The sched.it algorithm

Carlos Carral

What is sched.it?

sched.it is a scheduling web app. (No, not that scheduling)

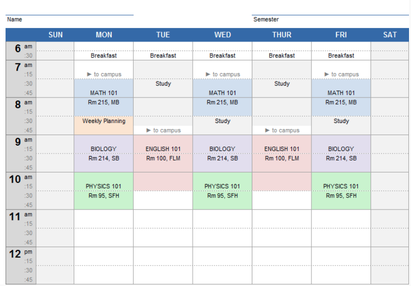
Dealing with schedules

- Your course sign up deadline is coming near
- You must sign up for a given number of subjects (you don't want to fall behind).
- You are presented with a long, fixed list of the courses available, possibly with a m2o relationship to their respective subjects (many available courses for a single subject).

How do I select the appropriate courses so that I can squeeze them all together, without falling behind?

In other words

“This semester I must take up Calculus, Algebra, Geometry, CS, Physics and an elective. I have many options for each subject, each with its own schedule. It would be nice to take some specific classes with specific professors. How can I find a schedule that actually fits my needs?”

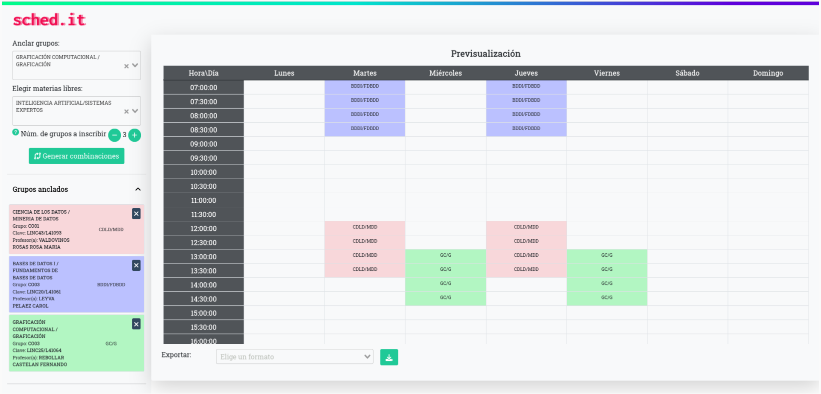


Name

Semester

	SUN	MON	TUE	WED	THUR	FRI	SAT
6 am							
:30		Breakfast	Breakfast	Breakfast	Breakfast	Breakfast	
7 am							
:15		▶ to campus		▶ to campus		▶ to campus	
:30			Study		Study		
:45		MATH 101		MATH 101		MATH 101	
8 am		Rm 215, MB		Rm 215, MB		Rm 215, MB	
:15							
:30		Weekly Planning		Study		Study	
:45			▶ to campus		▶ to campus		
9 am							
:15		BIOLOGY	ENGLISH 101	BIOLOGY	ENGLISH 101	BIOLOGY	
:30		Rm 214, SB	Rm 100, FLM	Rm 214, SB	Rm 100, FLM	Rm 214, SB	
:45							
10 am							
:15		PHYSICS 101		PHYSICS 101		PHYSICS 101	
:30		Rm 95, SFH		Rm 95, SFH		Rm 95, SFH	
:45							
11 am							
:15							
:30							
:45							
12 pm							
:15							
:30							
:45							

Enter the sched.it web app



sched.it

Anclar grupos:

GRAFICACIÓN COMPUTACIONAL / GRAFICACIÓN

Elegir materias libres:

INTELIGENCIA ARTIFICIAL/SISTEMAS EXPERTOS

Núm. de grupos a inscribir − 3 +

Generar combinaciones

Grupos anclados

CIENCIA DE LOS DATOS / MINERIA DE DATOS
Grupo: C001
Clave: LINC43/L41093
Profesor(a): VALDOVINOS ROSAS ROSA MARIA
CDLD/MDD

BASES DE DATOS I / FUNDAMENTOS DE BASES DE DATOS
Grupo: C003
Clave: LINC20/L41061
Profesor(a): LEYVA PELAEZ CAROL
BDDI/FDBDD

GRAFICACIÓN COMPUTACIONAL / GRAFICACIÓN
Grupo: C003
Clave: LINC25/L41064
Profesor(a): REBOLLAR CASTELAN FERNANDO
GC/G

Previsualización

Hora\Día	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
07:00:00		BDDI/FDBDD		BDDI/FDBDD			
07:30:00		BDDI/FDBDD		BDDI/FDBDD			
08:00:00		BDDI/FDBDD		BDDI/FDBDD			
08:30:00		BDDI/FDBDD		BDDI/FDBDD			
09:00:00							
09:30:00							
10:00:00							
10:30:00							
11:00:00							
11:30:00							
12:00:00		CDLD/MDD		CDLD/MDD			
12:30:00		CDLD/MDD		CDLD/MDD			
13:00:00		CDLD/MDD	GC/G	CDLD/MDD	GC/G		
13:30:00		CDLD/MDD	GC/G	CDLD/MDD	GC/G		
14:00:00			GC/G		GC/G		
14:30:00			GC/G		GC/G		
15:00:00							
15:30:00							
16:00:00							

Exportar: Elige un formato

Figure 1: A somewhat unassuming web app

The groundwork

A specific course C_i is represented as a 48x7 square matrix, where M=48 for 48 half-hour blocks between 00:00 and 23:59 and N=7 for 7 days in a week (granularity can be increased or decreased). If the class takes place monday, wednesday and friday from 00:30 to 01:30, a “1” is placed in its corresponding time blocks.

	MON	TUE	WED	THU	FRI	SAT	SUN
00:00							
00:30	1		1		1		
01:00	1		1		1		
01:30							
...	...	...	...	...	...	...	...
23:30							

The groundwork

So the same course in matrix notation looks like the following:

$$\mathbf{C}_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The groundwork

`The merge operation`

The merge operation tries to add two schedules together without overlapping.

It happens by adding two MxN course matrices. For each x_{ij} element in the resulting matrix, the following is checked:

$$x_{ij} \leq 1$$

If this condition holds for all elements, the merge was succesful.

The groundwork

`The merge operation`

Succesful merge:

$$\begin{bmatrix}0 & 0 & 0\\1 & 0 & 1\\0 & 0 & 0\end{bmatrix} + \begin{bmatrix}1 & 0 & 1\\0 & 0 & 0\\0 & 1 & 0\end{bmatrix} = \begin{bmatrix}1 & 0 & 1\\1 & 0 & 1\\0 & 1 & 0\end{bmatrix}$$

No overlapping

The groundwork

`The merge operation`

Unsuccesful merge:

$$\begin{bmatrix}0 & 0 & 0\\ 1 & 0 & 1\\ 0 & 0 & 0\end{bmatrix} + \begin{bmatrix}0 & 0 & 0\\ 1 & 1 & 1\\ 0 & 1 & 0\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ 2 & 1 & 2\\ 0 & 1 & 0\end{bmatrix}$$

Overlapping between courses

Course sets and stacks

A course set is defined as as a set of course matrices related to one another, via a common subject. All algebra courses could be grouped into a set $A = \{A_1, A_2, \cdots, A_i\}$

Course sets and stacks

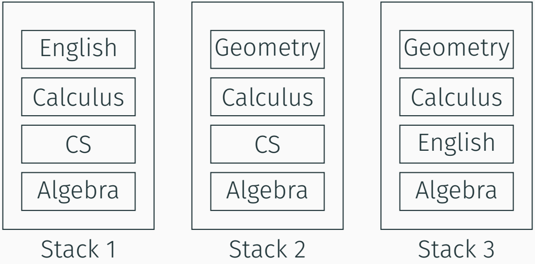
A subject stack is defined as a combination of different course sets. Please note that combination is being used in the context of combinatorics (i.e ordering is not important.). This concept is important because the total length of available subjects might be greater than the desired number of courses to sign up for.

Course sets and stacks

all subjects = {Algebra, CS, Calculus, Geometry, English}

|all subjects| = 5

desired stack size = 4



English
Calculus
CS
Algebra
Stack 1

Geometry
Calculus
CS
Algebra
Stack 2

Geometry
Calculus
English
Algebra
Stack 3

Now, for the actual algorithm

Input parameters:

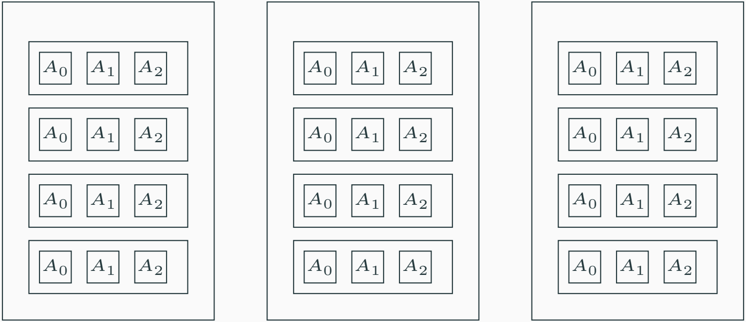
G: The global set of all available course sets (specific courses grouped by subject).

k: The desired total course set length to be fit into the final schedule (How many courses do we want to sign up for?)

Now, for the actual algorithm

`Step 1:`

Generate all possible combinations of length k over G into stacks

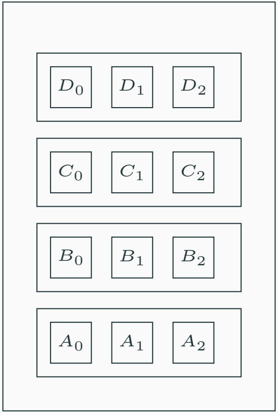


$k = 4$

Now, for the actual algorithm

`Step 2:`

For each stack, do the following:



Now, for the actual algorithm

`Step 2.1:`

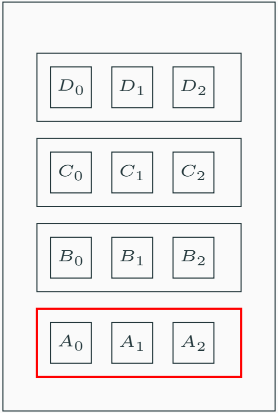
Initialize an empty schedule grid

$$\mathbf{C}_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Now, for the actual algorithm

`Step 2.1:`

Select the first stack level



Now, for the actual algorithm

`Step 2.2:`

Merge the first course with the blank schedule. This operation will always be succesful.

$C_i = \{A_0\}$

Now, for the actual algorithm

`Step 2.X:`

Recursively, perform the following, depth first:

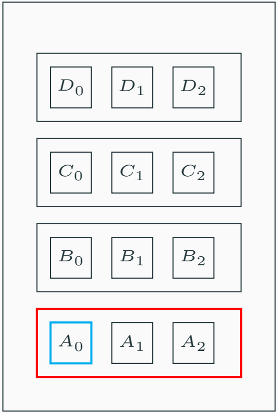
a) If there are no more stack levels, we have succesfully completed a path, add the resulting schedule C_i into the generated schedules Q.

b) Else, for each course, try to merge into the blank schedule. If succesful, descend into the next stack level.

c) If the merge was unsuccesful, go over the next schedule at the current stack level.

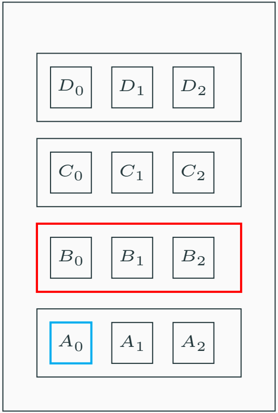
Now, for the actual algorithm

$$C_i = \{A_0\}$$



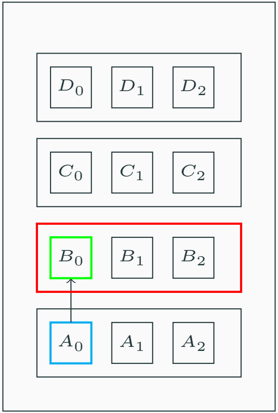
Now, for the actual algorithm

$$C_i = \{A_0\}$$



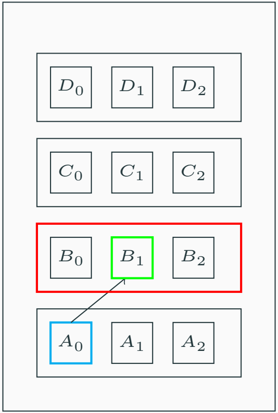
Now, for the actual algorithm

$$C_i = \{A_0\}$$



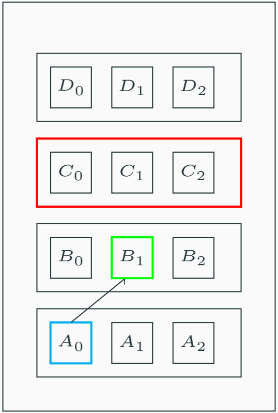
Now, for the actual algorithm

$C_i = \{A_0\}$



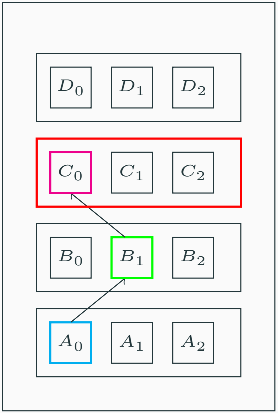
Now, for the actual algorithm

$$C_i = \{A_0, B_1\}$$



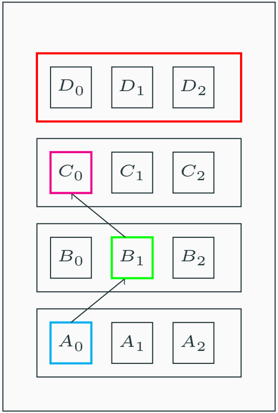
Now, for the actual algorithm

$$C_i = \{A_0, B_1\}$$



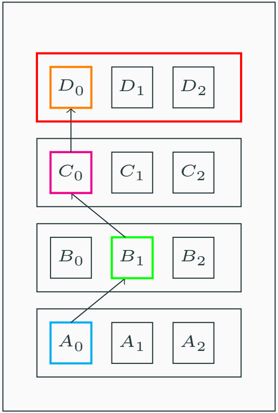
Now, for the actual algorithm

$$C_i = \{A_0, B_1, C_0\}$$



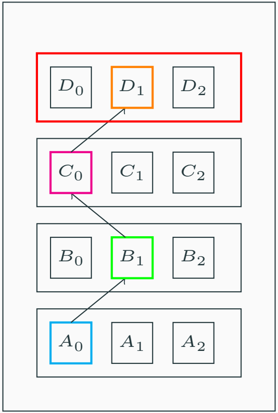
Now, for the actual algorithm

$$C_i = \{A_0, B_1, C_0\}$$



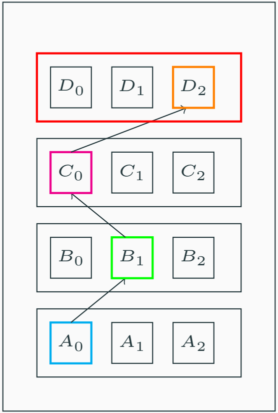
Now, for the actual algorithm

$$C_i = \{A_0, B_1, C_0\}$$



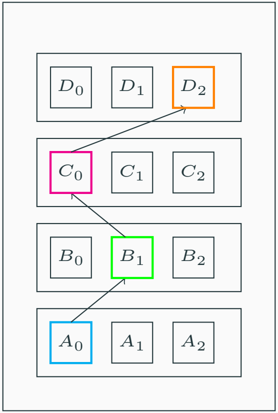
Now, for the actual algorithm

$$C_i = \{A_0, B_1, C_0\}$$



Now, for the actual algorithm

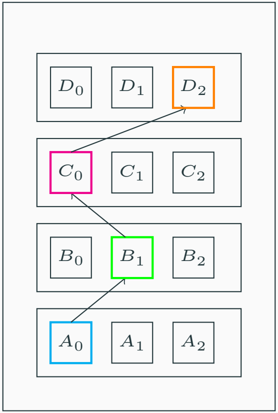
$$C_i = \{A_0, B_1, C_0, D_2\}$$



(End of stack)

Now, for the actual algorithm

$C_i = \{A_0, B_1, C_0, D_2\}$
$Q = \{C_i\}$



Now, for the actual algorithm

`The Output`

The resulting list will contain all possible schedules, based in the initial course list and desired schedule length.

Now, for the actual algorithm

Seeds

The algorithm has the advantage of allowing an optional “seeds” input parameter

S: Set of specific courses desired, chosen beforehand.

¿Do you want to take a class with a specific professor? That goes here.

Now, for the actual algorithm

Optional optimizations

If desired, an additional optimization step could be added so that unproductive time blocks are minimized.

Now, for the actual algorithm

`Other possible uses`

- Spreading coverage of limited resources over time.
- Employee shift scheduling

Bonus

Implementation details

Standalone web browser app, brought to life by Rust + WASM.

The End

Thank you!