Notes on "Arranger" 1.1.2

Xinyuan Zhou

May 25, 2025

Table of Contents

Chapter 1

Overview

1.1 What this Project Does

So... my teacher's rules for rooming on a very specific field trip that I am not going to disclose is as follows:

- No boys sleep with girls. (This is trivial to come up with, I guess).
- Most rooms by grade level, but people can *request* to sleep with people from other grade levels, and a very limited number of people can do this. We agreed that a numerical priority will be filled in to prioritize those requests, since there is a cap on the number of people that gets to do this.
- At most 4 people per room, wasting minimal rules.

Therefore, I came up with this approach:

- Let my teacher assign a priority to each cross-grade-level request, sort them from highest to lowest, consider throwing all people involved in the request into the multiple-grade-level pool (one for boys and one for girls), if it causes it to overflow, don't do this request. But we do this in the order of priority.
- There's multiple-grade-level boys & girls and $\{6, 7, 8\}$th grade boys and girls, and we call each of those a *sector*. A greedy algorithm documented in the README (and also at section 2.2) gets used to separate those into actual rooms.
- To waste minimal rooms, put non-complete rooms into a "leftovers" sector and just put them in groups of 4 (or the room size) artificially. This is only if there's *not* ≥ 3 leftovers in the sector.
- Some documents are generated to document those rooms and facilities are provided to change rooms.

A VBA script is provided to convert to a Word document. Everything is in the form of `txt` files.

1.2 Setup

Add a new office script, replace the default content with the ones from `Auto-Setup-Script.txt` in the root directory of the public distribution, and click run. Next, hook up the 4 scripts in `app-scripts/` in their respective locations indicated in the workbook set up by the script. To hook up a script, go to the Automate tab, create a new office script, replace its contents with the contents from the appropriate text file, then click on the three dots and choose "Add in Workbook".

Setting up the VBA conversion-to-Word thingy: You need to bring up the "Developer" tab as described in https://support.microsoft.com/en-us/office/show-the-developer-tab-e1192344-5e56-4d45-931b-e5fd9bea2d45. In the Developer tab, click on Visual Basic, on the top-level V.B. project on the left, click `Insert -> Module`. Put the code inside from the text file. After that, use the Developer tab to hook up a button (on Windows, use FORMS control). Choose "Export" for the macro it triggers.

Also you must include Microsoft Word object library in your References (see https://learn.microsoft.com/en-us/office/vba/language/how-to/check-or-add-an-object-library-reference).

Chapter 2

The Office Scripts

2.1 Things to Fill In

2.1.1 Student Info

Fill in Columns `A`, `B`, `C`, `D` starting with row `2`. Put the number of students in `F3`, multiple-grade-level-quota for boys in `F5`, and the quota for girls in `F7`. All of those are used by the program (all 3 scripts) because they need to lookup info. Fill in Boy or Girl in gender with capital B and capital G.

For the FSMS defaults, put 4 in `F9`, 2 in `F11`, 1 in `F13`, and 2 in `F15`. Their labels above them should be self-explanatory for what they actually do.

2.1.2 Requests

They should be straightforward, but student name has to be *first and last* name with a space without commas. Fill in number of requests in the highlighted grid.

Click the button we just setup on this sheet to highlight the multiple-grade-level requests. Blue is for boys, pink is for girls. Fill in the highlighted cells for priority (lower is higher). See this section for details.

2.1.3 Anti-Requests

Same as requests but no priorities (unnecessary) and those people does *not* want to sleep together.

2.1.4 Room Numbers

Fill in number of rooms in Boy's Floor and Girl's Floor on the respective fields, `A1` and `C1`. Starting with `A4` and `C4`, downwards, put the room numbers for the floors.

2.1.5 Execution

Go to the Students Info sheet, run it and enjoy!

There will be 3 sheets generated: Boy's Arrangement, Girl's Arrangement, and Roster.

2.2 Algorithm

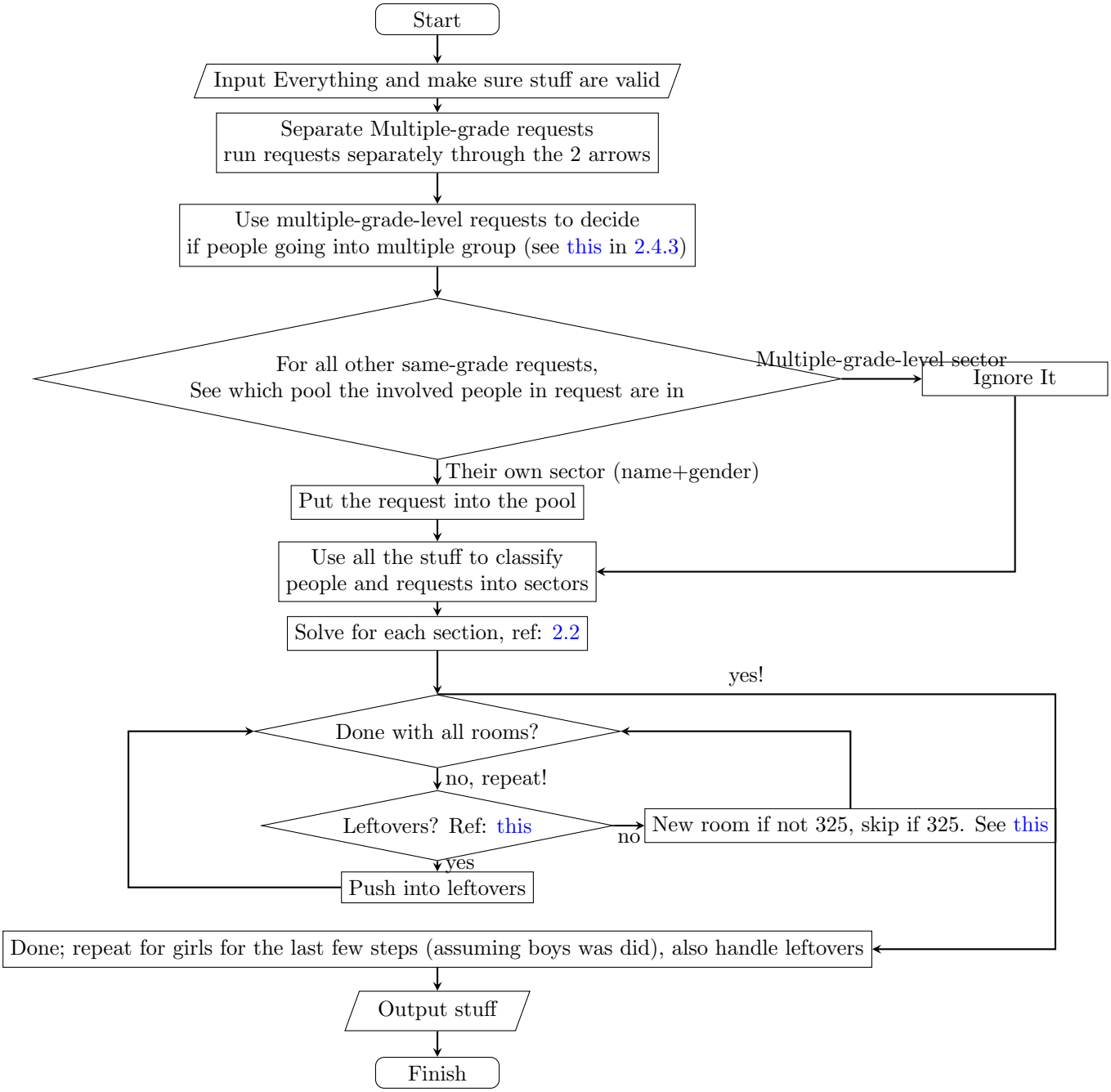
Since all rooms have to be of 4 (or another constant size specified in Student Info), we enumerate the people in a room, one room at a time; find the room with the least number of constraints we break. Each time a room is chosen, those people are removed from the pool.

The time complexity is $\mathcal{O}(n^m)$.

Anti-requests are added by comparing the number of requests broken and then the number of anti-requests broken if they are the same. This might not be optimal.

2.3 Process

Here's a (very) abstract graph that explains what this does:



See the source code for more details.

2.4 Concepts & Featues

2.4.1 Definitions

A **request** is a statement in the form "a wants to sleep with b". The direction does not matter. If they requests each other (internally known as "couples"), it is only counted as 1 request. A request may carry a **priority** if it is across grade levels.

An **anti-request** is similar that says "a does not *want to* sleep with b" without a priority. See anti-requests in Sheets to Fill In and 2.2 for more info.

A person, participant etc. is someone who is involved in the rooming.

2.4.2 Participant Identification

A participant is identified by its first name, last name, and his/hers grade level. There's duplicate names.

2.4.3 Sectoring

Definition. Separate participants into groups:

	6	7	8	multiple
Boy				
Girl				

where every blank space inside represents a "sector" or group.

"Multiple" (aka. "Inter-grade") Sectors

Those are across grade levels. Requests ("pairs") with high priorities are thrown in first, and then the low ones. If it gets stuck where the quota still allows 1 empty spot but a pair of 2 where neither person is already in the pool, it is skipped; however, if there is another pair with lower priority that has 1 already in the pool, it is thrown in.

Leftovers

Let's say you have 11 people one sector. The algorithm extracts the first 8 into 2 rooms, but there are still 3. They get thrown into a pool according to their gender named "leftovers". In the leftovers sector, people are not arranged but just put into there in the order their sectors go. Boys can never sleep with girls so there are separate leftover groups (kind of a mistake that it's implemented, there are only a few girls and they fit perfectly in one room last time).

Ignored Requests

Boys cannot sleep with girls so requests across multiple genders are ignored.

Requests where one or more of the two people that are not in the multiple-grade-level pool are ignored. Same-grade-level requests are also thrown away in the multiple-grade-level pools.

Anti-requests are ignored if the people involved are not in the same sector.

2.4.4 Special Rooms

Room numbers are reserved if such room number satisfying rules 325, $*325$, or 3251 are in the list, where $*$ is any digit ($0 \sim 9$), (only 3251 not $325*$ to avoid skipping too much rooms).

To disable 325 checking, make the `is325` function in main script always return `false`. The reason for doing this by default cannot be disclosed.

Chaperone rooms are marked at the end of each gender arrangement floor, by the numbers specified in the Student Info data entry sheet. Other extra rooms that has room numbers are marked as `OTHER`. Do not confuse with `other Boy/Girl` which is the external term for leftovers.

2.5 Output Format

2.5.1 Arrangement Sheets

Example. `1324|other Girl|2|b a 8|d c 8|x f 8|w2 z 8`

Column `A`: Room numbers.

Column `B`: Sector.

Column `C`: Room Ordinal in Sector – starting from room 1 in each sector.

Columns `D, E, F, G (and more if you have more pople in a room)`: Names in Room.

2.5.2 Adjusting Rooms

To adjust people in rooms, use the generated arrangement sheets – fill in the small "form" to the right. The highlighted parts should be the cell *address*, like `A3`. They will swap if there is already a person in the "to" field and it will update all output sheets, although it is still dependent on the input sheets so do not delete them.

To rename rooms, use the form below the previous form. Make sure there are no name collisions, although it is not that important. If you swap the names of two rooms, do

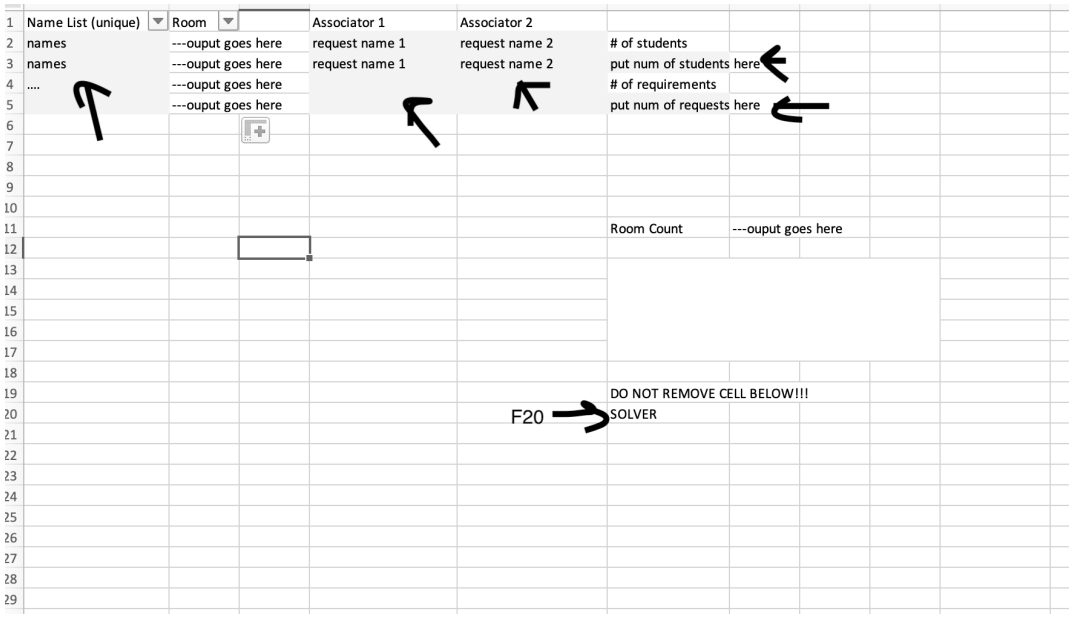
`Rename A to C` (C may be a random name like `Temp 1`)
`Rename B to A`
`Rename C (‘‘Original’’ A) to B`

To swap the rooms' locations on the sheet, simply do it. There are no line number references.

To change room numbers, there are no simple ways to do it. Start over. **That is why you should log all your actions. The algorithm is deterministic even though room numbers change.**

2.6 Final Remarks

To just do the part for the greedy algorithm (for seating charts etc), use the following form:



and run the main script from that sheet (in this case only about 200 lines are ran).

Non-features: Larger size may fail in this program. Larger arrangements, such as buses are a non-feature.

Chapter 3

Converting to a Word Document

The first line of the sheet that you run the VBA script from is a header that you can put anything in that gets skipped during processing. Starting from the second line paste in stuff from Arrangement sheets as output of first program, skipping the first line, remove everything you don't want in a final sheet of Word table to send out. If you want to color-code room grids, the color should be in the first column.

You may want to change up the sector names etc. Experiment with things to get the result you want.

There will be three dialogue boxes that pop up, the first two of which are self-explanatory and the last of which basically means the number of rows in the table.

Index