# Integer Optimization (Spring 2025): Project 2
# Heuristics for Set Covering

- This project will be worth 65% of the total credit for projects.
- You must submit a brief report outlining your results and your methodology (various choices for the heuristics). You must also submit your Matlab/Python/... files.
- Your email submission should include all files in a **compressed (e.g., tar-gzipped or zip-ed) folder** that identifies you starting with **`FirstnameLastname`**, e.g., **KrispyZoolander_Project2.zip**.
- Begin the **SUBJECT of your email** submission with the same `FirstnameLastname`, e.g., "Krispy-Zoolander Project 2 submission".
- This project is due **by 11:59 pm on Friday, May 2**.

You will be implementing the greedy and the modified greedy heuristics for solving the receiver location problem (modeled as a set covering problem—see Lecture 22 and Lecture 23) in Matlab or Python (or another language/package). We will consider the following two problem instances (both uploaded to the course web page):

- `phase1.txt`, with 108 meters and 117 candidate locations; and
- `cap360.txt`, with 7172 meters and 6550 candidate locations.

Each of these two data files looks like a matrix with two columns, and represents the $A$-matrix (incidence matrix). Thus, if a row of the data file has numbers $i$ and $j$, it means that $A_{ij} = 1$, and it indicates that the meter $i$ is covered by a receiver located in location $j$. You must represent the matrix $A$ in the sparse format in your code. In Matlab, **for**-loops will make your code really slow; you should try to avoid them wherever possible.

Recall that in each step of the greedy algorithm, the pole that covers the most uncovered meters is picked. In the modified greedy algorithm, the pole that maximizes a scoring function is chosen in each step. Try the various scores introduced class: $\text{Score}_1$, $\text{Score}_2$, and $\text{Score}_g$. After running either algorithm, you must clean-up the solution, as discussed in class. You can decide how frequently to run the clean-up steps.

1. Report the value of the solution (total number of poles selected) as well as the running time for both greedy and modified greedy algorithms on the two problem instances. Your code must attain the following conservative benchmarks in order to receive full credit.
   - greedy on phase1.txt: value=25, time=2 seconds.
   - modified greedy on phase1.txt: value=25, time=4 seconds.
   - greedy on cap360.txt: value=630, time=8 seconds.
   - modified greedy on cap360.txt: value=620, time=30 seconds.

   To measure the running time of your program, add the command `t0=clock;` at the top of your Matlab m-file, and then add the command `runtime=etime(clock,t0);` at the end of your file. The total running time is then stored in the variable `runtime`.

2. **Preprocessing:** Implement the steps of preprocessing the $A$ matrix as discussed in class -

   - remove singleton rows—rows that have a one in a single column, select the facility in question (corresponding to the column), and remove all rows covered by this selected column;
   - remove rows that contain row $j$; and
   - remove column $i$ if there is another column that contains it.

   Repeat the greedy and modified greedy algorithms, now doing preprocessing after each pole (facility) is selected. Report the value and the running time in each case. You still clean-up the solution as before. The benchmarks for the runs with preprocessing are:

   - greedy on phase1.txt: value=24, time=10 seconds.
   - modified greedy on phase1.txt: value=24, time=12 seconds.
   - greedy on cap360.txt: value=590, time=100 seconds
   - modified greedy on cap360.txt: value=590, time=250 seconds.

   When preprocessing the $A$ matrix, you could run the three steps described above repeatedly till the $A$ matrix does not change any more. But doing preprocessing this way might be quite time-consuming, and hence you might have to limit the number of preprocessing steps done after each pole selection to some finite number. You could also choose not to perform one of the preprocessing steps (if you think that the particular step is much more time-consuming than the others). The goal is to achieve the benchmarks specified. Do explain the set-up you choose to implement in your report.

3. Now let the code perform full preprocessing after each pole selection. Report the value and running time for the greedy and modified greedy algorithms applied to the cap360.txt instance.

   Your final Matlab code should ask the user to input the name of the data file (phase1.txt or cap360.txt), and also ask the user to select whether preprocessing should be done (after each pole selection) or not. It should output the final value (number of poles selected) and the total running time.