CS/ECE/ISyE 524 — Introduction to Optimization — Spring 2021

Final Course Project: Due 5/2/21, 12:05pm

UW-Madison Underguade Course Planning

Shawn Zhong (shawn.zhong@wisc.edu (mailto:shawn.zhong@wisc.edu)), Evan Wang (xwang2488@wisc.edu (mailto:xwang2488@wisc.edu)), Jun Lin (tan65@wisc.edu (mailto:tan65@wisc.edu))

Table of Contents

1. Introduction
    A. Constraints
    B. Data
    C. Outline
2. Mathematical Model
    A. Assumptions
    B. Notations
    C. Decision Variables
    D. Objective
    E. Constraints
    F. User-Defined Constraints
    G. Standard Form
3. Solution
    A. Data Gathering
    B. Data Preprossing
    C. Decision Variables
    D. CS Major Requirements
    E. Math Major Requirements
    F. Course Prerequisites
    G. Other Constraints
    H. Putting Things Together
    I. Interactive Planning
4. Results and Discussion
    A. Results
        a. CS Major
        b. Math Major
        c. CS & Math Double Major
    B. Interactive Planning
        a. Desired Courses
        b. Transferred Courses

        c. Your Turn
    C. Discussion
5. Conclusion
6. Appendix
    A. Course Data Example
    B. Enrollment Prerequisites Example

1. Introduction

The topic that our team settled on is undergrad course planning at UW-Madison. Selecting the appropriate courses for every semester can be a challenging process for some students who aren't certain about the degree structure and numerous requirements.

Given that students picked their courses for every semester without considering future courses that have strict prerequisites, some of which are compulsory courses, it is very likely that the student will pick a non-optimum route for graduation. For example, a student will not be able to enroll in a compulsory course in a later semester if he/she hasn't taken the prerequisites to that course. As a result, he/she will have to enroll in the prerequisite courses before being able to enroll in the compulsory courses in subsequent semesters, which will delay the student's graduation date.

The goal of this project is to find an optimal course selection for each semester in regards to completing the graduation requirements within the shortest period and the least number of courses taken.

1.A Constraints

We consider the following constraints for course planning:

1. **Prerequisites**: Some courses are required to be completed before enrolling in certain classes
2. **Graduation requirements**: Some courses are required to be taken to meet the graduation requirements
3. **No retaking**: A course cannot be taken twice

We also have the following constraints that can be added interactively:

4. **Credit Limit**: The number of courses taken in any given semester cannot exceed a certain number of credit hours
5. **Desired courses**: Through the interactive console, the student can specify which courses must be included in the plan.
6. **Limit workload**: We also allow the student to cap the maximum credit.
7. **Transferred courses**: We allow the user to specify the transferred courses before entering the university, so our planning program can take that into account.

1.B Data

- The graduation requirements are gathered from the Online Underguade Guide (https://guide.wisc.edu/undergraduate/#majorscertificatestext).
- The course list and prerequisite relationships are obtained from the Course Search and Enroll website (https://public.enroll.wisc.edu/).

1.C Outline

The rest of the report is structured as follows: We first introduce the mathematical model for this problem, in this part, we will introduce the assumptions, notations, decision variables, objective, constraints, user-defined constraints, and standard form.

Next, we will introduce the solution we used for this model which includes how we gather data, how we process data, how to generate decision variables by codes, define the function of constraints to CS major requirements, define the function of constraints to Math major requirements, define the function of constraints to course prerequisites, define a function of constraints to other constraints, define a function with all constraint and optimal cost function, and code of interactive planning system for course planning at Wisconsin madison.

In the results and discussion part, we will show sample CS major, Math major, and double CS&Math major schedules. Also, for the interactive planning, we show two samples that will tell students how to use this interactive planning system and let students use it by themselves. We will show you some limitations of this system.

In the conclusion part, we will conclude what we did in this project, and show how should we improve this system in the future.

Appendix part shows all other information you needs to understand which includes course data example and enrollment prerequisites example.

2. Mathematical Model

2.A Assumptions

We made the following assumptions for our course planning problem:

1. We assume that there is no time conflict for class attendance in a given semester. One can think that all the courses can be taken asynchronously in this pandemic time. This assumption is made since we haven't figured out a way to gather section-level data, and it complicates the optimization problem.
2. We assume that the student can pass all the courses taken so there is no retaking.

2.B Notations

Variable	Description	Example
T	maximum number of semesters	For 4-year college, we have $T = 8$
$t \in \{0, \cdots, T\}$	a specific semester	$t = 2$ is the second semester $t = 0$ is used for transferred courses
C	the set of all classes	$C = \{\text{CS } 524, \text{MATH } 240, \cdots\}$
$c \in C$	a specific class	$c = \text{CS } 524$

2.C Decision Variables

$x[t, c] \in \{0, 1\}$ is a Boolean variable to denote whether to take the class $c \in C$ at semester $t \in \{0, \cdots, T\}$.

For example, if a student takes CS 524 on the thrid semester, then $x[3, \text{CS } 524] = 1$. If the student has CS 200 transferred, then $x[0, \text{CS } 200] = 1$

2.D Objective

A naive objective would be to minimize the sum of x:

$$\min \sum_{t=0}^{T} \sum_{c \in C} x[t, c]$$

But this would lead to many equally good solutions. To avoid this, we add a weight t to the class c if its taken at semester t

$$\min \sum_{t=0}^{T} \sum_{c \in C} x[t, c] \cdot t$$

This avoid students procrastinating class to later semesters.

2.E Constraints

Prerequisite

To take the class c, students may need to take c' in the previous semesters, we can encode such prerequisite using the following constraint:

$$x[t, c] \leq \sum_{i=0}^{t-1} x[i, c'] \quad \text{for all semesters } t \in \{1, \cdots, T\}$$

For example, CS 524 has the prerequisite similar to "(CS 200 or 300) and (MATH 340 or 341)", we can encode it with the constraint:

$$x[t, \text{CS 524}] \leq \sum_{i=0}^{t-1} x[i, \text{CS 200}] + \sum_{i=0}^{t-1} x[i, \text{CS 300}]$$

$$x[t, \text{CS 524}] \leq \sum_{i=0}^{t-1} x[i, \text{CS 340}] + \sum_{i=0}^{t-1} x[i, \text{CS 341}]$$

Graduation Requirement

In order to meet the graduation requirement, some class c must be taken, so we have the constraint:

$$\sum_{t=0}^{T} x[t, c] \geq 1$$

There are other kinds of graduation requirements, such as taking k coureses from a list of C, we can encode such requirements using the following constraint:

$$\sum_{t=0}^{T} \sum_{c \in C} x[t, c] \geq k$$

No Retaking

$$\sum_{t=1}^{T} x[t, c] \leq 1, \quad \text{for all classes } c \in C$$

2.F User-Defined Constraints

Max Workload

The user is able to set the desired maximum credit every semester

$$\sum_{c \in C} x[t, c] \cdot \text{credit}(c) \leq \text{max\_credit}[t], \quad \text{for all semesters } t \in \{1, \cdots, T\}, \text{classes } c \in C$$

Desired Courses

The user can specify a list of desired courses C_{desired}

$$\sum_{t=1}^{T} x[t, c] \geq 1 \quad \text{for all classes } c \in C_{\text{desired}}$$

Transferred Courses

The user is able to specify transferred courses C_{trans}

$$x[0, c] = 1, \quad \text{for all classes } c \in C_{\text{trans}}$$

$$x[0, c] = 0, \quad \text{for all classes } c \in C \setminus C_{\text{trans}}$$

2.F Standard Form

We model this problem as a mixed integer programming problem, and the standard form is shown below

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & \sum_{t=0}^{T} \sum_{c \in C} x[t, c] \cdot t \\
\text{subject to:} \quad & x[t, c] \leq \sum_{i=0}^{t-1} x[i, c'] && \text{for all classes } c \in C, \text{classes } c' \in \text{prerequisite}(c), \text{semesters } t \in \{1, \cdots, T\} \\
& \sum_{t=0}^{T} \sum_{c \in C_i} x[t, c] \geq k_i && \text{for all requirement set } C_i \text{required number of courses } k_i \\
& \sum_{t=0}^{T} x[t, c] \leq 1 && \text{for all class } c \in C \\
& \sum_{c \in C} x[t, c] \cdot \text{credit}(c) \leq \text{max\_credit}[t] && \text{for all semesters } t \in \{1, \cdots, T\}, \text{classes } c \in C \\
& \sum_{t=1}^{T} x[t, c] \geq 1 && \text{for all classes } c \in C_{\text{desired}} \\
& x[0, c] = 1 && \text{for all classes } c \in C_{\text{trans}} \\
& x[0, c] = 0 && \text{for all classes } c \in C \setminus C_{\text{trans}} \\
& x[t, c] \in \{0, 1\} && \text{for all semesters } t \in \{0, \cdots, T\}, \text{classes } c \in C
\end{aligned}
$$

3. Solution

3.A Data Gathering

We first collect the course data via the public-facing API from the [Course Search and Enroll website (https://public.enroll.wisc.edu/)](https://public.enroll.wisc.edu/).

For example, the following URL gives information about the courses provided in Spring 2021 (with term code 1214).

[https://public.enroll.wisc.edu/api/search/v1?query={"selectedTerm":"1214"} (https://public.enroll.wisc.edu/api/search/v1?query=%7B"selectedTerm":"1214"%7D)](https://public.enroll.wisc.edu/api/search/v1?query=%7B"selectedTerm":"1214"%7D)

We wrote a short script to download the data and saved the file to `data/1214-spring-2021.json` so that it can be later processed by Julia.

3.B Data Preprossing

We first use the `JSON` package to load the raw data into a Julia list.

In [1]:
```
using JSON

raw_data = JSON.parsefile("data/1214-spring-2021.json")["hits"];
print("There are ", length(raw_data), " items in raw_data")
```

```
There are 6510 items in raw_data
```

Each item in the `raw_data` array is a dictionary with the keys listed below. An example item in the array can be found in the Appendix.

In [3]:
```
join(keys(first(raw_data)), ", ")
```

Out[3]:
```
"honors, allCrossListedSubjects, breadths, matched_queries, termCode, levels, subjectAggregate, courseId, academicGroupCode, g
radingBasis, advisoryPrerequisites, ethnicStudies, lettersAndScienceCredits, approvedForTopics, courseDesignationRaw, openToFi
rstYear, gradCourseWork, catalogPrintFlag, sustainability, instructorProvidedContent, courseRequirements, subject, repeatable,
fullCourseDesignationRaw, typicallyOffered, foreignLanguage, enrollmentPrerequisites, titleSuggest, minimumCredits, title, las
tUpdated, workplaceExperience, courseDesignation, generalEd, topics, description, lastTaught, creditRange, catalogSort, curren
tlyTaught, firstTaught, catalogNumber, maximumCredits, fullCourseDesignation"
```

We notice that the same course can appear multiple times in the list `raw_data` since a course can be cross-listed in multiple departments, so we need to add another layer of indirection to use the course id instead of the course name as the unique identifier for a given course.

The mapping from the course name to the course id is saved in the dictionary `cls_name_to_id`, and the mapping from the id to the actual data is saved in `cls_dict`.

In [4]:
```
cls_dict = Dict(
    Symbol(c["courseId"]) => c
    for c in raw_data
)

cls_name_to_id = Dict(
    c["courseDesignation"] => Symbol(c["courseId"])
    for c in raw_data
)

for cls_name in ["COMP SCI 525", "I SY E 525", "MATH 525", "STAT 525"]
    println("The course id for \"", cls_name, "\" is ", cls_name_to_id[cls_name])
end
```

```
The course id for "COMP SCI 525" is 004272
The course id for "I SY E 525" is 004272
The course id for "MATH 525" is 004272
The course id for "STAT 525" is 004272
```

In [5]:
```
println("We have ", length(cls_dict), " courses in total")
```

```
We have 5611 courses in total
```

3.C Decision Variables

The decision variable is a Boolean matrix of shape 5611×8.

In [6]:
```
using JuMP, Gurobi

gurobi_env = Gurobi.Env()

C = keys(cls_dict)
T = 8

m = Model(with_optimizer(Gurobi.Optimizer, LogToConsole=0, gurobi_env))
@variable(m, x[C, T], Bin)
size(x)
```

```
Academic license - for non-commercial use only - expires 2021-06-19
```

Out[6]:
```
(5611, 1)
```

3.D CS Major Requirements

We use the CS major requirements from the following URL to construct the constraints.

https://guide.wisc.edu/undergraduate/letters-science/computer-sciences/computer-sciences-ba/#requirementstext (https://guide.wisc.edu/undergraduate/letters-science/computer-sciences/computer-sciences-ba/#requirementstext).

We note that all of the CS constraints are in the form of choosing k courses from a course set C'.

In [7]:
```
"""
Convert CS course number to course id
"""
get_cs_cls_id(number) = get(cls_name_to_id, "COMP SCI " * string(number), nothing)

"""
Convert a list of CS course numbers to a list of course ids
"""
cs_ids(arr...) = filter(id -> id in C, [get_cs_cls_id(e) for e in arr])


function add_cs_major_req()
    CS_basic = cs_ids(240, 252, 300, 354, 400)
    CS_theory = cs_ids(577, 520)
    CS_xware = cs_ids(
        407, 506, 536, 538, 537, 552, 564, 640, 642
    )
    CS_app = cs_ids(
        412, 425, 513, 514, 524, 525, 534, 540, 545, 547, 559, 570
    )
    CS_elec = cs_ids(
        407, 412, 425, 435, 471, 475, 506, 513, 514, 520, 524, 525, 526, 532,
        533, 534, 536, 537, 538, 539, 540, 545, 547, 552, 558, 559, 564, 567,
        570, 576, 577, 579, 635, 640, 642, 679, 639
    )


    # Take all from basic computer sciences
    for c in CS_basic
        @constraint(m, sum(x[c, t] for t in 0:T) >= 1)
    end

    # Complete 1 for Theory of computer science
    @constraint(m, sum(x[c, t] for t in 0:T for c in CS_theory) >= 1)

    # Complete 2 for Software & Hardware
    @constraint(m, sum(x[c, t] for t in 0:T for c in CS_xware) >= 2)

    # Complete 1 for Applications
    @constraint(m, sum(x[c, t] for t in 0:T for c in CS_app) >= 1)

    # Complete 2 for Electives
    @constraint(m, sum(x[c, t] for t in 0:T for c in CS_elec) >= 2)
end;
```

3.E Math Major Requirements

We use the Math major requirements from the following URL to construct the constraints.

https://guide.wisc.edu/undergraduate/letters-science/mathematics/mathematics-ba/mathematics-mathematics-programming-computing-ba/#requirementstext (https://guide.wisc.edu/undergraduate/letters-science/mathematics/mathematics-ba/mathematics-mathematics-programming-computing-ba/#requirementstext)

The constraints for Math major are more complicated, which includes the following forms:

- Choosing k courses from a course set C'
- c_1 and c_2 can be either both taken to count as satisfying one course in a course set C'
- Only one of the courses $c \in C'$ should be taken. If $c_1 \in C'$ has been taken before, then the student shouldn't take $c_2 \in C'$.

In [8]:

```
"""
Convert Math course number to course id
"""
get_math_cls_id(number) = get(cls_name_to_id, "MATH " * string(number), nothing)

"""
Convert a list of Math course numbers to a list of course ids
"""
math_ids(arr...) = filter(id -> id in C, [get_math_cls_id(e) for e in arr])

function add_math_major_req()
    MATH_linear_algebra = math_ids(320, 341, 340, 375)
    MATH_intermediate = math_ids(321, 322, 375, 421, 467)
    MATH_advanced = math_ids(514, 521, 531, 535, 540, 541, 571)
    MATH_elective_A = math_ids(
        513, 522, 525, 542, 567, 570,
        605, 619, 627, 629, 632, 635
    )
    MATH_elective_B = math_ids(
        310, 319, 376, 415, 425, 431, 309, 435, 443, 475
    )
    CS_basic = cs_ids(300, 400)
    CS_elective = cs_ids(
        412, 471, 520, 524, 526, 532, 533, 534, 538, 539,
        540, 545, 558, 559, 567, 576, 577, 635, 642
    )

    # Complete 1 for Linear Algebra
    @constraint(m, sum(x[c, t] for t in 0:T for c in MATH_linear_algebra) >= 1)

    # Complete 1 for Linear Algebra Intermediate Mathematics Requirement
    @constraint(m, sum(x[c, t] for t in 0:T for c in MATH_intermediate) >= 1)

    # Complete 1 for Advanced Mathematics Requirement
    @constraint(m, sum(x[c, t] for t in 0:T for c in MATH_advanced) >= 1)

    # MATH Elective to reach required 6 courses
    # Select one or more from MATH_elective_A
    @constraint(m, sum(x[c, t] for t in 0:T for c in MATH_elective_A) >= 1)
    # Select Select remaining courses from MATH_elective_B
    @constraint(m,
        sum(x[c, t] for t in 0:T for c in MATH_elective_A) +
        sum(x[c, t] for t in 0:T for c in MATH_elective_B) >= 6
    )

    # Programming and Computations Requirement at least 12 credit hours
    for c in CS_basic
        @constraint(m, sum(x[c, t] for t in 0:T) >= 1)
    end
    # CS_elective
    @constraint(m, sum(x[c, t] for t in 0:T for c in CS_elective) >= 2)
```

```
    # Additional requirements
    # 321 & 322 must be taken together
    @constraint(m,
        sum(x[get_math_cls_id(321), t] for t in 0:T) ==
        sum(x[get_math_cls_id(322), t] for t in 0:T)
    )

    # 319, 320, & 376 cannot be taken together
    @constraint(m,
        sum(x[c, t] for t in 0:T for c in math_ids(319, 320, 376)) <= 1
    )

    # 309 & 431 cannot be taken together
    @constraint(m,
        sum(x[c, t] for t in 0:T for c in math_ids(309, 431)) <= 1
    )
end;
```

3.F Course Prerequisites

Although we have the course prerequisites information in `raw_data`, it's in string format (see Appendix), so we need to manually encode this information.

`_add_prereq` function is a helper function for adding prerequisites. Thus function shouldn't use it directly, we can use it to make other functions: `add_cs_cs_prereq`, `add_cs_math_prereq`, `add_math_math_prereq`, to specify CS, Math class prerequisite for give CS or Math class.

`add_all_prereq` function using above three functions to add all course prerequisites as Math and CS department listed.

In [9]:
```
"""
A helper function for adding prerequisites

Don't use this function directly.
"""
function _add_prereq(id1, id2, cls, prereq)
    for t in 1:T
        prereq_id = filter(id -> id in C, [id2(p) for p in prereq])
        @constraint(m, x[id1(cls), t] <= sum(x[id, i] for i in 0:t-1 for id in prereq_id))
    end
end

"""
Specify CS class prerequisite for a given CS class
"""
add_cs_cs_prereq(cs_cls, one_of...) =
    _add_prereq(get_cs_cls_id, get_cs_cls_id, cs_cls, one_of)

"""
Specify Math class prerequisite for a given CS class
"""
add_cs_math_prereq(cs_cls, one_of...) =
    _add_prereq(get_cs_cls_id, get_math_cls_id, cs_cls, one_of)

"""
Specify Math class prerequisite for a given Math class
"""
add_math_math_prereq(math_cls, one_of...) =
    _add_prereq(get_math_cls_id, get_math_cls_id, math_cls, one_of)


"""
Add all course prerequisite
"""
function add_all_prereq()
    add_cs_cs_prereq(300, 200)
    add_cs_cs_prereq(354, 252)
    add_cs_cs_prereq(354, 300)

    add_cs_cs_prereq(506, 400)
    add_cs_cs_prereq(506, 407, 536, 537, 559, 564, 570, 679, 552)

    add_cs_cs_prereq(552, 352)
    add_cs_cs_prereq(552, 354)

    add_cs_cs_prereq(559, 400)

    for c in [400, 407, 412, 513, 514, 524, 532, 534, 539, 540, 570, 576]
        add_cs_cs_prereq(c, 300)
    end

    for c in [536, 537, 538, 558, 564]
```

```
    add_cs_cs_prereq(c, 354)
    add_cs_cs_prereq(c, 400)
end

for c in [520, 577]
    add_cs_cs_prereq(c, 400)
    add_cs_cs_prereq(c, 240, 475)
end

for c in [640, 642]
    add_cs_cs_prereq(c, 537)
end

for c in [435, 513, 524, 525]
    add_cs_math_prereq(c, 340, 341, 375)
end

for c in [412, 532, 576]
    add_cs_math_prereq(c, 222)
end

for c in [425, 475, 513, 533, 567]
    add_cs_math_prereq(c, 320, 340, 341, 375)
end

add_cs_math_prereq(412, 240, 234)
add_cs_math_prereq(435, 320)
add_cs_math_prereq(558, 234)




for c in [234, 310, 319, 320, 340]
    add_math_math_prereq(c, 222, 276)
end

for c in [309, 321, 341, 421, 431]
    add_math_math_prereq(c, 234, 376)
end

for c in [321, 443, 540]
   add_math_math_prereq(c, 319, 320, 340, 341, 375)
end


add_math_math_prereq(222, 221)
add_math_math_prereq(322, 321)
add_math_math_prereq(376, 375)


add_math_math_prereq(521, 234, 322, 341, 376, 421)
add_math_math_prereq(522, 521)
```

```
    add_math_math_prereq(531, 376, 421, 521)

    add_math_math_prereq(540, 234, 375)
    add_math_math_prereq(540, 341, 375, 421, 476, 521)

    add_math_math_prereq(541, 234, 375)
    add_math_math_prereq(541, 320, 341, 375, 421, 476, 521)

    add_math_math_prereq(542, 541)
    add_math_math_prereq(567, 541)

    add_math_math_prereq(619, 322, 421, 521)
    add_math_math_prereq(619, 319, 320, 376, 415, 519)

    add_math_math_prereq(629, 552)

    add_math_math_prereq(632, 431, 309, 531)
    add_math_math_prereq(632, 320, 340, 341, 375, 421, 531)
end;
```

3.G Other Constraints

We define the max credit constraint `add_max_credit_constraint`, no retaking constraint `add_no_retaking_constraint`, transferred courses constraint `add_transferred_courses_constraint`, and desired courses constraint in the following blocks `add_desired_courses_constraint`:

In [10]:
```
"""
Returns the minimum number of credit given a course id
"""
credit(c) = cls_dict[c]["minimumCredits"]

function add_max_credit_constraint(max_credit)
    for t in 1:length(max_credit)
        @constraint(m, sum(x[c, t] * credit(c) for c in C) <= max_credit[t])
    end
end;

function add_no_retaking_constraint()
    for c in C
        @constraint(m, sum(x[c, t] for t in 0:T) <= 1)
    end
end;

function add_transferred_courses_constraint(transferred_courses)
    for c in C
        if (c in transferred_courses)
            @constraint(m, x[c, 0] == 1)
        else
            @constraint(m, x[c, 0] == 0)
        end
    end
end;

function add_desired_courses_constraint(desired_courses)
    for c in desired_courses
        @constraint(m, sum(x[c, t] for t in 0:T) >= 1)
    end

end;
```

3.H Putting Things Together

We define the `find_optimal_schedule` function below to solve for the optimal schedule given the constraints above.

There are six arguments in this function: `num_semester` number of semester, `max_credit` max credit hours, `math_major` does the student in math major, `cs_major` does the student in cs major, `transferred_courses` the courses student want to transfer, `desired_courses` and the course student.

We can change the value of `num_semester` or `max_credit` as we wish to find a different strategy for generating different course plans. If the student in math major, and the `math_major` we are set as true, same as a student in cs_major. For cs and math double major students, they can set both `math_major` and `cs_major` arguments as true.

In [11]:
```
function find_optimal_schedule(;
        num_semester=8,
        max_credit=6,
        math_major=false,
        cs_major=false,
        transferred_courses=[],
        desired_courses=[]
    )
    global T = num_semester
    global m = Model(with_optimizer(Gurobi.Optimizer, LogToConsole=0, gurobi_env))
    @variable(m, x[C, 0:T], Bin)

    if isa(max_credit, Number)
        max_credit = repeat([max_credit], num_semester)
    end
    add_max_credit_constraint(max_credit)

    add_transferred_courses_constraint(transferred_courses)
    add_desired_courses_constraint(desired_courses)
    add_no_retaking_constraint()
    add_all_prereq()

    if math_major
        add_math_major_req()
    end

    if cs_major
        add_cs_major_req()
    end

    @objective(m, Min, sum(x[c, t] * t for c in C, t in 1:T))

    optimize!(m)
    global x = x
end;
```

3.I Interactive Planning

We allow users to change the following constraints for interactive planning:

Desired courses

Users will be able to choose which are the courses they want to be included in the course schedule. Note that all the prerequisites of the desired courses will be accounted for as well.

Transferred courses

Users will be able to specify which are the courses they already completed before enrolling in the university. Note that only courses without any prerequisites will be allowed. This is based on the assumption that transfer courses are elementary-level courses.

Max workload on each semester

Users can specify the maximum number of credits they want to take on a given semester between 0 and 22. Note that a maximum credit load of "0" would mean the user wants to take a gap semester

Change number of semester

The user can modify the total number of semesters. The default number of the semester is 8. Changes on the maximum credit load each semester made through "Max workload on each semester" will be preserved. If the new specified number of the semester is greater than the current value, the maximum credit loads of the additional semester will be set to the default number, 9.

Code

Here are the functions we will use for this interactive planning. All user input are passed through a robust error checking script to check for invalid inputs, which will cause difficult to trace errors later on if not properly dealt with

In [18]:
```
"""
Prompt user to enter a list of courses
Returns the ids for the courses entered
"""
function prompt_courses()
    println("Please enter a list of courses seperated by comma:")
    println("eg: \"COMP SCI 525, MATH 320, MUSIC 113\"")
    course_ids = []

    flush(stdout)
    line = readline()
    course_names = split(line, ",")

    for course_name in course_names
        course_name = uppercase(strip(course_name))
        course_id = get(cls_name_to_id, course_name, nothing)
        if(course_id == nothing)
            println("Course not found:", course_name)
            continue
        else
            push!(course_ids, course_id)
        end
    end

    return course_ids
end;
```

In [19]:

```
"""
Prase int from a string
"""
function parse_int(s)
    try
        return parse(Int64, s)
    catch e
        println("Invalid number format: ", s)
        return nothing
    end
end;


function prompt_max_work_load(max_work_load)
    println("Enter maximum Y credits for X semester, default workload is 9 credits per semester")
    println("format:\n eg: \"2, 6\" means maximum of 6 credits in semester 2")
    flush(stdout)
    response = readline()

    if(length(split(response, ',')) != 2)
        println("Invalid format")
        return
    end

    semester, max_credit = split(response, ',')

    semester = parse_int(semester)
    max_credit = parse_int(max_credit)
    if semester == nothing || max_credit == nothing
        return
    end

    if semester > num_semester
        println("Invalid semester number: semester input is greater than number of semester")
    elseif (max_credit < 0) || (max_credit > 22)
        println("Invalid credit amount, please enter a number between 0 and 22")
    else
        max_work_load[semester] = max_credit
        println("Maximum workload on semester: ", semester," successfully set to: ", max_credit)
    end
end;
```

In [20]:
```
function prompt_num_semester(old_num_semester)
    println("Enter number of semester")
    flush(stdout)
    line = readline()
    num_semester = parse_int(line)
    if num_semester == nothing
        return old_num_semester
    elseif num_semester < 4
        println("You're not a genius, lets be real here. number of semester is still: ", old_num_semester)
        return old_num_semester
    elseif num_semester > 12
        println(
            "You're not seriosly gonna spend more than 6 years in undergrad are you?",
            "number of semester is still: ", old_num_semester
        )
        return old_num_semester
    else
        println("number of semester successfully set to: ", num_semester)
        return num_semester
    end
end;
```

In [21]:
```
MAJOR_PROMPT = "
What major do you like to take? (enter number)
    1. CS
    2. Math
    3. Math + CS"

function prompt_major()
    cs_major = false
    math_major = false

    println(MAJOR_PROMPT)
    flush(stdout)
    response = readline()
    if response == "1"
        cs_major = true
    elseif response == "2"
        math_major = true
    elseif response == "3"
        cs_major = true
        math_major = true
    else
        println("Invalid response: ", response)
    end

    return cs_major, math_major
end;
```

In [22]:
```
MODIFY_PROMPT = "
Do you want to modify your schedule? (enter number)
    1. Add desired courses
    2. Add prior transferred courses
    3. Limit workload on each semester
    4. Change number of semester, default is 8
    5. Exit"

function interactive_planning()
    desired_courses = []
    transferred_courses = []
    num_semester = 8
    max_work_load = [9 for i in 1: num_semester]
    cs_major, math_major = prompt_major()

    while(true)
        find_optimal_schedule(
            max_credit=max_work_load,
            cs_major=cs_major,
            math_major=math_major,
            desired_courses=desired_courses,
            transferred_courses=transferred_courses,
            num_semester=num_semester
        )

        println("==================================================")
        print_result()
        println("==================================================")

        println(MODIFY_PROMPT)
        flush(stdout)
        response = readline()

        if (response == "1")
            desired_courses = prompt_courses()
            println("The desired course list is: ", get_cls_names(desired_courses))
        elseif(response == "2")
            transferred_courses = prompt_courses()
            println("The transferred course list is: ", get_cls_names(transferred_courses))
        elseif(response == "3")
            prompt_max_work_load(max_work_load)
        elseif(response == "4")
            num_semester = prompt_num_semester(num_semester)
            if(length(max_work_load) > num_semester)
                max_work_load = max_work_load[1: num_semester]
            else
                max_work_load = vcat(max_work_load, [9 for i in 1: num_semester - length(max_work_load)])
            end
            println("max_work_load after changing number of semester: ", max_work_load)
        elseif(response == "5")
            break
        else
```

```
            println("Invalid response: ", response)
        end
    end
end;
```

4. Results and Discussion

4.A Results

We first define some helpful functions to print out the results from the optimization problem.

The `get_cls_name` function is a function that returns the name for a course by given course `id`. The `get_cls_names` function use the `get_cls_name` function to return the maes of a list of courses. The `print_result` function can be used to print the schedule we want to show.

In [23]:
```
function get_cls_name(id)
    cls = cls_dict[id]
    subjects = [
        e["shortDescription"]
        for e in cls["allCrossListedSubjects"]
    ]
    if length(subjects) == 0
        return cls["courseDesignation"]
    else
        return join(subjects, "/") * " " * cls["catalogNumber"]
    end
end


get_cls_names(ids) = join([get_cls_name(id) for id in ids], ", ")


function print_result()
    if termination_status(m) != MOI.OPTIMAL
        println(termination_status(m))
        return
    end

    x = value.(m[:x])
    n_cls = 0
    total_credit = 0
    for t in 1:T
        for c in C
            if x[c, t] > 0
                name = get_cls_name(c) * ": " * cls_dict[c]["title"]
                if(t == 0)
                    println("Transfered: \"", name, "\"")
                else
                    println("Semester ", t, ": take \"", name, "\"")

                    n_cls += 1
                    total_credit += credit(c)
                end

            end
        end
    end
    printstyled(stdout, "Summary: ", n_cls, " courses taken with ", total_credit, " credits", bold=true)
    println()
end;
```

4.A.a CS Major

Here is the schedule we generated if a student wants to take a CS major:

In [85]:
```
find_optimal_schedule(max_credit=6, cs_major=true)
print_result()
```

```
Semester 1: take "COMP SCI/MATH 240: Introduction to Discrete Mathematics"
Semester 1: take "COMP SCI 200: Programming I"
Semester 2: take "COMP SCI/E C E 252: Introduction to Computer Engineering"
Semester 2: take "COMP SCI 300: Programming II"
Semester 3: take "COMP SCI/E C E 354: Machine Organization and Programming"
Semester 3: take "COMP SCI 400: Programming III"
Semester 4: take "COMP SCI 520: Introduction to Theory of Computing"
Semester 4: take "COMP SCI 538: Introduction to the Theory and Design of Programming Languages"
Semester 5: take "COMP SCI 536: Introduction to Programming Languages and Compilers"
Semester 5: take "COMP SCI 534: Computational Photography"
Summary: 10 courses taken with 29 credits
```

As we can see, if the student wants to take a CS major and only take 6 credits per semester, then the student needs a total of 5 semesters and 10 courses to finish the major in 5 semesters.

4.A.b Math Major

This time, we set the same max credit as 6 credit hours per semester and try to find the schedule for the student who wants to do a Math major:

In [86]:
```
find_optimal_schedule(max_credit=6, math_major=true)
print_result()
```

```
INFEASIBLE_OR_UNBOUNDED
```

As we can see, unlike the CS major, we are unable to find a schedule for Math major if the student only takes 6 credits per semester. We can change the `max_credit` value to make it feasible.

We can choose to increase the max credit limit from 6 to 9:

In [87]:
```
find_optimal_schedule(max_credit=9, math_major=true)
print_result()
```

```
Semester 1: take "COMP SCI 200: Programming I"
Semester 1: take "MATH 221: Calculus and Analytic Geometry 1"
Semester 2: take "MATH 222: Calculus and Analytic Geometry 2"
Semester 2: take "COMP SCI 300: Programming II"
Semester 3: take "MATH/STAT 310: Introduction to Probability and Mathematical Statistics II"
Semester 3: take "MATH 340: Elementary Matrix and Linear Algebra"
Semester 3: take "COMP SCI 400: Programming III"
Semester 4: take "COMP SCI/I SY E/MATH 425: Introduction to Combinatorial Optimization"
Semester 4: take "COMP SCI/MATH 514: Numerical Analysis"
Semester 4: take "COMP SCI/MATH/STAT 475: Introduction to Combinatorics"
Semester 5: take "COMP SCI/E C E/M E 532: Matrix Methods in Machine Learning"
Semester 5: take "COMP SCI/I SY E/MATH/STAT 525: Linear Optimization"
Semester 5: take "MATH 443: Applied Linear Algebra"
Semester 6: take "MATH 234: Calculus--Functions of Several Variables"
Semester 6: take "COMP SCI/MATH 513: Numerical Linear Algebra"
Semester 7: take "MATH 421: The Theory of Single Variable Calculus"
Semester 7: take "COMP SCI 577: Introduction to Algorithms"
Summary: 17 courses taken with 56 credits
```

As we can see, this time we set the max_credit as 9, then this optimal will be feasible. The student can take 17 courses (56 credit hours) to finish the major in 7 semesters.

Or we can generate a schedule that takes 10 semesters:

In [88]:
```
find_optimal_schedule(max_credit=6, math_major=true, num_semester=10)
print_result()
```

```
Semester 1: take "MATH 221: Calculus and Analytic Geometry 1"
Semester 2: take "MATH 222: Calculus and Analytic Geometry 2"
Semester 3: take "MATH 319: Techniques in Ordinary Differential Equations"
Semester 3: take "MATH 340: Elementary Matrix and Linear Algebra"
Semester 4: take "MATH/STAT 310: Introduction to Probability and Mathematical Statistics II"
Semester 4: take "COMP SCI/E C E 533: Image Processing"
Semester 5: take "COMP SCI/I SY E/MATH/STAT 525: Linear Optimization"
Semester 5: take "MATH 443: Applied Linear Algebra"
Semester 6: take "COMP SCI 200: Programming I"
Semester 6: take "COMP SCI/MATH/STAT 475: Introduction to Combinatorics"
Semester 7: take "COMP SCI/I SY E/MATH 425: Introduction to Combinatorial Optimization"
Semester 7: take "COMP SCI 300: Programming II"
Semester 8: take "COMP SCI 534: Computational Photography"
Semester 8: take "COMP SCI 400: Programming III"
Semester 9: take "MATH 234: Calculus--Functions of Several Variables"
Semester 10: take "MATH 421: The Theory of Single Variable Calculus"
Semester 10: take "MATH 521: Analysis I"
Summary: 17 courses taken with 55 credits
```

As we can see, the student can still finish the math major (optimal feasible), but this time, the student needs 10 semesters to finish the major as he wishes. The student can take fewer courses per semester than the previous schedule to reach the course requirements.

4.A.c CS & Math Double Major

This time, suppose the student want to take CS & Math double major, and per semester he wants to take a max 9 credit hours and finish the double major in 8 semesters:

In [89]:
```
find_optimal_schedule(max_credit=9, cs_major=true, math_major=true, num_semester=8)
print_result()
```

```
Semester 1: take "COMP SCI 200: Programming I"
Semester 1: take "MATH 221: Calculus and Analytic Geometry 1"
Semester 2: take "MATH 222: Calculus and Analytic Geometry 2"
Semester 2: take "COMP SCI/E C E 252: Introduction to Computer Engineering"
Semester 2: take "COMP SCI 300: Programming II"
Semester 3: take "COMP SCI/MATH 514: Numerical Analysis"
Semester 3: take "MATH 340: Elementary Matrix and Linear Algebra"
Semester 3: take "COMP SCI 400: Programming III"
Semester 4: take "MATH 319: Techniques in Ordinary Differential Equations"
Semester 4: take "COMP SCI/MATH 240: Introduction to Discrete Mathematics"
Semester 4: take "MATH 443: Applied Linear Algebra"
Semester 5: take "COMP SCI/I SY E/MATH 425: Introduction to Combinatorial Optimization"
Semester 5: take "COMP SCI/E C E 354: Machine Organization and Programming"
Semester 5: take "COMP SCI/MATH/STAT 475: Introduction to Combinatorics"
Semester 6: take "COMP SCI 536: Introduction to Programming Languages and Compilers"
Semester 6: take "MATH/STAT 310: Introduction to Probability and Mathematical Statistics II"
Semester 6: take "COMP SCI/I SY E/MATH/STAT 525: Linear Optimization"
Semester 7: take "MATH 234: Calculus--Functions of Several Variables"
Semester 7: take "COMP SCI 520: Introduction to Theory of Computing"
Semester 8: take "MATH 421: The Theory of Single Variable Calculus"
Semester 8: take "COMP SCI 538: Introduction to the Theory and Design of Programming Languages"
Summary: 21 courses taken with 66 credits
```

As we can see, with CS & Math double major, only 21 courses are taken instead of 17 + 10 = 27 courses, and the number of credits is reduced as well.

4.B Interactive Planning

4.B.a Desired Courses

Let's suppose that a student wants to take `MATH 320` before graduation. From the following output, we can see that, in addition to `MATH 320`, its prerequisites `MATH 221` and `MATH 222` are added to the schedule as well.

In [94]:
```
interactive_planning()
```

```
What major do you like to take? (enter number)
1. CS
2. Math
3. Math + CS

stdin> 1
================================================
Semester 1: take "COMP SCI/MATH 240: Introduction to Discrete Mathematics"
Semester 1: take "COMP SCI 200: Programming I"
Semester 1: take "COMP SCI/E C E 252: Introduction to Computer Engineering"
Semester 2: take "COMP SCI 300: Programming II"
Semester 3: take "COMP SCI/E C E 354: Machine Organization and Programming"
Semester 3: take "COMP SCI 400: Programming III"
Semester 3: take "COMP SCI 407: Foundations of Mobile Systems and Applications"
Semester 4: take "COMP SCI/E C E 506: Software Engineering"
Semester 4: take "COMP SCI 534: Computational Photography"
Semester 4: take "COMP SCI 520: Introduction to Theory of Computing"
Summary: 10 courses taken with 29 credits
================================================

Do you want to modify your schedule? (enter number)
    1. Add desired courses
    2. Add prior transferred courses
    3. Limit workload on each semester
    4. Change number of semester, default is 8
    5. Exit
stdin> 1
Please enter a list of courses seperated by comma:
eg: "COMP SCI 525, MATH 320, MUSIC 113"
stdin> MATH 320
The desired course list is: MATH 320
================================================
Semester 1: take "COMP SCI 200: Programming I"
Semester 1: take "MATH 221: Calculus and Analytic Geometry 1"
Semester 2: take "MATH 222: Calculus and Analytic Geometry 2"
Semester 2: take "COMP SCI/E C E 252: Introduction to Computer Engineering"
Semester 2: take "COMP SCI 300: Programming II"
Semester 3: take "COMP SCI/E C E 354: Machine Organization and Programming"
Semester 3: take "COMP SCI 534: Computational Photography"
Semester 3: take "MATH 320: Linear Algebra and Differential Equations"
Semester 4: take "COMP SCI/MATH 240: Introduction to Discrete Mathematics"
Semester 4: take "COMP SCI 400: Programming III"
Semester 4: take "COMP SCI 407: Foundations of Mobile Systems and Applications"
Semester 5: take "COMP SCI 520: Introduction to Theory of Computing"
Semester 5: take "COMP SCI 538: Introduction to the Theory and Design of Programming Languages"
Summary: 13 courses taken with 41 credits
================================================

Do you want to modify your schedule? (enter number)
```

```
    1. Add desired courses
    2. Add prior transferred courses
    3. Limit workload on each semester
    4. Change number of semester, default is 8
    5. Exit
stdin> 5
```

4.B.b Transferred Courses

For the Math major, if the student already 2 intermediate-level courses (i.e., `COMP SCI 300`, `MATH 234`) transferred, we can see that 5 courses are removed from the schedule, including the prerequisites of the intermediate-level courses: `COMP SCI 300`, `MATH 221`, and `MATH 222`.

In [99]:
```
interactive_planning()
```

```
What major do you like to take? (enter number)
1. CS
2. Math
3. Math + CS

stdin> 2
================================================
Semester 1: take "COMP SCI 200: Programming I"
Semester 1: take "MATH 221: Calculus and Analytic Geometry 1"
Semester 2: take "MATH 222: Calculus and Analytic Geometry 2"
Semester 2: take "COMP SCI 300: Programming II"
Semester 3: take "MATH/STAT 310: Introduction to Probability and Mathematical Statistics II"
Semester 3: take "MATH 340: Elementary Matrix and Linear Algebra"
Semester 3: take "COMP SCI 400: Programming III"
Semester 4: take "COMP SCI/I SY E/MATH 425: Introduction to Combinatorial Optimization"
Semester 4: take "COMP SCI/MATH 514: Numerical Analysis"
Semester 4: take "COMP SCI/MATH/STAT 475: Introduction to Combinatorics"
Semester 5: take "COMP SCI/E C E/M E 532: Matrix Methods in Machine Learning"
Semester 5: take "COMP SCI/I SY E/MATH/STAT 525: Linear Optimization"
Semester 5: take "MATH 443: Applied Linear Algebra"
Semester 6: take "MATH 234: Calculus--Functions of Several Variables"
Semester 6: take "COMP SCI/MATH 513: Numerical Linear Algebra"
Semester 7: take "MATH 421: The Theory of Single Variable Calculus"
Semester 7: take "COMP SCI 577: Introduction to Algorithms"
Summary: 17 courses taken with 56 credits
================================================

Do you want to modify your schedule? (enter number)
    1. Add desired courses
    2. Add prior transferred courses
    3. Limit workload on each semester
    4. Change number of semester, default is 8
    5. Exit
stdin> 2
Please enter a list of courses seperated by comma:
eg: "COMP SCI 525, MATH 320, MUSIC 113"
stdin> COMP SCI 300, MATH 234
The transferred course list is: COMP SCI 300, MATH 234
================================================
Semester 1: take "MATH/STAT 309: Introduction to Probability and Mathematical Statistics I"
Semester 1: take "COMP SCI 400: Programming III"
Semester 1: take "MATH 341: Linear Algebra"
Semester 2: take "COMP SCI/MATH/STAT 475: Introduction to Combinatorics"
Semester 2: take "COMP SCI/I SY E/MATH/STAT 525: Linear Optimization"
Semester 2: take "MATH 443: Applied Linear Algebra"
Semester 3: take "COMP SCI/MATH 514: Numerical Analysis"
Semester 3: take "COMP SCI 540: Introduction to Artificial Intelligence"
Semester 3: take "MATH 421: The Theory of Single Variable Calculus"
Semester 4: take "COMP SCI/I SY E/MATH 425: Introduction to Combinatorial Optimization"
```

```
Semester 4: take "COMP SCI 534: Computational Photography"
Semester 4: take "COMP SCI/MATH 513: Numerical Linear Algebra"
Summary: 12 courses taken with 36 credits
================================================

Do you want to modify your schedule? (enter number)
    1. Add desired courses
    2. Add prior transferred courses
    3. Limit workload on each semester
    4. Change number of semester, default is 8
    5. Exit
stdin> 5
```

4.B.c Your Turn

Feel free to play with the `interactive_planning` function here!

In [100]:
```
interactive_planning()
```

```
What major do you like to take? (enter number)
1. CS
2. Math
3. Math + CS

stdin> 1
================================================
Semester 1: take "COMP SCI/MATH 240: Introduction to Discrete Mathematics"
Semester 1: take "COMP SCI 200: Programming I"
Semester 1: take "COMP SCI/E C E 252: Introduction to Computer Engineering"
Semester 2: take "COMP SCI 300: Programming II"
Semester 3: take "COMP SCI/E C E 354: Machine Organization and Programming"
Semester 3: take "COMP SCI 400: Programming III"
Semester 3: take "COMP SCI 407: Foundations of Mobile Systems and Applications"
Semester 4: take "COMP SCI/E C E 506: Software Engineering"
Semester 4: take "COMP SCI 534: Computational Photography"
Semester 4: take "COMP SCI 520: Introduction to Theory of Computing"
Summary: 10 courses taken with 29 credits
================================================

Do you want to modify your schedule? (enter number)
    1. Add desired courses
    2. Add prior transferred courses
    3. Limit workload on each semester
    4. Change number of semester, default is 8
    5. Exit
stdin> 5
```

4.C Discussion

In these two parts of the results, we showed you the regular schedule results and interactive schedule results. Currently, we only have two major courses as options for students. This is one limitation of our project. Also, we don't add general courses and graduate-level courses in our project, thus, most 1st year, 2nd-year students, and graduate students cannot use this system to generate their schedule. Furthermore, if we want the student to use this system, we need to consider an infeasible optimal problem under some conditions shows above, a manual for this system is needed. However, as our current approach, these two problems can be solved easily, we only need to manually add general courses and other majors in our code for student to choose. Otherwise, this undergrad course planning system is very useful for students to use at Wisconsin Madison.

5. Conclusion

For our undergrad course planning system at UW-Madison. We successfully make a system for Math and CS major students to generate their course schedules. We also make an interactive planning system for student use. We add most main constraints for course selection in our optimal system to make sure this system can be used practically to match department requirements. In the result, we showed the students both Math and CS major or double Math & CS major course schedule. We also showed the students two examples for how to use an interactive planning system, and the students can use these two examples as a reference to make their own choice.

There are several limitations in our undergrad course planning system. We only have two major courses as options for students. We don't add general courses and graduate-level courses in our project, thus, most of the 1st year, 2nd year students, and graduate students cannot use this system to generate their schedules. If we want students to use this system, we need to consider an infeasible optimal problem under some conditions shows above, a manual for this system is needed.

In the future, we can add more courses and majors for the students to choose from, and we can include general courses and graduate-level courses for 1st year and 2nd-year students. Also, we will add a manual for the students to understand infeasible optimal situations and make them can use this system easily, to avoid infeasible optimal situations.

6. Appendix

6.A Course Data Example

In [20]:
```
show(IOContext(stdout, :limit => false), "text/plain", cls_dict[get_cs_cls_id(524)]);
```

```
Dict{String,Any} with 44 entries:
  "honors" => nothing
  "allCrossListedSubjects" => Any[Dict{String,Any}("departmentURI"=>"http://www.cs.wisc.edu/","footnotes"=>Any["Courses taught
and managed by the Computer Sciences department often have enrollment restrictions that give students in UW-Madison Computer S
ciences programs priority access during initial enrollment periods.\n\nEvening exams are likely for most of our undergraduate
courses."],"formalDescription"=>"COMPUTER SCIENCES","undergraduateCatalogURI"=>"http://guide.wisc.edu/undergraduate/letters-sc
ience/computer-sciences/","termCode"=>"1214","departmentOwnerAcademicOrgCode"=>"L0780","description"=>"COMPUTER SCIENCES","gra
duateCatalogURI"=>"http://guide.wisc.edu/graduate/computer-sciences/","uddsFundingSource"=>"A4820","shortDescription"=>"COMP S
CI","subjectCode"=>"266","schoolCollege"=>Dict{String,Any}("schoolCollegeURI"=>"http://www.ls.wisc.edu/","shortDescription"=
>"Letters and Science","formalDescription"=>"Letters and Science, College of","academicOrgCode"=>"L","uddsCode"=>nothing,"acad
emicGroupCode"=>"L&S")), Dict{String,Any}("departmentURI"=>"http://www.engr.wisc.edu/department/electrical-computer-engineerin
g/","footnotes"=>Any["Due to capacity limits the department cannot guarantee enrollment in any ECE courses even for ECE major
s. When necessary, enrollment priority for students registering on schedule will be given to: 1) EE & CMPE majors, ECE graduat
e students & AMEP program students; 2) students admitted to another engineering major or PhD minor.  Evening exams may be sche
duled for all courses.\n\nFor enrollment questions, please email: ece-enrollment@engr.wisc.edu."],"formalDescription"=>"ELECTR
ICAL AND COMPUTER ENGINEERING","undergraduateCatalogURI"=>"http://guide.wisc.edu/undergraduate/engineering/electrical-computer
-engineering/","termCode"=>"1214","departmentOwnerAcademicOrgCode"=>"E0480","description"=>"ELECTRICAL AND COMPUTER ENGR","gra
duateCatalogURI"=>"http://guide.wisc.edu/graduate/electrical-computer-engineering/","uddsFundingSource"=>"A1925","shortDescrip
tion"=>"E C E","subjectCode"=>"320","schoolCollege"=>Dict{String,Any}("schoolCollegeURI"=>"http://www.engr.wisc.edu/","shortDe
scription"=>"Engineering","formalDescription"=>"Engineering, College of","academicOrgCode"=>"E","uddsCode"=>nothing,"academicG
roupCode"=>"EGR")), Dict{String,Any}("departmentURI"=>"http://www.engr.wisc.edu/department/industrial-systems-engineering/","f
ootnotes"=>Any["Jeff Linderoth , Chair, 3270 Mech Engr, (608) 262-9660.\n\nEnrollment in ISyE classes:  \nPlease read the cour
se notes for enrollment details and restrictions. Most ISyE classes are restricted to ISyE undergrad or grad students until Ja
nuary 11th at noon.  \n\nNon-ISyE students: After this date, for authorization or enrollment problems contact enrollment@ie.wi
sc.edu\n\nISyE students: for pre-requisites or enrollment problems contact enrollment@ie.wisc.edu\n\nOnline demo on how to joi
n the waitlist:   http://registrar.wisc.edu/isis_helpdocs/enrollment_demos/V90WaitList/V90WaitList.htm"],"formalDescription"=
>"INDUSTRIAL (http://registrar.wisc.edu/isis_helpdocs/enrollment_demos/V90WaitList/V90WaitList.htm"],"formalDescription"=>"IND
USTRIAL) AND SYSTEMS ENGINEERING","undergraduateCatalogURI"=>"http://guide.wisc.edu/undergraduate/engineering/industrial-syste
ms-engineering/","termCode"=>"1214","departmentOwnerAcademicOrgCode"=>"E0525","description"=>"INDUSTRIAL & SYSTEMS ENGR","grad
uateCatalogURI"=>"http://guide.wisc.edu/graduate/industrial-systems-engineering/","uddsFundingSource"=>"A1950","shortDescripti
on"=>"I SY E","subjectCode"=>"490","schoolCollege"=>Dict{String,Any}("schoolCollegeURI"=>"http://www.engr.wisc.edu/","shortDes
cription"=>"Engineering","formalDescription"=>"Engineering, College of","academicOrgCode"=>"E","uddsCode"=>nothing,"academicGr
oupCode"=>"EGR"))]
  "breadths" => Any[Dict{String,Any}("code"=>"N","description"=>"Natural Science")]
  "matched_queries" => nothing
  "termCode" => "1214"
  "levels" => Any[Dict{String,Any}("code"=>"I","description"=>"Intermediate")]
  "subjectAggregate" => "INDUSTRIAL & SYSTEMS ENGR 490"
  "courseId" => "024408"
  "academicGroupCode" => nothing
  "gradingBasis" => Dict{String,Any}("code"=>"OPT","description"=>"Student Option")
  "advisoryPrerequisites" => nothing
  "ethnicStudies" => nothing
  "lettersAndScienceCredits" => Dict{String,Any}("code"=>"C","description"=>"Counts as LAS credit (L&S)")
  "approvedForTopics" => false
  "courseDesignationRaw" => "I SY E 524"
  "openToFirstYear" => false
  "gradCourseWork" => nothing
  "catalogPrintFlag" => true
  "sustainability" => nothing
```

```
  "instructorProvidedContent" => nothing
  "courseRequirements" => Dict{String,Any}("015976="=>Any[51010, 33566])
  "subject" => Dict{String,Any}("departmentURI"=>"http://www.engr.wisc.edu/department/industrial-systems-engineering/","footnotes"=>Any["Jeff Linderoth , Chair, 3270 Mech Engr, (608) 262-9660.\n\nEnrollment in ISyE classes:  \nPlease read the course notes for enrollment details and restrictions. Most ISyE classes are restricted to ISyE undergrad or grad students until January 11th at noon.  \n\nNon-ISyE students: After this date, for authorization or enrollment problems contact enrollment@ie.wisc.edu \n\nISyE students: for pre-requisites or enrollment problems contact enrollment@ie.wisc.edu\n\nOnline demo on how to join the waitlist:  http://registrar.wisc.edu/isis_helpdocs/enrollment_demos/V90WaitList/V90WaitList.htm"],"formalDescription"=>"INDUSTRIAL (http://registrar.wisc.edu/isis_helpdocs/enrollment_demos/V90WaitList/V90WaitList.htm"],"formalDescription"=>"INDUSTRIAL) AND SYSTEMS ENGINEERING","undergraduateCatalogURI"=>"http://guide.wisc.edu/undergraduate/engineering/industrial-systems-engineering/","termCode"=>"1214","departmentOwnerAcademicOrgCode"=>"E0525","description"=>"INDUSTRIAL & SYSTEMS ENGR","graduateCatalogURI"=>"http://guide.wisc.edu/graduate/industrial-systems-engineering/","uddsFundingSource"=>"A1950","shortDescription"=>"I SY E","subjectCode"=>"490","schoolCollege"=>Dict{String,Any}("schoolCollegeURI"=>"http://www.engr.wisc.edu/","shortDescription"=>"Engineering","formalDescription"=>"Engineering, College of","academicOrgCode"=>"E","uddsCode"=>nothing,"academicGroupCode"=>"EGR"))
  "repeatable" => "N"
  "fullCourseDesignationRaw" => "INDUSTRIAL AND SYSTEMS ENGINEERING 524"
  "typicallyOffered" => "Fall, Spring"
  "foreignLanguage" => nothing
  "enrollmentPrerequisites" => "(COMP SCI 200, 220, 300, 301, 302, or 310) and (MATH 320, 340, 341, or 375) or graduate/professional standing"
  "titleSuggest" => Dict{String,Any}("payload"=>Dict{String,Any}("courseId"=>"024408"),"input"=>Any["Introduction to Optimization"])
  "minimumCredits" => 3
  "title" => "Introduction to Optimization"
  "lastUpdated" => 1618558603523
  "workplaceExperience" => nothing
  "courseDesignation" => "I SY E 524"
  "generalEd" => nothing
  "topics" => Any[]
  "description" => "Introduction to mathematical optimization from a modeling and solution perspective. Formulation of applications as discrete and continuous optimization problems and equilibrium models. Survey and appropriate usage of basic algorithms, data and software tools, including modeling languages and subroutine libraries. Enroll Info: None"
  "lastTaught" => "1214"
  "creditRange" => "3"
  "catalogSort" => "00524"
  "currentlyTaught" => true
  "firstTaught" => "1164"
  "catalogNumber" => "524"
  "maximumCredits" => 3
  "fullCourseDesignation" => "INDUSTRIAL AND SYSTEMS ENGINEERING 524"
```


6.B Enrollment Prerequisites Example

In [21]:
```
for (k, v) in sort(cls_dict)
    cls_name = get_cls_name(k)
    if occursin("MATH", cls_name)
        printstyled(stdout, cls_name, bold=true)
        println(": ", v["enrollmentPrerequisites"])
    end
end
```

COMP SCI/I SY E/MATH 425: (MATH 320, 340, 341, or 375) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
COMP SCI/MATH 514: (MATH 320, 340, 341, or 375) and (MATH 322, 376, 421, or 521) and (COMP SCI 200, 220, 300, 310, or 301 prior to Spring 2020) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
COMP SCI/I SY E/MATH/STAT 525: MATH 320, 340, 341, 375, or 443 or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
COMP SCI/I SY E/MATH 728: Graduate/professional standing
COMP SCI/I SY E/MATH/STAT 726: Graduate/professional standing
MATH 112: MATH 96 or placement into MATH 112. MATH 118 does not fulfill the requisite
MATH 113: MATH 112 or placement into MATH 113
MATH 114: MATH 96 or placement into MATH 114. MATH 118 does not fulfill the requisite
MATH 130: MATH 96 or placement into MATH 130 and classified as Elementary Education, Pre-Elementary Education or Pre-Special Education. MATH 118 does not fulfill the requisite
MATH 131: Grade of C in MATH 130 and classified as Elementary Education, Pre-Elementary Education, or Pre-Special Education
MATH 132: MATH 130 and 131 with grades of C or better. Open only to students classified as Elementary Education, Pre-Elementary Education, or Pre-Special Education
MATH 141: MATH 96 or placement into MATH 141. MATH 118 does not fulfill the requisite
MATH 211: MATH 112 or 114 or placement into MATH 211
MATH 213: MATH 211, 217, 221, or 275
MATH 221: MATH 114 or (MATH 112 and 113) or placement into MATH 221. MATH 211 or MATH 213 does not fulfill the requisite.
MATH 222: MATH 217, 221, or 275. MATH 211 or 213 does not fulfill the requisite.
MATH 234: MATH 222 or 276
COMP SCI/MATH 240: MATH 217, 221, or 275
MATH 217: MATH 171
MATH 298: Consent of instructor
MATH 319: MATH 222 or 276 or graduate/professional standing
MATH 320: MATH 222 or 276 or graduate/professional standing
MATH 321: MATH 376, (MATH 234 and 319), (MATH 234 and 320), (MATH 234 and 340), (MATH 234 and 341), (MATH 234 and 375), or graduate/professional standing
MATH 322: MATH 321 or 376 or graduate/professional standing
MATH 340: MATH 222. Not open to students with credit for MATH 341 or 375
MATH/STAT 431: MATH 234 or 376 or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 441: (MATH 320, 340, 341, or 375) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 443: (MATH 320, 340, 341, or 375) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 461: MATH 234 or (MATH 222 and MATH/COMP SCI 240) or MATH 375 or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
HIST SCI/MATH 473: Consent of instructor
COMP SCI/MATH/STAT 475: (MATH 320, 340, 341, or 375) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
COMP SCI/MATH 513: (MATH 340, 341, or 375) and (COMP SCI 200, 300, 301, 302 or 310) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) program

MATH 521: (MATH 234 and 467) or (MATH 322, 341, 376, 421) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 522: MATH 521 and (MATH 320, 340, 341, or 375) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 541: (MATH 234 or 375), (MATH 320, 340, 341, or 375), and (MATH 341, 375, 421, 467, or 521), or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 542: MATH 541 or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 551: (MATH 234 or 375), (MATH 320, 340, 341, or 375), and (MATH 341, 375, 421, 467, or 521), or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 552: (MATH 551 and 541) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 561: (MATH 320, 340, 341, or 375) and (MATH 322, 376, 421, or 521) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 567: MATH 541 or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 629: MATH 522 or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
OTM/I SY E/MATH/STAT 632: (MATH/STAT 431, 309, STAT 311 or MATH 531) and (MATH 320, 340, 341, 375, 421 or 531) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 698: Consent of instructor
MATH 699: Consent of instructor
MATH 704: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 705: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 716: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 722: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 725: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 742: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 752: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 761: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 764: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 773: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 776: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 801: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 807: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 820: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 825: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH/STAT 734: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH/STAT 833: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 853: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 873: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 921: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 941: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 951: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 967: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 990: Consent of instructor
MATH/STAT 309: MATH 234 or concurrent enrollment; not open to students with credit for MATH/STAT 431 or STAT 311
MATH/STAT 310: (MATH/STAT 309, STAT 311, or MATH/STAT 431) and (STAT 224, STAT 301, STAT 302, STAT 324, STAT 371, or ECON 310); or graduate/professional standing
MATH/STAT 710: MATH/STAT 709
COMP SCI/E C E/MATH 435: (MATH 320, 340, 341, or 375) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 376: MATH 375
MATH 341: MATH 234
MATH 421: MATH 234 or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 747: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program

MATH 519: (MATH 320, 340, 341, or 375) and (MATH 322, 376, 421, or 521) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 207: None
MATH 407: None
MATH 607: None
COMP SCI/MATH 715: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 135: Grade of C in MATH 130, and (MATH 112, 114, or 171), and classified in Elementary Education, Pre-Elementary Education, or Pre-Special Education
MATH 228: Member of Wisconsin Emerging Scholars--MATH Program
MATH 619: (MATH 322, 421, or 521) and (MATH 319, 320, 376, 415, or 519) or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 531: MATH 376, 421, or 521 or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
CURRIC/MATH 471: (MATH 341, 375, or 421) and MATH 461
MATH 848: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 849: Graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program
MATH 96: Placement into MATH 96. Department consent required to drop/swap from course
MATH 540: (MATH 234 or 375), (MATH 320, 340, 341, or 375), and (MATH 341, 375, 421, 467, or 521), or graduate/professional standing or member of the Pre-Masters Mathematics (Visiting International) Program

In [ ]: