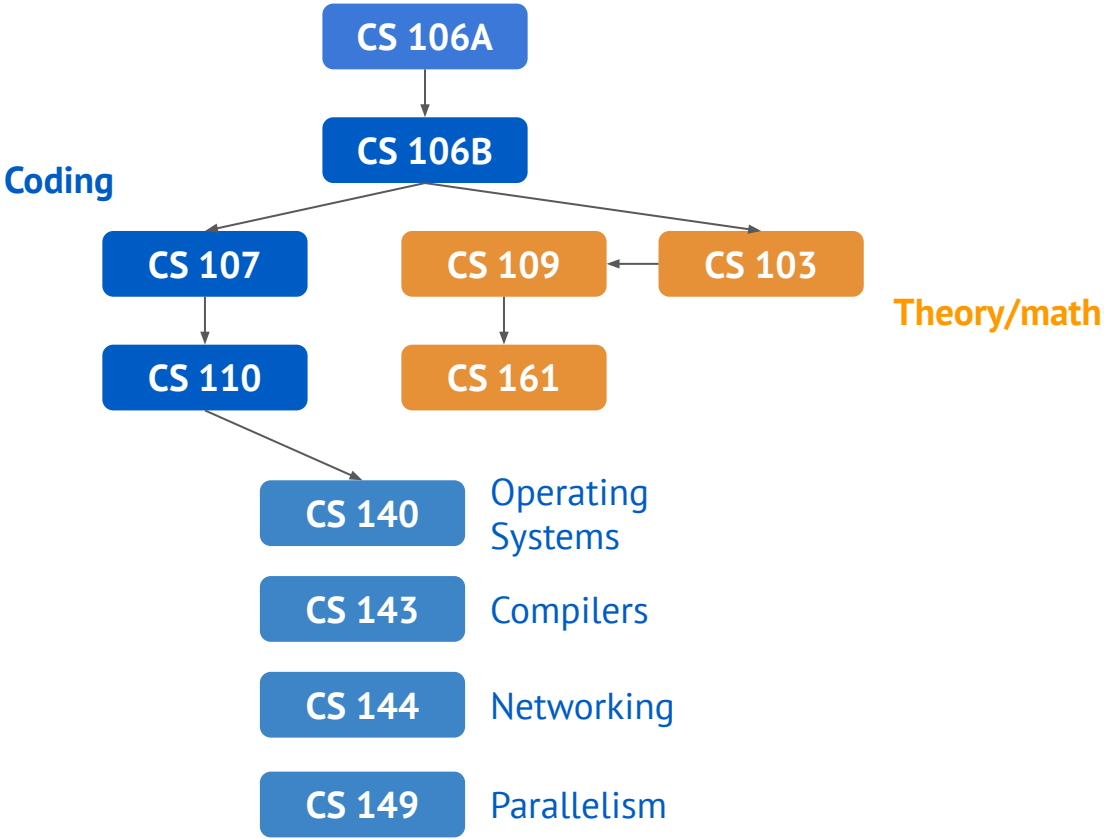


xkcd.com/1938/

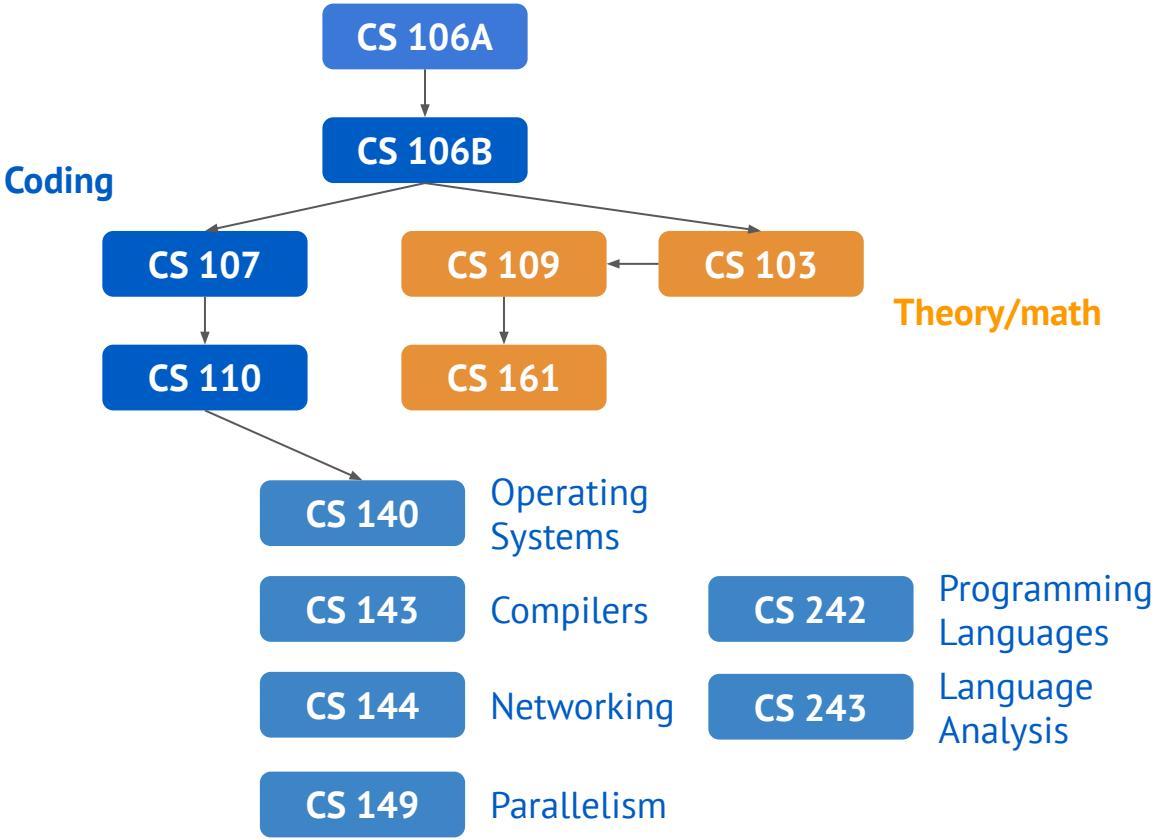
Future Directions in CS


CS 106A
CS 106B
Coding
CS 107
CS 109
CS 103
Theory/math
CS 110
CS 161

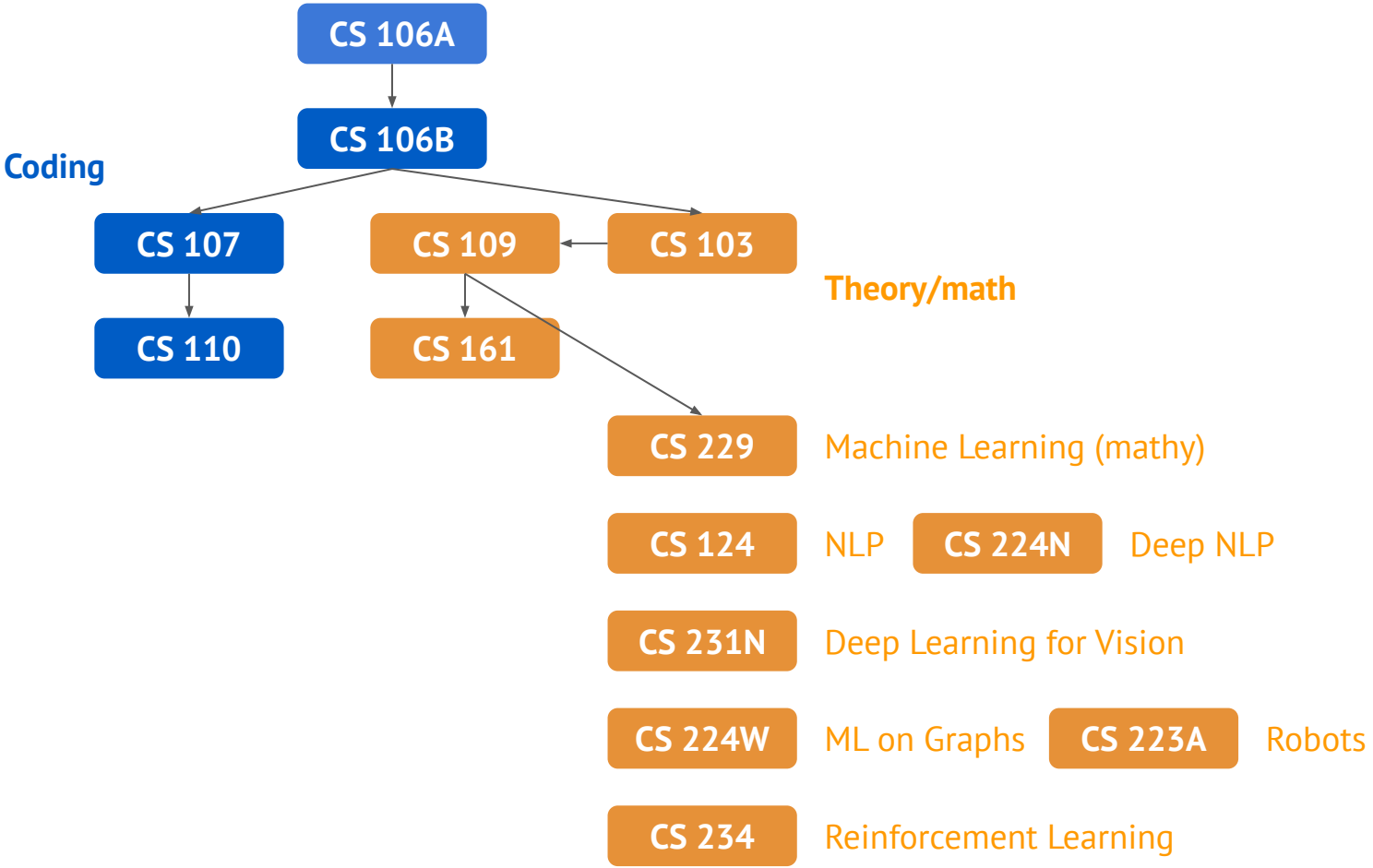
Systems


CS 106A
CS 106B
Coding
CS 107
CS 109
CS 103
Theory/math
CS 110
CS 161
CS 140
Operating Systems
CS 143
Compilers
CS 144
Networking
CS 149
Parallelism

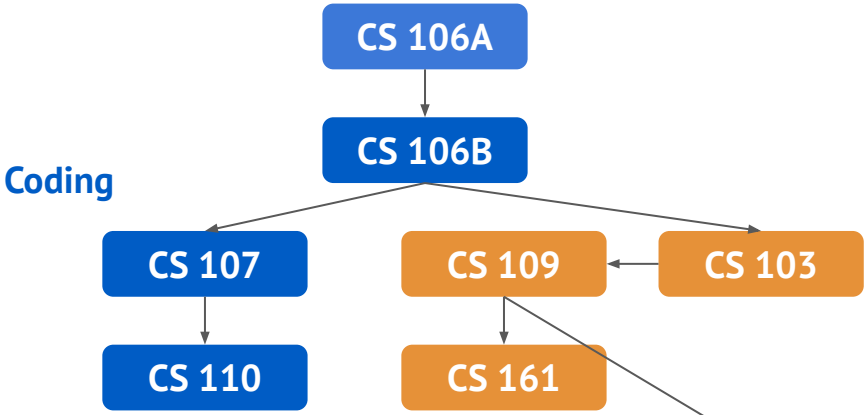
Systems


CS 106A
CS 106B
Coding
CS 107
CS 109
CS 103
Theory/math
CS 110
CS 161
CS 140
Operating Systems
CS 143
Compilers
CS 242
Programming Languages
CS 144
Networking
CS 243
Language Analysis
CS 149
Parallelism

AI


CS 106A
CS 106B
Coding
CS 107
CS 110
CS 109
CS 161
CS 103
Theory/math
CS 229
Machine Learning (mathy)
CS 124
NLP
CS 224N
Deep NLP
CS 231N
Deep Learning for Vision
CS 224W
ML on Graphs
CS 223A
Robots
CS 234
Reinforcement Learning

AI



Advanced courses:

- CS 229M: Statistical machine learning theory
- CS 329D: ML under distributional shift
- CS 468: ML in non-euclidean spaces

CS 229 Machine Learning (mathy)

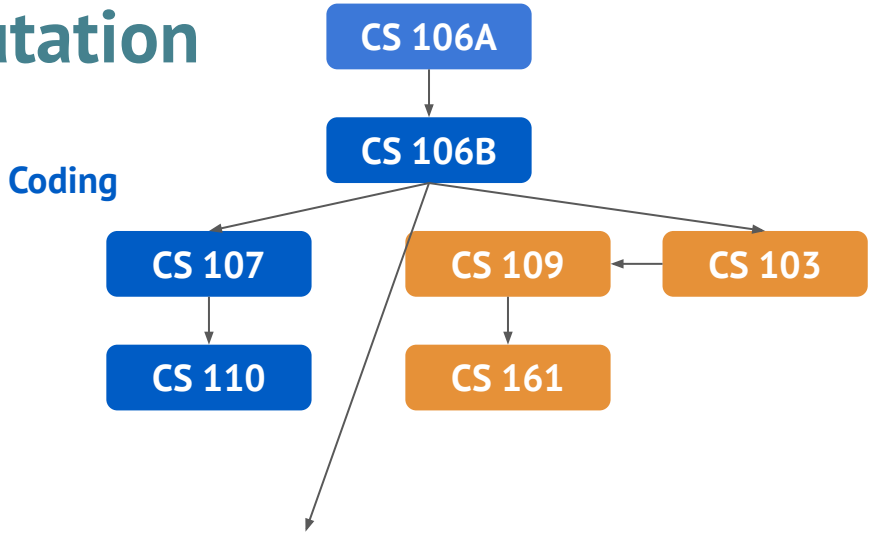
CS 124 NLP **CS 224N** Deep NLP

CS 231N Deep Learning for Vision

CS 224W ML on Graphs **CS 223A** Robots

CS 234 Reinforcement Learning

Biocomputation



CS 274 Representations and Algorithms for Computational Molecular Biology

CS 279 Biomolecules

CS 275 Translational Informatics

Other fields

- **Theory:** theory of computation, crypto, algorithm design
- **HCI:** interface design, Going Viral

Research: CURIS

- Research projects in all areas of CS!
- $7500

Foundation of Algorithmic Fairness

Professor	Omer Reingold
Fields	Theory of Computation
Quarter	Win_spr
Compensation	Paid or_credit

Tock: Secure Embedded Operating Systems Design in Rust

Professor	Philip A Levis
Fields	Operating Systems, Securi ...
Quarter	Aut_win_spr
Compensation	Paid or_credit

Medical imaging AI in COVID-19

Professor	Daniel Rubin
Fields	AI, Vision, Algorithms
Quarter	Aut_win
Compensation	Course credit