**Challenge: HiTec Registry**

**Justification**

I chose this problem for the challenge because it is something that has being causing a lot of rework since the foundation of the Welcome Committee. Our current system distributes the students into teams using a random number with no hard constraints involved, therefore some of the teams were exceeding the maximum size while other teams had no members. Using AI to solve this problem is the best solution, specially using searches, as a sequential method would have a really high cost when the number of students rises to one thousand or more (which happens).

**Documentation**

To properly run the program needs only two things: Python 3.5 and the library networkx (<https://networkx.github.io/>) as it is the one used to generate the graph.

To run it in console use: python run.py and the program will ask for the number of expected students and the number of teams.

**Analysis**

*Description*

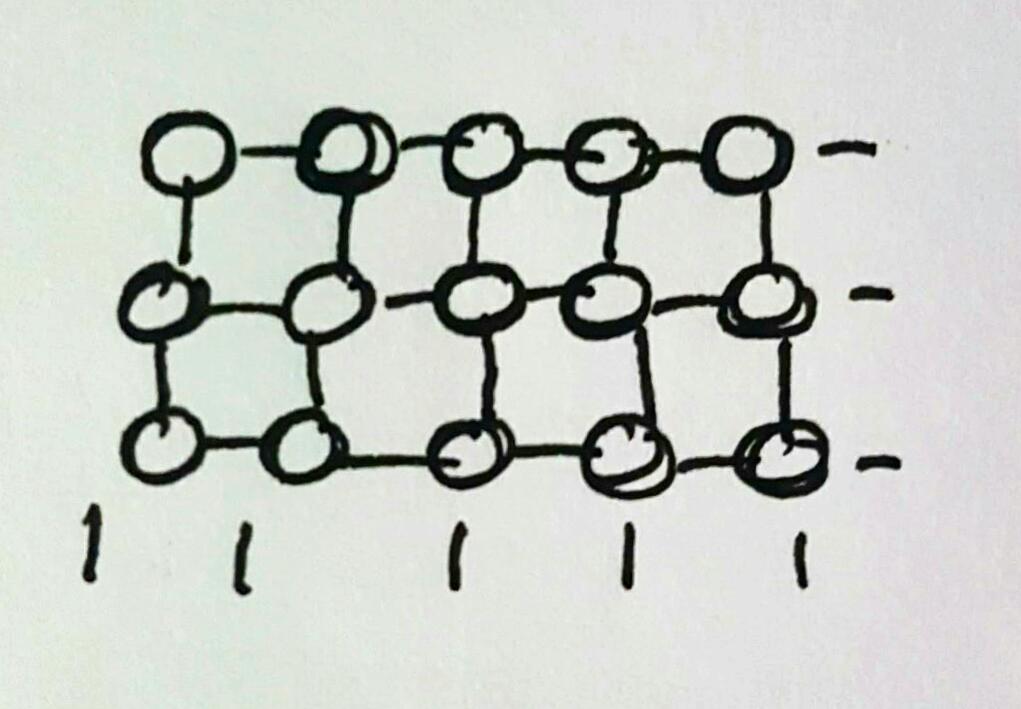
For this problem the initial state would be the given starting team, the possible actions would be to assign the student to the current team or to search in the neighbor nodes. Assigning the student would mean that the search is over and a new student can be assigned with a new initial state, to search in the neighbor nodes would mean to keep looking until the student can be assigned to a team. The goal state of the node is to have space and have maximum 80% of Mexicans in the team and there would be no path cost for the search.

*First Approach*

The first approach was a failed attempt of bound and branch with uniformed cost search. It failed because the nature of this problem doesn’t fit any type of tree as it is too hard to come up with a parameter to organize the teams into branches. What this attempt pretended to do was to create a tree based on the cost that would have the insertion of a new student on each team and then choosing the lowest cost as the destiny team. The main flaw was that a new tree had to be created for each student with the branch and bound algorithm which was extremely inefficient and then choose the lowest cost of all which was a greedy approach.

*Graph vs Tree*

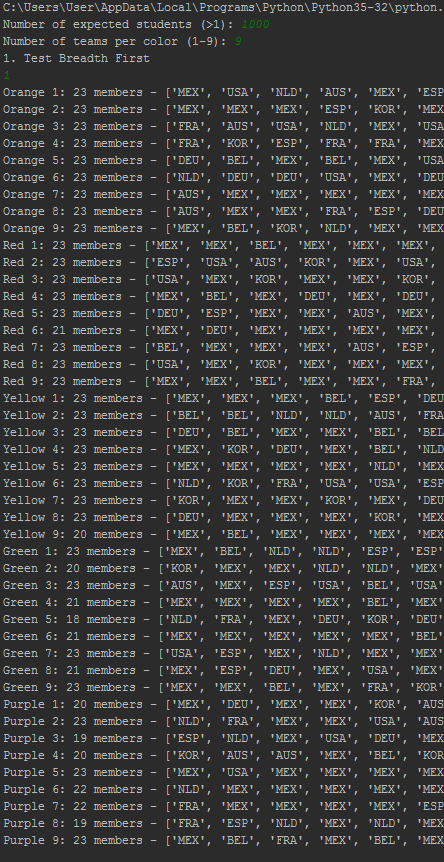
As the main problem was to create a tree, I decided to create a graph instead. Using the teams as nodes and stablishing the edges to create a grid-like graph. As the teams are divided in colors and each color from 1 up to 9, the edges connect the teams with the same color and the ones with the same number.

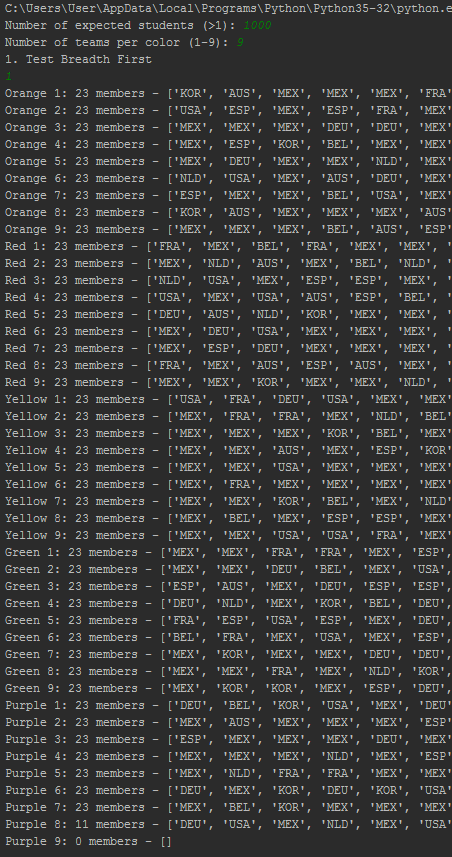


*Breadth First instead of Uniform Cost or A\**

Another of the main problems I found during the development of this challenge was the cost that had to be assigned to the edges so that algorithms like A\* and USC could work. As I mentioned before, the cost has to be calculated differently for every student. This has a great impact on the performance as the graph (or tree) has to be remade every iteration. Also, as it is a real-time insertion of a student into a team, I thought that breadth first would fit this scenario perfectly as it is given a start point and then finds for the shortest path to its goal condition, which in this case would be a team with available space and less than 80% of the members being Mexicans.

The second discovery I made was the importance of changing the initial state for each student, at first every search would begin at the team number one but what this created was a similar problem than the one we already have, the last teams were a lot smaller than the first teams (being the first teams full). By observing this I decided to implement a random between 1 and *n* being *n* the number of total teams in the graph. By adding this random number the student distribution balanced quite considerably.





The first image shows the distribution of students into teams with the random initial state and it can be observed that the difference of sizes is not that great (being 23 the max size) while the second image shows the attempt starting the search from the same point every time causing the teams to be filled in order and leaving the last two teams with a huge difference between the number of members.

It is important to note that the maximum size of the teams is calculated by a ceiling division, therefore the total number of expected students is (sometimes) smaller than the sum of all the possible members in the teams (in this case causing the team Purple 9 to be empty).

**Conclusion**

It is almost impossible to create the perfect balance between teams and for this specific problem is not that important for the teams to have some differences, the main focus is to do it fast and good enough, that is the main reason I chose Breadth First, because it is used to find the shortest path while algorithms as A\* analyze every possible outcome or Uniform Cost that take decisions based on parameters that are hard to stablish in this case. By stablishing a hard constraint as the maximum team size (which is so simple and obvious but at the same time it is not taken into account in the actual registry system) I was able to distribute the students into teams with similar numbers of members between them. And by setting soft constraints as the maximum percentage of Mexicans per team, I was able to create international teams.