## Administrative

* Team Name: County Rankers
* Team Members + Github user names:
  + Yash Patel - ynpatel910
  + William Chi - williamchiii
  + Philip Valvo Schnotalla – SpaceJunk285
* Link to GitHub repo: https://github.com/ynpatel910/AlgoP3
* Link to Video demo: <https://www.youtube.com/watch?v=CfGwWa0RVX0>

## Extended and Refined Proposal [Suggested 2 Pages]

* Problem: What problem are we trying to solve?
  + Country Rank addresses challenges of comparing and evaluating counties across the United States based on customizable demographic metrics. The main problem we aim to solve is the lack of accessible, user-friendly tools that assist individuals in choosing an ideal place to live by comparing counties on metrics such as income, education, homeownership, and more. Given the breadth of available data, manually evaluating and comparing counties can be overwhelming. Our goal is to automate this process, allowing users to select and prioritize metrics that matter to them, and receive a ranked list of counties that best match their preferences.
* Motivation: Why is this a problem?
  + Relocating to a new county is a major life decision. People typically spend countless hours researching and recording public data, which is often unorganized, or not tailored to their specific needs. We as a team have personally experienced this challenge and wanted to help streamline this process. Our motivation comes from the desire to build a tool that reduces the users’ research time, stress, improves their decision-making, and provides clear, data-driven county comparisons based on user-defined priorities.
* Features implemented
  + Interactive Metric Selection Menu
  + Custom Scoring Engine
  + User-Defined Metric Weighting
  + Input Validation
  + CSV Output
* Description of data
  + The data used in this project comes from the CORGIS Project.
  + The dataset includes demographic on U.S. counties such as poverty rate, education levels, or age distribution.
* Tools/Languages/APIs/Libraries used
  + **Language:** C++
  + **Libraries Used:**
    - <iostream>, <fstream>, <sstream>, <vector>, <map>, <unordered\_map>, <queue>, algorithm>, <iomanip>, <limits> <chrono>
* Algorithms implemented
  + MergeSort, HeapSort
* Additional Data Structures/Algorithms used
  + Struct County: stores name, state, and a map of metric values
  + Unordered\_map<string, double>: Used to dynamically store and access weights for each selected metric
  + Map<int, string>: Maintains the metric priority rankings
  + Pair<double, const County\*> : Used for score-county pairing.
* Distribution of Responsibility and Roles: Who did what?
  + Yash: Max Heap, Rank metric, load counties, compute scores
  + William: Heap and merge sort, ranking map, load initial metrics
  + Philip: Menu and input/output/data handling

## Analysis [Suggested 1.5 Pages]

* Any changes the group made after the proposal? The rationale behind the changes.
  + We didn’t have a GUI and we didn’t use the B Tree. This is because it didn’t fit in with what we were doing. B Tree wasn’t necessary for the implementation and running of the file.
  + We also added the ability for users to select whether they want to output the counties in ascending or descending score.
* Big O worst case time complexity analysis of the major functions/features you implemented
  + loadCoreMetrics() / loadFullMetrics()
    - O(1) - These functions initialize fixed-size vectors (10 & 36)
  + runMenu()
    - O(n) - Where N is the number of available metrics. It iterates over all metrics to display them
  + rankMetric()
    - O(M) – Where M is the number of selected metrics, which is limited to 5.
  + loadCounties()
    - O(N\*M) – Where N is the number of rows and M is columns. In the worst case every cell is visited.
  + compute\_score()
    - O(W) - Where w is the selected weights. It iterates over 5 weighted metrics and looks up each in a map O(1).
  + get\_ranked\_counties()
    - O(N log N) - n is the number of counties. Scores each county based on selected weights O(N). Sorts the counties by score O(N log N). Returns the top or bottom K counties O(K).
  + mergeSortArr()
    - O(N log N) - it is a standard merge sort recursion and merging with an array of size N (number of counties).
  + heapifyDown()
    - O(log n) where n = the size of the heap, this is because it is a binary tree
  + get\_ranked\_with\_heap()
    - O(m + k log m) where m = number of counties int the csv file, k is the number of elements to extract for the rankings. This is because we build the heap which takes O(m) and then we also must extract the top elements which takes k log m
  + printRankedFromCSV()
    - O(R) where R is the number of rows in the ranked.csv file. This is because it prints every row to the terminal.
  + main()
    - O(N log N)

## Reflection [Suggested 1-1.5 Page]

* As a group, how was the overall experience for the project?
  + It was a fun and interactive experience. We learned from our mistakes and explained certain things such as algorithms to each other when we had questions. Most in the group were also not very familiar with GitHub so learning the capabilities, and how to effectively and efficiently use GitHub is a skill we will certainly use in the future.
* Did you have any challenges? If so, describe.
  + Two challenges we came across were having different time schedules which made it difficult for us to create meetings where everyone can attend. The last challenge we came across was how to merge the branches into the main branch via the pull request. Each member ended up working on their own branch and some merging was required afterward to ensure the code was up to date and included everyone’s work.
* If you were to start once again as a group, any changes you would make to the project and/or workflow?
  + We would create a more detailed and organized plan on who would work on what that way we don’t work on the same functions at once and cause conflicts. There were a few instances where work was done twice or lost in a push so better communication and checking of the GitHub would help smooth any future projects.
* Comment on what each of the members learned through this process.
  + William learned how to code in a collaborative with multiple group members within the same file. He also learned how to use GitHub features such as merge, pull, and push effectively.
  + Yash – I learned how to efficiently and effectively create and apply algorithms to the csv dataset. I also learned how to interact with GitHub using its various features like pull requests.
  + Philip – I learned how to interact more efficiently with Github and use its various features in a collaborative way. In the past my experience with Git was quite limited and I now feel comfortable using and working on projects with others in GitHub. I also had to refresh how to interact and use csv data in C++.

## References

* <https://corgis-edu.github.io/corgis/csv/county_demographics/>
* https://www.geeksforgeeks.org/dsa/heap-sort/