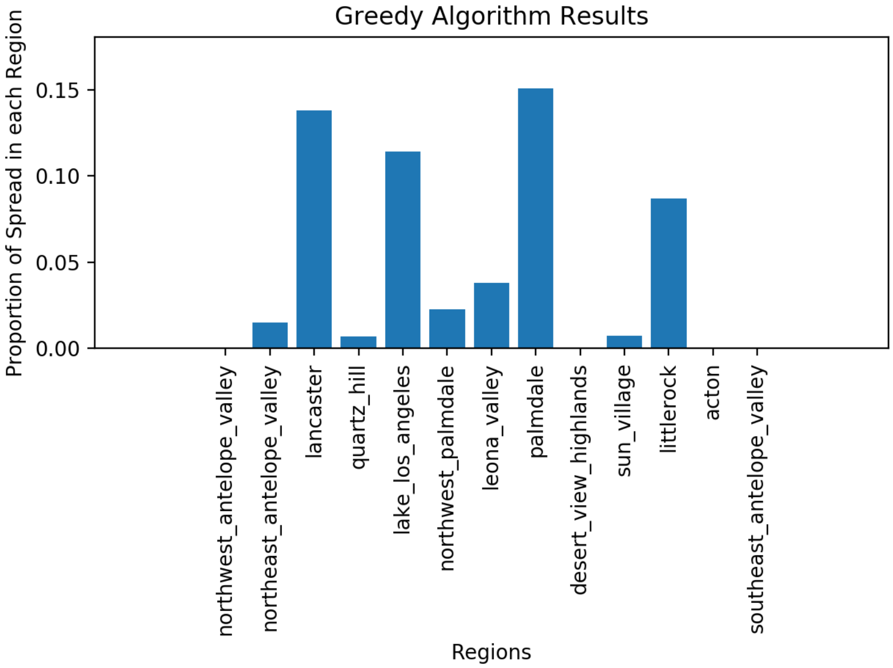
Task 1

1. The independent cascade model is called `independent_cascade` in `main.py.`
2. The algorithm implemented is the Greedy algorithm called `greedy` in `main.py.`
   - After running influence propagation 500 times, I have obtained the following results :
     - 25 people chosen as seeds are as follows :
       - `19, 264, 17, 13, 265, 282, 263, 460, 18, 296, 26, 240, 155, 136, 266, 82, 30, 409, 12, 268, 424, 74, 313, 29, 399`
     - The average number of people influenced is as follows :
       - `61.686`
3. For my best performing seed set which is shown above, I have obtained the results as shown in the graph below averaged over 500 simulations :

Name of the region	Percentage of people in that region receiving the information
northwest_antelope_valley	0.0
northeast_antelope_valley	1.48616
lancaster	13.80596
quartz_hill	0.68034
lake_los_angeles	11.41276
northwest_palmdale	2.24
leona_valley	3.8
palmdale	15.06466
desert_view_highlands	0.0
sun_village	0.72036
little_rock	8.7
acton	0.0
southeast_antelope_valley	0.0

The results can also be visualized as follows :



Results using Greedy Algorithm averaged over 500 simulations

My Analysis :

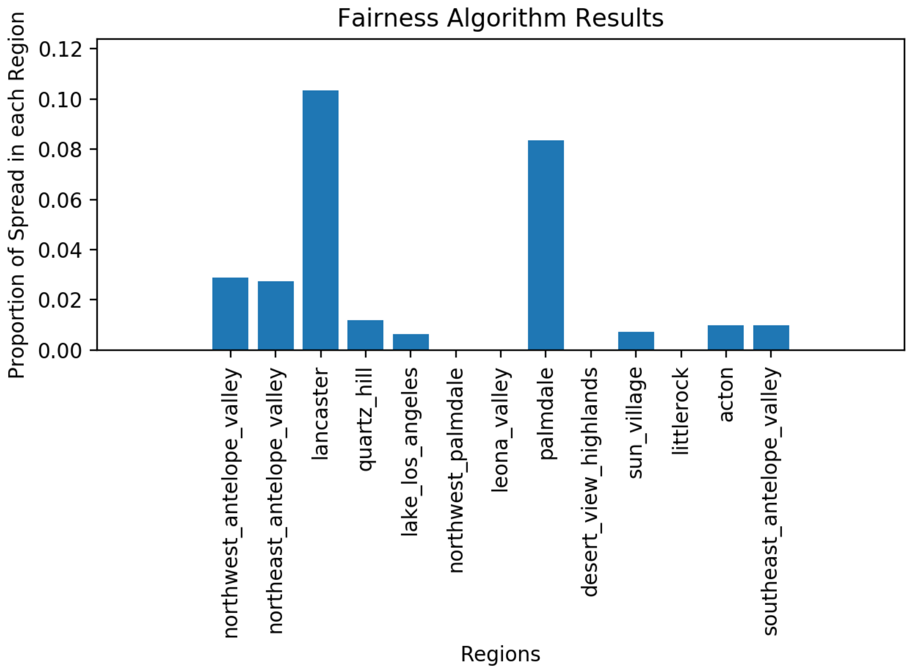
- What I mainly see is that the proportion of people being influenced in each region varies wildly for each region.
- For some regions around 8-15% of their population have received the information, however, in some other regions, around 0-4% of their population have received the information.
- This may be explained by the population of the regions itself.
- Regions with very high population (lancaster and palmdale) have a higher proportion of people receiving the information because of the increased influence probability among edges in the same region and vice versa.
- However, some very low population regions (lake_los_angeles and littlerock) still have around 8-12% of their population receiving the information.
- This is not fair to other regions with similar population sizes and low proportion of people receiving the information.

Task 2

1. The algorithm implemented is called `max_fair` in `main.py.`
   - The `max_fair` algorithm chooses one seed at a time like the Greedy algorithm, however, unlike the Greedy algorithm, the `max_fair` algorithm calculates the variance between proportion of people receiving information for each region along with the average number of influenced nodes of the seed set called spread.
   - Then, it chooses the seed set which has minimum variance and maximum spread by :
     1. Sorting according to variance.
     2. Reverse sorting for those seed sets which have the same variance.
   - Thus, it maximizes the minimum influence received by any of the region, as proportional to their population.
   - After running influence propagation 500 times, I have obtained the following results :
     - 25 people chosen as seeds are as follows :
       - `111, 320, 56, 447, 458, 74, 314, 99, 31, 344, 210, 47, 334, 179, 430, 366, 184, 304, 30, 350, 114, 446, 122, 373, 35`
     - The average minimal fraction of people influenced is as follows :
       - `38.71`
   - For my best performing seed set which is shown above, I have obtained the results as shown in the graph below averaged over 500 simulations :

Name of the region	Percentage of people in that region receiving the information
northwest_antelope_valley	2.88
northeast_antelope_valley	2.7431
lancaster	10.32586
quartz_hill	1.18726
lake_los_angeles	0.64496
northwest_palmdale	0.0
leona_valley	0.0
palmdale	8.36292
desert_view_highlands	0.0
sun_village	0.72036
little_rock	0.0
acton	0.98172
southeast_antelope_valley	1.0

- The results can be visualized as follows :



Results using Fairness Algorithm averaged over 500 simulations

2. The results are as follows :
   - The average minimal fraction of people influenced using the `max_fair` algorithm is as follows :
     - `38.71`
   - The average number of people influenced using the `greedy` algorithm is as follows :
     - `61.686`
   - The price we have to pay for fairness is as follows :
     - `(61.686 - 38.71) / 61.686 = 37.25%`
     - We influence around 37.25% less people using the `max_fair` algorithm compared to the `greedy` algorithm.

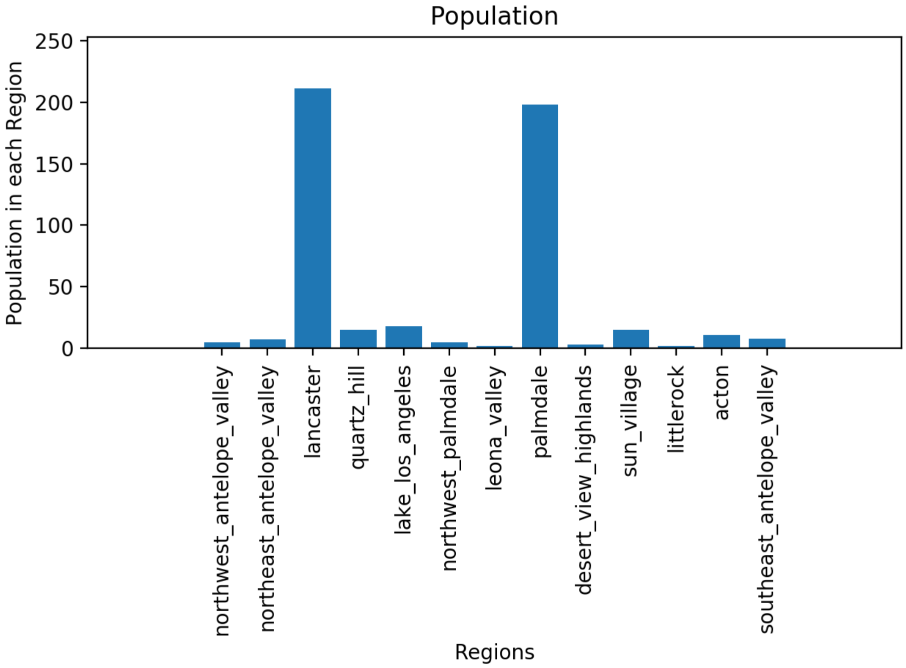
My Analysis :

- We influence fewer people, but our output ensures some level of fairness of resource allocation according to population size.
- Regions with very high population (lancaster and palmdale) have a higher proportion of people receiving the information because of the increased influence probability among edges in the same region and vice versa.
- However, compared to before, now, there is no region which has very low population along with a higher proportion of people receiving the information.
- Therefore, we can say that it is somewhat balanced.
- We could've improved the fairness if we had a bigger dataset.

- This is because influence probability is low and number of nodes in most regions is very low, therefore, the spread for a node in a very low population region is usually very low.
- Therefore, if number of nodes would’ve been higher in most regions, we could’ve produced improved fairness.

Population Size

The population size according to region is visualized as follows :



Total Population Size

This visualization is for reference purposes only.

References

Starter code for Influence Propagation and Greedy algorithm taken from : https://hautahi.com/im_greedycelf