State of the project

* Runs for anything make-able by the make\_random\_input (except the corner case)
* Runs on Haverford Data
* Some experimental analysis is done
  + 4 cases, discussion of optimality and timing for each
    - SHOULD do more and compare #4 and #5 (hc) optimalities
  + Parsed the HC data and enumerated the cases where a student takes a 100-level and a 300-level course in the same subject, they are in the report
    - Use that as one of the constraints – come back here
* What’s left
  + Implement the constraints
    - Analyze what they mean and how they work / how we can make recommendations based off what we observe – Report
  + Clean up the report
  + Deal with the bug

Constraints

1. Consider major/minor
   1. Change setCost to just add 1
   2. Make setCostSorted which assumes a sort and adds the cost(i,j) as we had it before
   3. Pass in major/minor info for each student and then sort accordingly in the preference parsing – algorithm calls setCost depending on whether we want to handle this constrain this time
   4. Is\_valid\_sorted needs to be made to check that
   5. Parsing needs to handle weighted preferences for the max optimality #
2. Consider distribution requirements (and do the same weighting as above)
3. Bin 100 and 300-level courses from the start if they are in the same subject (not the one’s we noticed do it a lot)
   1. Pairwise courses 100 and 300? If so bin and then move on (copy code from current algo)
4. Check subject for setCost and if the two classes are in a list of [(physics, math)…] of subjects that shouldn’t collide, give them a higher conflict # (not sure what)
5. HC and BMC class back-to-back? Avoid that in createschedule
6. Same requirement different school, bit right away. A lot like #3. We know their conflict is 0. But the enrollment data but have them signed up for both for lottery purposes.