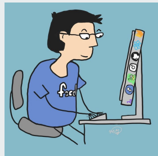
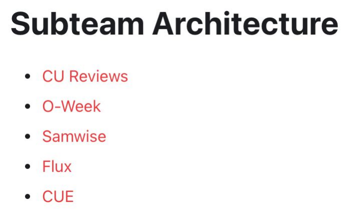
CoursePlan Requirement Computation

dev-sam



Some history

In 2019 FA, every old subteam present their architecture and interesting design decisions in DevSeshes.



Subteam Architecture

- CU Reviews
- O-Week
- Samwise
- Flux
- CUE

CoursePlan and Carriage were still new and their codebases have not stabilized yet…

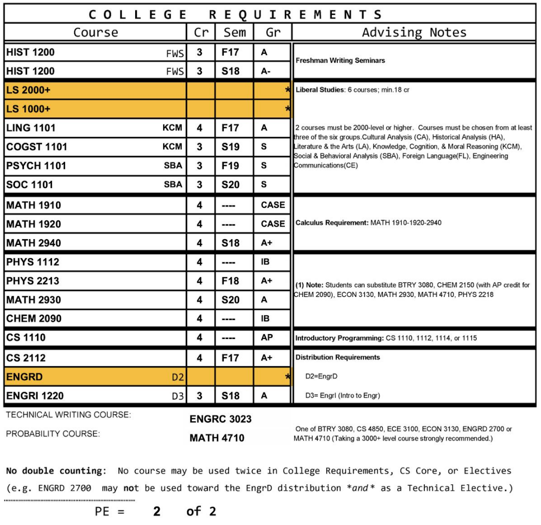
Now it's a good time to reflect on 1.5 years of CoursePlan development.

Why you should care about the topic?

It’s usually good to know what other subteams are doing, if you are considering switching teams.

The journey of designing CoursePlan requirement computation algorithm teaches you something important about system design.

Your imagination



C O L L E G E   R E Q U I R E M E N T S

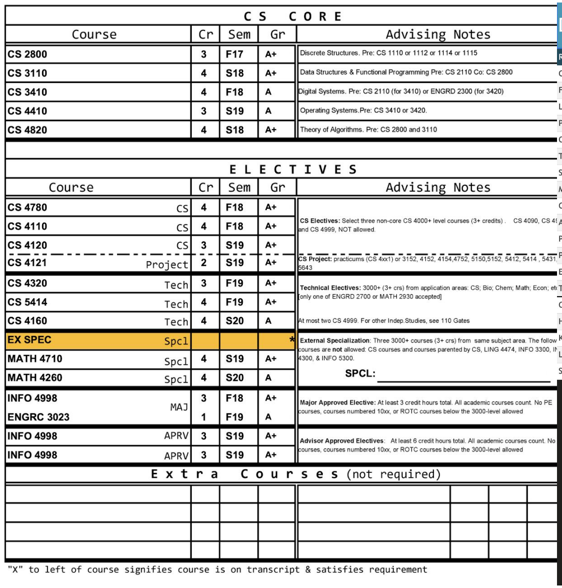
Course		Cr	Sem	Gr	Advising Notes
HIST 1200	FWS	3	F17	A	**Freshman Writing Seminars**
HIST 1200	FWS	3	S18	A-	
LS 2000+				*	**Liberal Studies**: 6 courses; min.18 cr
LS 1000+				*	
LING 1101	KCM	4	F17	A	2 courses must be 2000-level or higher. Courses must be chosen from at least three of the six groups.Cultural Analysis (CA), Historical Analysis (HA), Literature & the Arts (LA), Knowledge, Cognition, & Moral Reasoning (KCM), Social & Behavioral Analysis (SBA), Foreign Language(FL), Engineering Communications(CE)
COGST 1101	KCM	3	S19	S	
PSYCH 1101	SBA	3	F19	S	
SOC 1101	SBA	3	S20	S	
MATH 1910		4	----	CASE	**Calculus Requirement:** MATH 1910-1920-2940
MATH 1920		4	----	CASE	
MATH 2940		4	S18	A+	
PHYS 1112		4	----	IB	**(1) Note:** Students can substitute BTRY 3080, CHEM 2150 (with AP credit for CHEM 2090), ECON 3130, MATH 2930, MATH 4710, PHYS 2218
PHYS 2213		4	F18	A+	
MATH 2930		4	S20	A	
CHEM 2090		4	----	IB	
CS 1110		4	----	AP	**Introductory Programming:** CS 1110, 1112, 1114, or 1115
CS 2112		4	F17	A+	**Distribution Requirements**
ENGRD	D2			*	D2=EngrD
ENGRI 1220	D3	3	S18	A	D3= EngrI (Intro to Engr)

TECHNICAL WRITING COURSE: **ENGRC 3023**

PROBABILITY COURSE: **MATH 4710** One of BTRY 3080, CS 4850, ECE 3100, ECON 3130, ENGRD 2700 or MATH 4710 (Taking a 3000+ level course strongly recommended.)

No double counting: No course may be used twice in College Requirements, CS Core, or Electives (e.g. ENGRD 2700 may **not** be used toward the EngrD distribution **and* as a Technical Elective.)

PE = **2** **of 2**



C S   C O R E

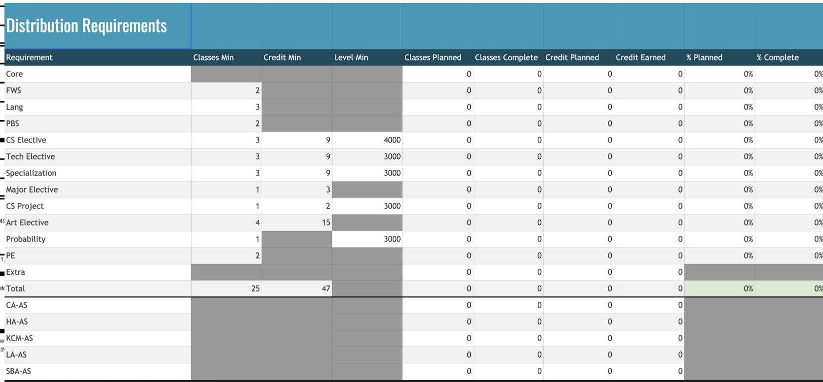
Course	Cr	Sem	Gr	Advising Notes
CS 2800	3	F17	A+	Discrete Structures. Pre: CS 1110 or 1112 or 1114 or 1115
CS 3110	4	S18	A+	Data Structures & Functional Programming Pre: CS 2110 Co: CS 2800
CS 3410	4	F18	A	Digital Systems. Pre: CS 2110 (for 3410) or ENGRD 2300 (for 3420)
CS 4410	3	S19	A	Operating Systems.Pre: CS 3410 or 3420.
CS 4820	4	S18	A+	Theory of Algorithms. Pre: CS 2800 and 3110

E L E C T I V E S

Course		Cr	Sem	Gr	Advising Notes
CS 4780	CS	4	F18	A+	**CS Electives:** Select three non-core CS 4000+ level courses (3+ credits) . CS 4090, CS 4 and CS 4999, NOT allowed.
CS 4110	CS	4	F18	A+	
CS 4120	CS	3	S19	A+	
CS 4121	Project	2	S19	A+	**CS Project:** practicums (CS 4xx1) or 3152, 4152, 4154,4752, 5150,5152, 5412, 5414 , 5431, 5643
CS 4320	Tech	3	F19	A+	**Technical Electives:** 3000+ (3+ crs) from application areas: CS; Bio; Chem; Math; Econ; et [only one of ENGRD 2700 or MATH 2930 accepted]
CS 5414	Tech	4	F19	A+	
CS 4160	Tech	4	S20	A	At most two CS 4999. For other Indep.Studies, see 110 Gates
EX SPEC	Spcl			*	**External Specialization**: Three 3000+ courses (3+ crs) from same subject area. The follow courses are **not** allowed: CS courses and courses parented by CS, LING 4474, INFO 3300, I 4300, & INFO 5300.
MATH 4710	Spcl	4	S19	A+	
MATH 4260	Spcl	4	S20	A	**SPCL:**
INFO 4998	MAJ	3	F18	A+	**Major Approved Elective:** At least 3 credit hours total. All academic courses count. No PE courses, courses numbered 10xx, or ROTC courses below the 3000-level allowed
ENGRC 3023		1	F19	A	
INFO 4998	APRV	3	S19	A+	**Advisor Approved Electives**: At least 6 credit hours total. All academic courses count. No courses, courses numbered 10xx, or ROTC courses below the 3000-level allowed
INFO 4998	APRV	3	S19	A+	

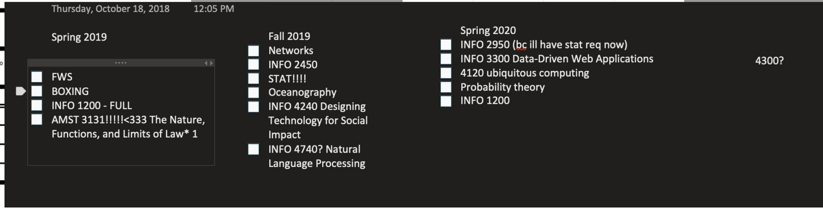
E x t r a   C o u r s e s (not required)

"X" to left of course signifies course is on transcript & satisfies requirement



Distribution Requirements

Requirement	Classes Min	Credit Min	Level Min	Classes Planned	Classes Complete	Credit Planned	Credit Earned	% Planned	% Complete
Core				0	0	0	0	0%	0%
FWS	2			0	0	0	0	0%	0%
Lang	3			0	0	0	0	0%	0%
PBS	2			0	0	0	0	0%	0%
CS Elective	3	9	4000	0	0	0	0	0%	0%
Tech Elective	3	9	3000	0	0	0	0	0%	0%
Specialization	3	9	3000	0	0	0	0	0%	0%
Major Elective	1	3		0	0	0	0	0%	0%
CS Project	1	2	3000	0	0	0	0	0%	0%
Art Elective	4	15		0	0	0	0	0%	0%
Probability	1		3000	0	0	0	0	0%	0%
PE	2			0	0	0	0	0%	0%
Extra				0	0	0	0		
Total	25	47		0	0	0	0	0%	0%
CA-AS				0	0	0	0		
HA-AS				0	0	0	0		
KCM-AS				0	0	0	0		
LA-AS				0	0	0	0		
SBA-AS				0	0	0	0		



Thursday, October 18, 2018   12:05 PM

Spring 2019

- ☐ FWS
- ☐ BOXING
- ☐ INFO 1200 - FULL
- ☐ AMST 3131!!!!!<333 The Nature, Functions, and Limits of Law* 1

Fall 2019

- ☐ Networks
- ☐ INFO 2450
- ☐ STAT!!!!
- ☐ Oceanography
- ☐ INFO 4240 Designing Technology for Social Impact
- ☐ INFO 4740? Natural Language Processing

Spring 2020

- ☐ INFO 2950 (bc ill have stat req now)
- ☐ INFO 3300 Data-Driven Web Applications
- ☐ 4120 ubiquitous computing
- ☐ Probability theory
- ☐ INFO 1200

4300?

The reality: problem statement

Given

- college, major, minor
- user provided courses
- AP/IB/Transfer credits

determine the requirement fulfillment progress.

```
function computeRequirementProgress(
  college: College,
  majors: readonly Major[],
  minors: readonly Minor[],
  coursesTaken: readonly Course[],
  examCredits: readonly ExamCredit[],
  transferClasses: readonly Course[],
): readonly FulfilmentProgress[] {
  // ...
}
```

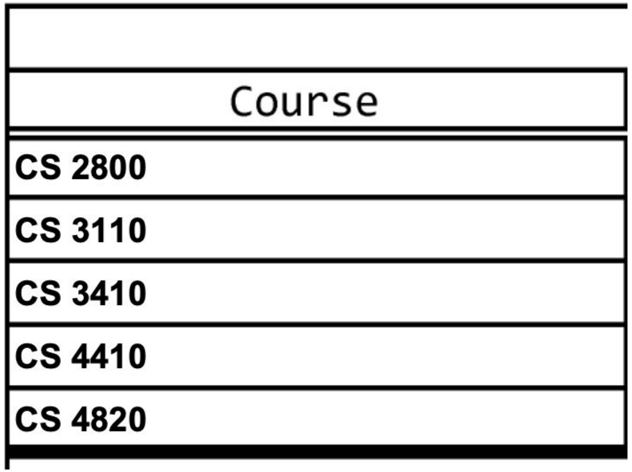
Act I: Start Simple

Observe the CS core requirements

Some requirements are easy to check:

Hardcode a list of classes that can satisfy it.

Actually we can even do better!

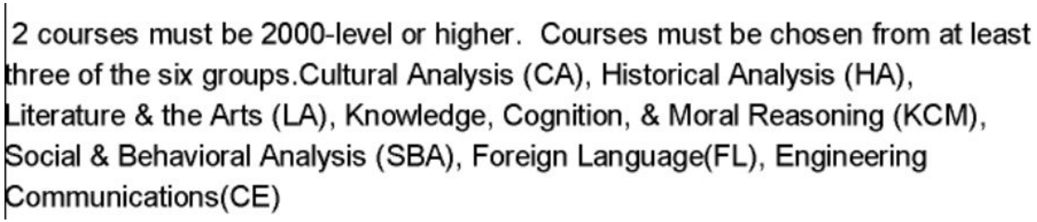


Course
CS 2800
CS 3110
CS 3410
CS 4410
CS 4820

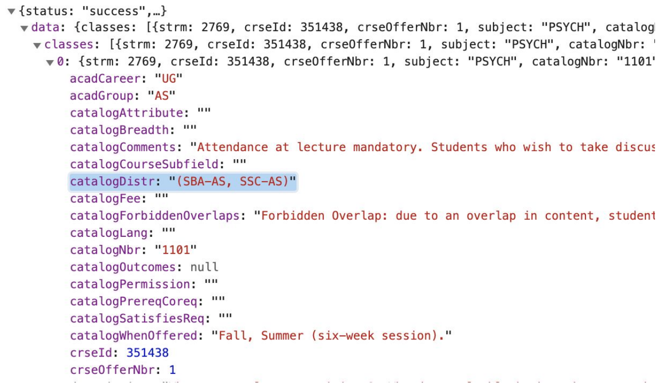
Include/Exclude Requirement JSON

```
"ENGL": {
    "name": "English",
    "schools": [
        "AS"
    ],
    "requirements": [
        {
            "name": "Total Credits",
            "description": "To graduate with a major in English, a student must complete with a grade of C or better 40 credit hours approved
            "source": "https://english.cornell.edu/majoring-and-minoring-english-cornell#requirements-for-the-major",
            "search": [
                "code"
            ],
            "includes": [
                [
                    "ENGL 2***",
                    "ENGL 3***",
                    "ENGL 4***",
                    "ENGL 5***",
                    "ENGL 6***"
                ]
            ],
            "excludes": [
                [
                    "ENGL 2800",
                    "ENGL 2810",
                    "ENGL 2880",
                    "ENGL 2890"
                ]
            ],
            "fulfilledBy": "credits",
            "minCount": 40
        },
```

Some requirements are not about class code!



2 courses must be 2000-level or higher. Courses must be chosen from at least three of the six groups.Cultural Analysis (CA), Historical Analysis (HA), Literature & the Arts (LA), Knowledge, Cognition, & Moral Reasoning (KCM), Social & Behavioral Analysis (SBA), Foreign Language(FL), Engineering Communications(CE)



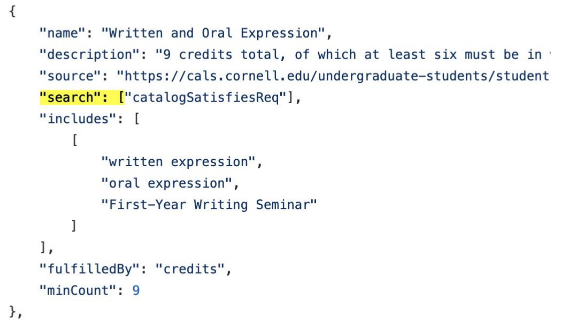
```
▼{status: "success",…}
  ▼data: {classes: [{strm: 2769, crseId: 351438, crseOfferNbr: 1, subject: "PSYCH", catalog
    ▼classes: [{strm: 2769, crseId: 351438, crseOfferNbr: 1, subject: "PSYCH", catalogNbr: '
      ▼0: {strm: 2769, crseId: 351438, crseOfferNbr: 1, subject: "PSYCH", catalogNbr: "1101'
          acadCareer: "UG"
          acadGroup: "AS"
          catalogAttribute: ""
          catalogBreadth: ""
          catalogComments: "Attendance at lecture mandatory. Students who wish to take discus
          catalogCourseSubfield: ""
          catalogDistr: "(SBA-AS, SSC-AS)"
          catalogFee: ""
          catalogForbiddenOverlaps: "Forbidden Overlap: due to an overlap in content, student
          catalogLang: ""
          catalogNbr: "1101"
          catalogOutcomes: null
          catalogPermission: ""
          catalogPrereqCoreq: ""
          catalogSatisfiesReq: ""
          catalogWhenOffered: "Fall, Summer (six-week session)."
          crseId: 351438
          crseOfferNbr: 1
```

Class Roster API to the rescue!

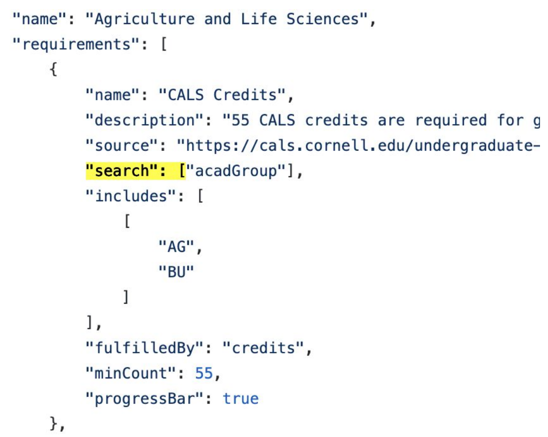
⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒

“search” field in requirement JSON

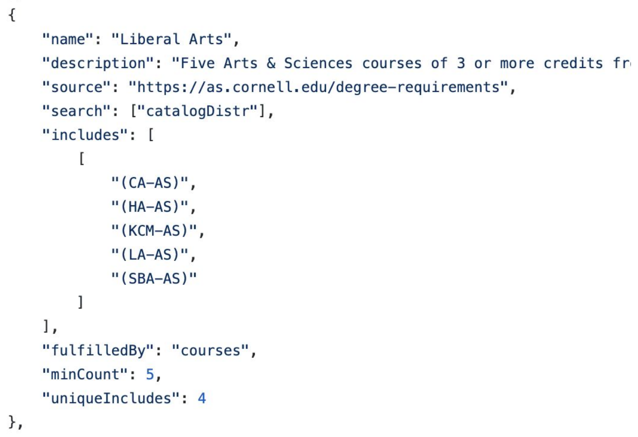
Direct us what kind of data to look for in the roster API.



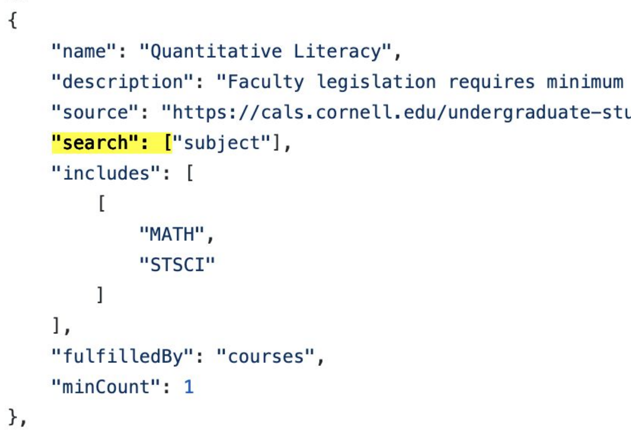
```
{
    "name": "Written and Oral Expression",
    "description": "9 credits total, of which at least six must be in
    "source": "https://cals.cornell.edu/undergraduate-students/student
    "search": ["catalogSatisfiesReq"],
    "includes": [
        [
            "written expression",
            "oral expression",
            "First-Year Writing Seminar"
        ]
    ],
    "fulfilledBy": "credits",
    "minCount": 9
},
```



```
"name": "Agriculture and Life Sciences",
"requirements": [
    {
        "name": "CALS Credits",
        "description": "55 CALS credits are required for g
        "source": "https://cals.cornell.edu/undergraduate-
        "search": ["acadGroup"],
        "includes": [
            [
                "AG",
                "BU"
            ]
        ],
        "fulfilledBy": "credits",
        "minCount": 55,
        "progressBar": true
    },
```



```
{
    "name": "Liberal Arts",
    "description": "Five Arts & Sciences courses of 3 or more credits fr
    "source": "https://as.cornell.edu/degree-requirements",
    "search": ["catalogDistr"],
    "includes": [
        [
            "(CA-AS)",
            "(HA-AS)",
            "(KCM-AS)",
            "(LA-AS)",
            "(SBA-AS)"
        ]
    ],
    "fulfilledBy": "courses",
    "minCount": 5,
    "uniqueIncludes": 4
},
```

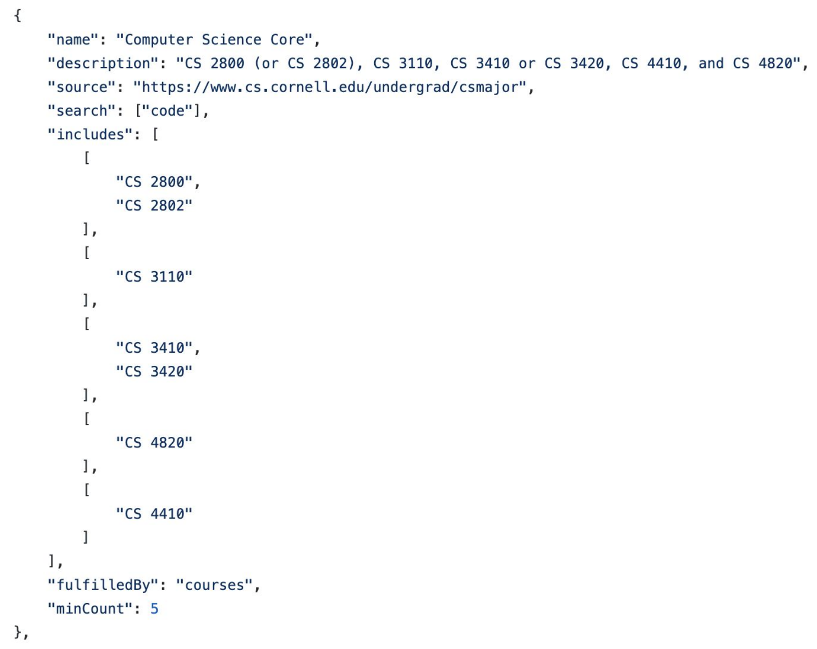


```
{
    "name": "Quantitative Literacy",
    "description": "Faculty legislation requires minimum
    "source": "https://cals.cornell.edu/undergraduate-stu
    "search": ["subject"],
    "includes": [
        [
            "MATH",
            "STSCI"
        ]
    ],
    "fulfilledBy": "courses",
    "minCount": 1
},
```

Sub-requirements

Some requirements have multiple options to fulfill them.

Without this sub-array, if you take CS 3410 and CS 3420, your progress will be 2/5 instead of 1/5.



```
{
    "name": "Computer Science Core",
    "description": "CS 2800 (or CS 2802), CS 3110, CS 3410 or CS 3420, CS 4410, and CS 4820",
    "source": "https://www.cs.cornell.edu/undergrad/csmajor",
    "search": ["code"],
    "includes": [
        [
            "CS 2800",
            "CS 2802"
        ],
        [
            "CS 3110"
        ],
        [
            "CS 3410",
            "CS 3420"
        ],
        [
            "CS 4820"
        ],
        [
            "CS 4410"
        ]
    ],
    "fulfilledBy": "courses",
    "minCount": 5
},
```

and we happily solved all problems ~~🎉~~

Sign-in Link

https://www.youtube.com/watch?v=HMhrRovP9qA

~~and we happily solved all problems 🎉~~

Fact check:
The above statement is COMPLETELY FALSE.

Act II: Oof

Oof 1: DDoS Roster API

Requirement checking happens on frontend.

To check whether a class satisfies a requirement (more complicated one like liberal studies), we fetch roster API.

With some additional bad code, we make O(m*n) calls to the roster API every time we check requirements.

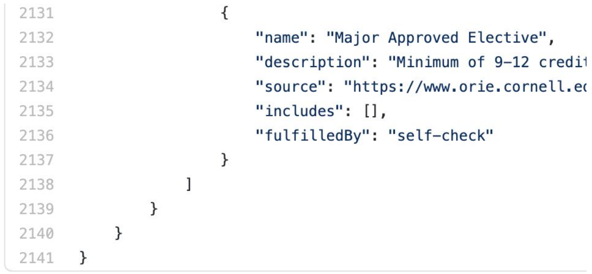
Oof 2: More complicated requirements

Some requirements have different sub-requirements depending on your strategy.

e.g. in Biological Science, you can do CHEM 2070 + 2080 or just CHEM 2150.

Refactoring it means changing the 2000+ json.

(BTW, JSON doesn't support comments)



```
                    {
                        "name": "Major Approved Elective",
                        "description": "Minimum of 9-12 credit
                        "source": "https://www.orie.cornell.ec
                        "includes": [],
                        "fulfilledBy": "self-check"
                    }
                ]
            }
        }
    }
```

Oof 3: No double counting detection

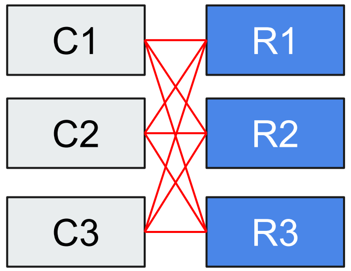
Many requirements can't be double counted, but some can.

If we falsely report some requirements are done but actually they are not, and caused students to pay tuition for an extra semester, then ...

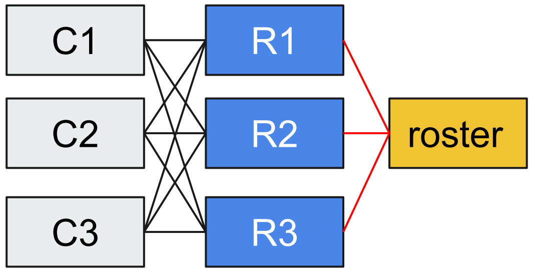
The infra is really not ready!

Act III: Pre-computation

We really need to prevent DDoS-ing roster API



Every pair of C-R check requires a fetch.
Can we do better?

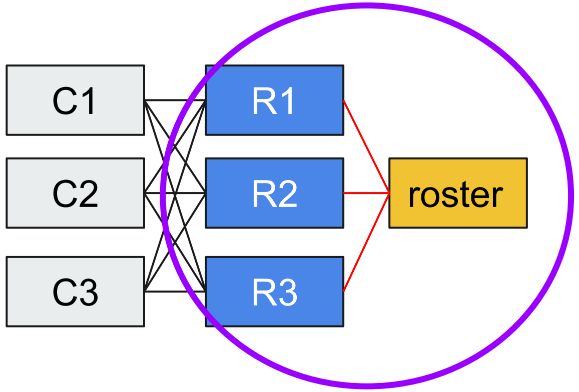


Total roster fetch: 9
Complexity: O(mn)

Total roster fetch: 3
Complexity: O(n)

Now we have O(n) fetches? Can be do better?

Look at this picture more closely 👀



- Fetching info from roster has nothing to do with user courses!
- We maintain a hardcoded list of requirements

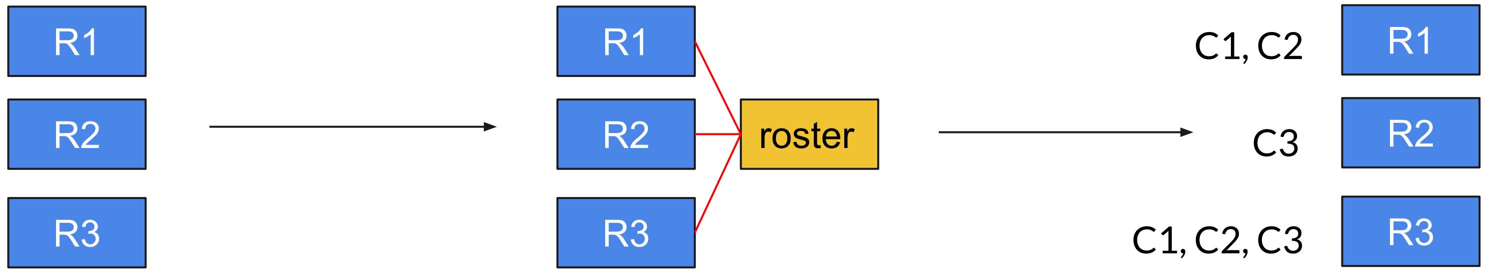
We should pre-fetch all the info from roster!

Problem: All the roster info combined is > 100MB.

Solution: we should pre-compute a list of satisfying courses!

Pre-computing satisfying course list

Fact: this computed courses list with requirements is less than 800K!



You start with requirements.

Using the requirement spec to fetch relevant data from roster

Using fetched data from roster to decide a list of satisfying courses.

Act IV: Scaling requirement specs

Storing the spec in JSON doesn’t scale

v1: initial

```
include: [“CS 211*”, “CS 3110”, ...], exclude: [...]
```

v2: lookup data with “search”

```
search: [“code”], include: [...], exclude: [...]

search: [“description”], include: [“history”]
```

V3: sub-requirements

```
include: [[“CS 211*”], [“CS 3110”], ...], exclude: [...]
```

v4: ??? 🤯

v5: ??? 💀

Observation:

As we gradually make the JSON more expressive, it becomes closer and closer to a programming language, where the code that processes the JSON acts as an interpreter

Think beyond the current codebase

Now we start to pre-compute the requirements.

All we care about is a list of courses that can satisfy a requirement. i.e.

```
function getSatisfyingCourses(requirement: Requirement): readonly Course[] { /* … */ }
```

The inner workings of how you produce such list is irrelevant.

Instead of recording "search", "includes", "excludes" and other helper data, we only record:

```
"checker": (course: Course): boolean { /* … */ }
```

Examples of pre-computation under new setup

```
{ name: “Intro to programming”, checker: (c) => [“CS 1110”, “CS 1112”].includes(c.code) }
⇒
{ name: “Intro to programming”, courses: [“CS 1110”, “CS 1112”] }


{ name: “FWS”, checker: (c) => c.category.includes(“FWS”) }
⇒
{ name: “FWS”, courses: [“HIST 1200”, “ENGL 1234”, ...] }
```

Now we have a much stabler requirement JSON interface for the rest of the system!

and we happily solved all problems ~~🎉~~

~~and we happily solved all problems 🎉~~

Fact check:
The above statement is *partially* false.

Act V: Scaling requirement checking

Recap

We avoided DDoS-ing Cornell Roster API

We make requirement specification much more principled and maintainable.

Huge success!

But we only solved the problem for half of the system: requirement data generation part.

The requirement checking part is still in a big mess.

Hard problems with requirement checking

- Detecting illegally double-counted courses
- Requirements with multiple fulfillment strategies
- AP/IB/Transfer Credits
- Crosslisted courses

A simple structure like `{“req1”: [“course1”, “course2”], “req2”: [“course2”], … }` doesn’t scale.

We need better abstractions!

Problem of double counting

Suppose we already build some requirement to course mapping like

```
{“req1”: [“course1”, “course2”], “req2”: [“course2”], … }
```

But imagine req1 and req2 don’t allow double-counting. So we need to report that course2 is illegally double counted.

Simple solution: build a reversed map!

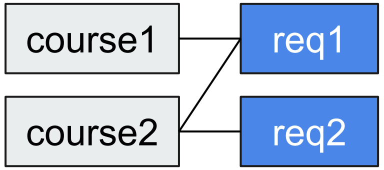
```
{“course1”: [“req1”], “course2”: [“req1”, “req2”], … }
```

Now we can clearly see course2 is double-counted!

Let's look more closely to this structure...

```
{"req1": ["course1", "course2"], "req2": ["course2"], … }

{"course1": ["req1"], "course2": ["req1", "req2"], … }
```



Does this remind you some data structure you learned in CS classes?

e.g. CS 2110/2

This is a graph! More specifically, a bipartite graph between requirements and courses!

Graph is a good abstraction for our problem

An edge between requirement R and course C means that C can be used to satisfy requirement R.

We can first build a coarse graph, and gradually refine it.

Setup

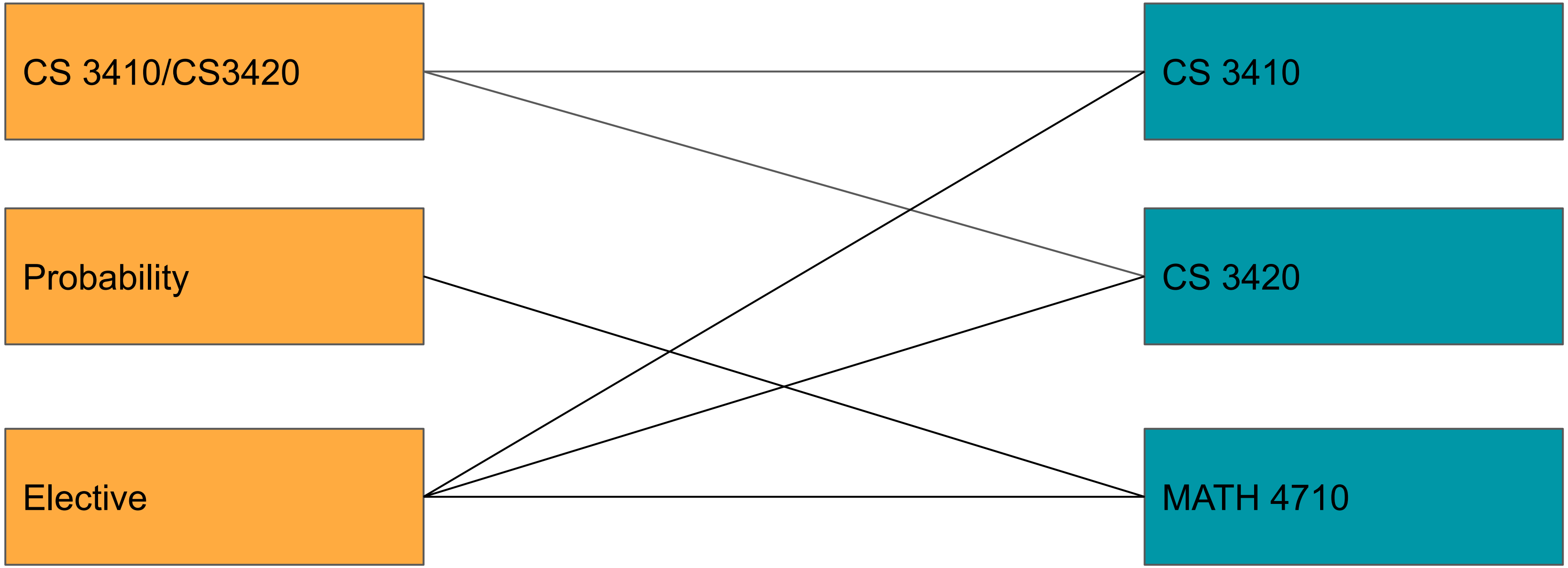
We have three requirements to consider:

- CS3410/CS3420, which can be satisfied by two strategies:
    - Strategy 1: [CS 3410]
    - Strategy 2: [CS 3420]
- Probability
    - Can be satisfied by MATH 4710
- Elective
    - Can be satisfied by everything
- NO DOUBLE COUNTING

User takes: CS 3410, CS 3420, MATH 4710

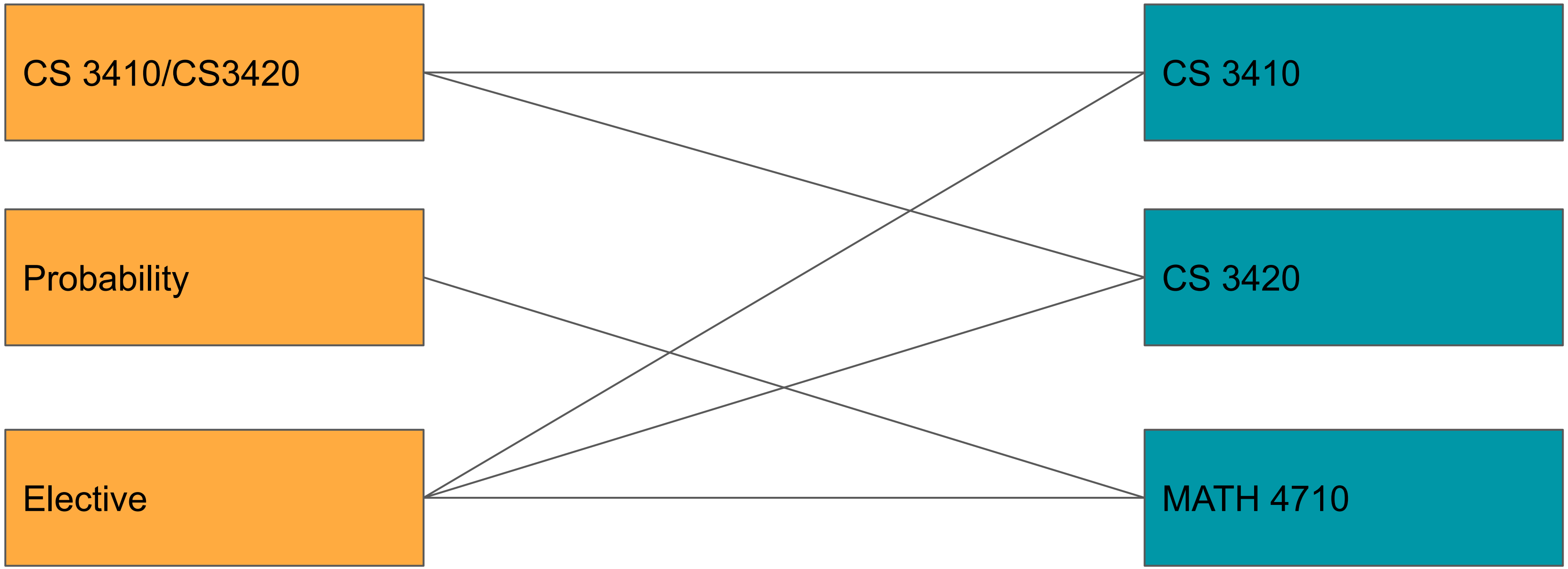
Example Walkthrough

Phase 1: Building the graph naively



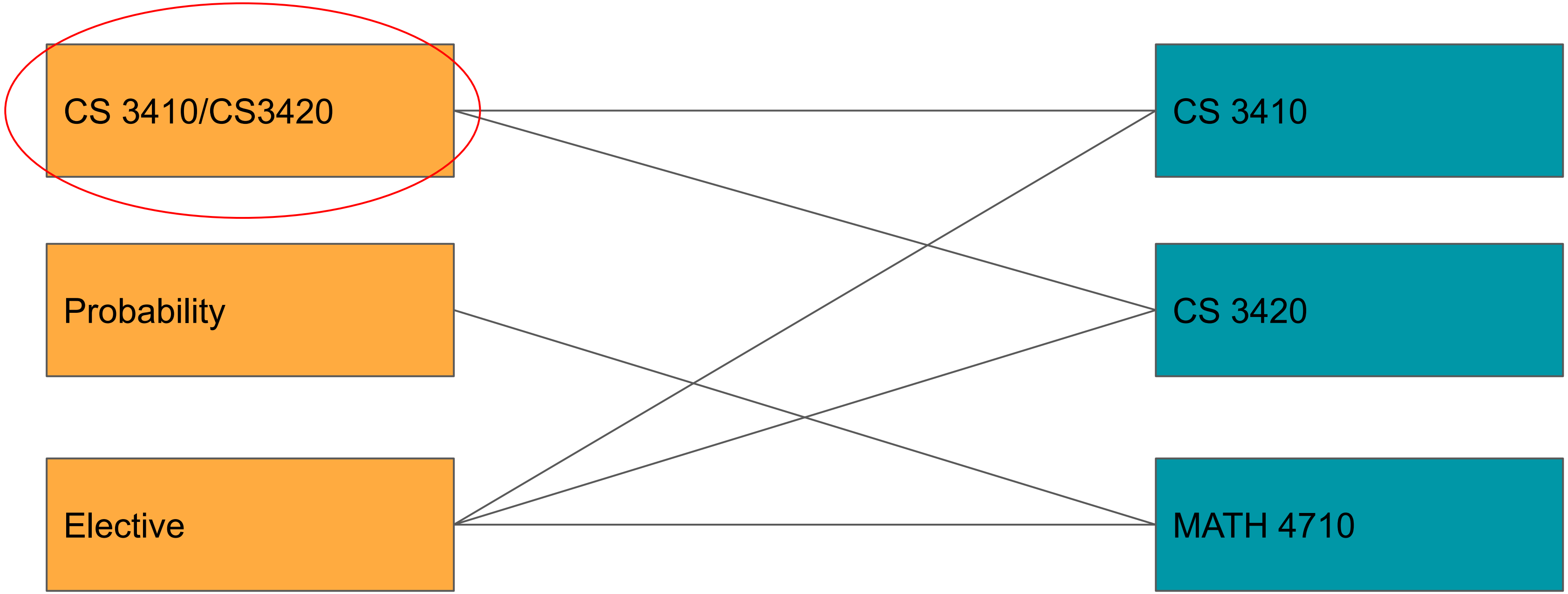
Now we completed phase 1

Phase 2: Consider User's Fulfillment Strategy Choice



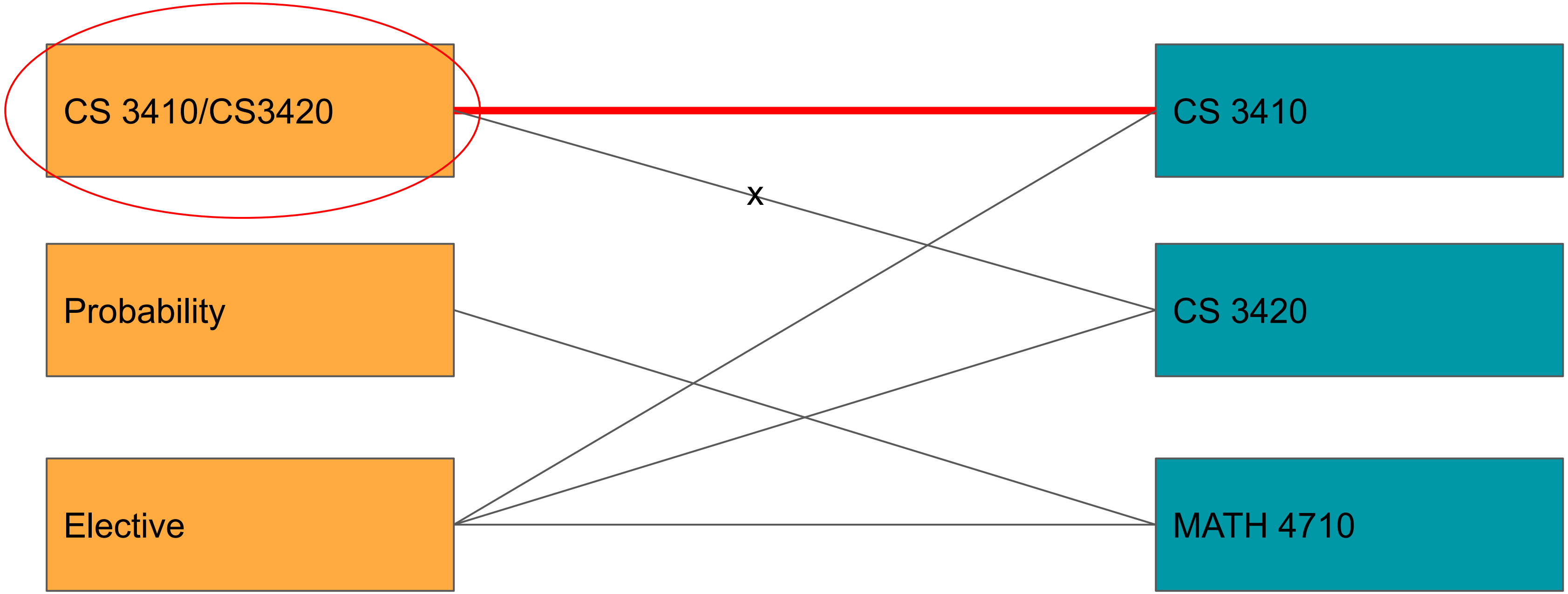
Suppose user chooses the [CS 3410] strategy for CS 3410/CS3420.

Phase 2: Consider User's Fulfillment Strategy Choice



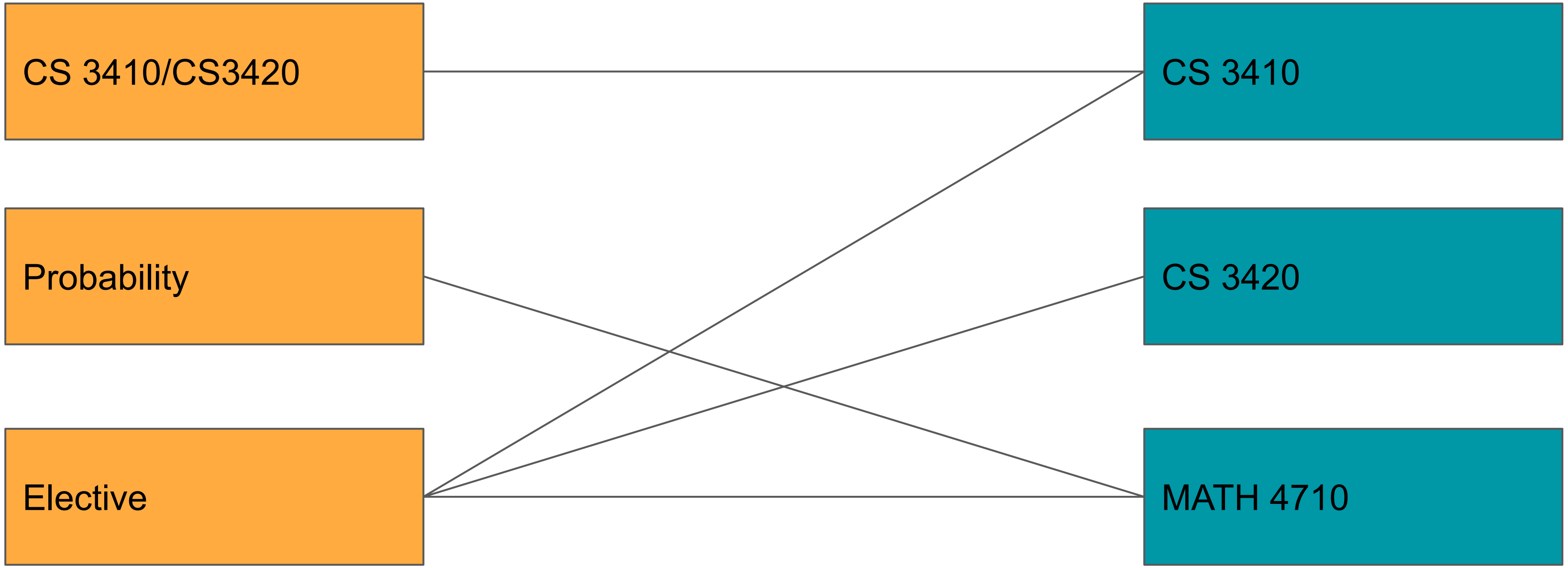
Suppose user chooses the [CS 3410] strategy for CS 3410/CS3420.

Phase 2: Consider User’s Fulfillment Strategy Choice



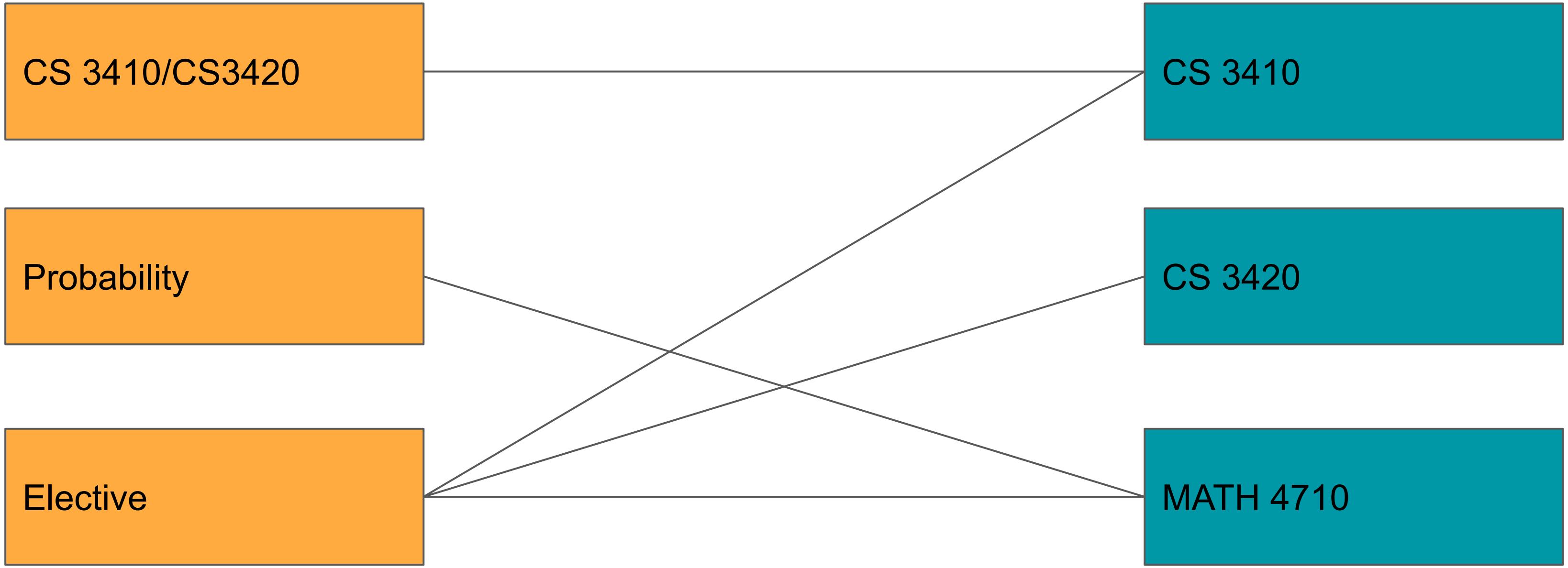
Suppose user chooses the [CS 3410] strategy for CS 3410/CS3420.

Phase 2: Consider User’s Fulfillment Strategy Choice



Now we complete phase 2.

Phase 3: Detect Illegal Double Counting



User's double-counting breaking choice:
(cs 3410, cs 3410/cs 3420)

Illegal double-counting
detected!

Act VI: Remaining Problems

Representing the graph

You might do something like this in Java:

```
class Graph {
  private final Map<Requirement, Set<Course>> req2CourseMap = new HashMap<>();
  private final Map<Course, Set<Requirement>> course2ReqMap = new HashMap<>();
}
```

Potential problem: how to implement `equals` and `hashCode` for course and requirements?

Aka: Define the the notion of equality between two requirements/courses.

Equality of requirements

Easy answer: every field of two requirement object must be completely equal.

Concern: expensive to check equality and compute hashCode, bad for performance.

Better idea: give every requirement an unique ID, and compare ID directly.

Equality of courses

Give every course an unique ID?

Actually course roster already has it for us.

What’s more, crosslisted courses share the same course ID!

If we use course ID from the roster, the problem of accounting for cross listed courses is automatically solved!

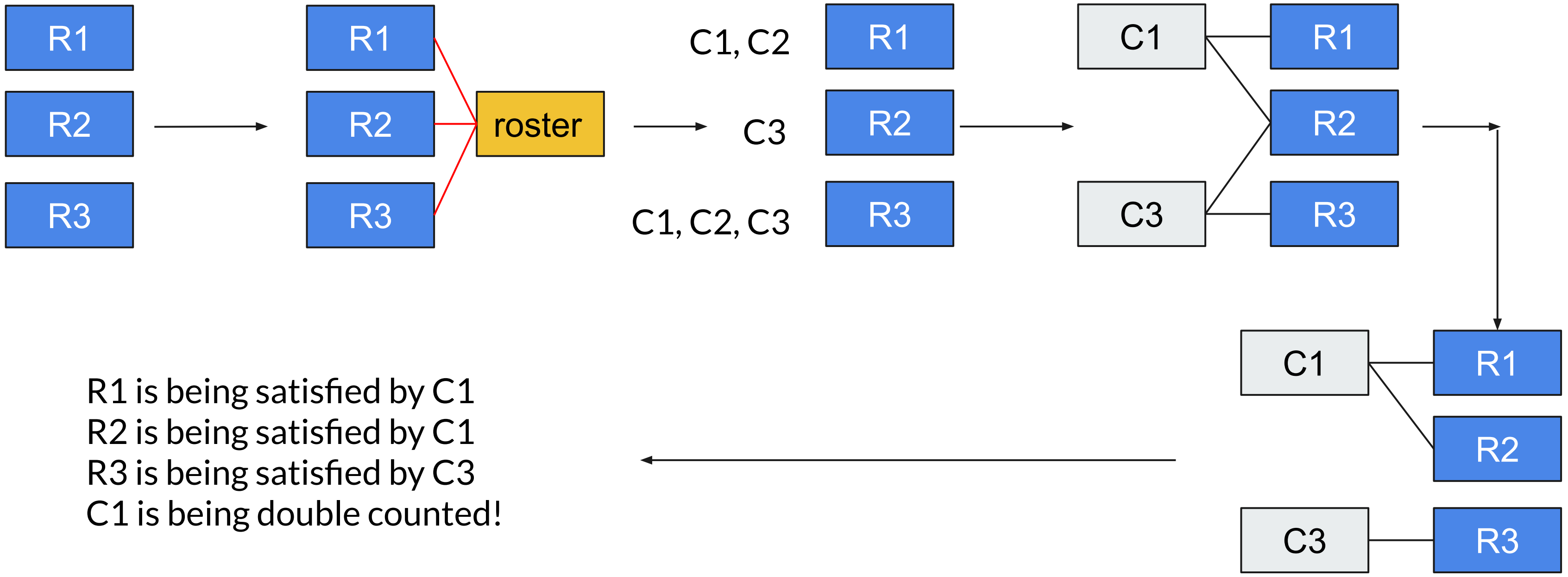
For AP/IB/Transfer credits, just generate a dummy course object with the same course ID as an equivalent course!

Recap

Recap in words

1. Write down the specification of each requirement using checker
2. Run the checkers on fetched roster data to get a list of satisfying courses
3. Use the satisfying courses data to build a initial coarse requirement fulfillment graph
4. Use user’s choices to refine the requirement fulfillment graph
5. Now we have all the requirement fulfillment status, and a list of double counted courses!

Recap in picture



R1 is being satisfied by C1
R2 is being satisfied by C1
R3 is being satisfied by C3
C1 is being double counted!

and we happily solved all problems ~~🎉~~

~~and we happily solved all problems 🎉~~

Fact check:
The above statement is **still** *partially* false.

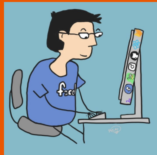
Unsolved problems

- How to check all aspects of liberal arts requirement in engineering
  - 18 credits, 6 courses, 3 categories
- How to handle information science concentration requirement
- How to make self-check requirements more useful
- etc

Most important takeaway

Don't just add more if-else branches to hack around problems.

Instead, find better abstractions!



Sign-in Link

https://forms.gle/8rdsQwT5yQLkWaKA8