CSC-151 Class Project Report: College Searching Algorithm

> ***Implemented by RyuuiChiro Sonoda, Masaki Nawa, and Austin Yu***

- Table of Contents

Description of Project Goals

We are writing a program that provides recommended lists of various colleges based on the user's input such as their SAT/ACT score, the location of the college, Tuition and fees, and the size of the school. The final report consists of a list of 10 colleges matching user's criteria and sorting from the highest acceptance rate to the lowest acceptance rate.

Description of data

Source of Data

> *We would be using a dataset, "College Admissions" from Kaggle.com*
>
> *https://www.kaggle.com/samsonqian/college-admissions*

Form of Data

The form of data is a table with comprehensive information for each college. Though the table contains dozens of different characters for a college, we would only use the following columns in this algorithm.

- Name
- Applicants total
- Admissions total
- SAT Critical Reading 25th percentile score
- SAT Critical Reading 75th percentile score
- SAT Math 25th percentile score
- SAT Math 75th percentile score
- SAT Writing 25th percentile score
- SAT Writing 75th percentile score
- ACT Composite 25th percentile score
- ACT Composite 75th percentile score
- Estimated freshman undergraduate enrollment, total
- Tuition and fees, 2013-14
- Total price for in-state students living on campus 2013-14
- Total price for out-of-state students living on campus 2013-14
- State

Type of Each Data Element

Column	Data Type
Name	String
Applicants Total	Integer
Admissions Total	Integer
SAT Critical Reading 25th percentile score	Integer
SAT Critical Reading 75th percentile score	Integer
SAT Math 25th percentile score	Integer
SAT Math 75th percentile score	Integer
SAT Writing 25th percentile score	Integer
SAT Writing 75th percentile score	Integer
ACT Composite 25th percentile score	Integer
ACT Composite 75th percentile score	Integer
Estimated freshman undergraduate enrollment, total	Integer
Tuition and fees, 2013-14	Integer
Total price for in-state students living on campus 2013-14	Integer
Total price for out-of-state students living on campus 2013-14	Integer
State	String

Range of Each Data Element

We could use the following procedure to produce the range of each column.

```
;;; Procedure:
;;;   range
```

```
;;; Parameters:
;;;   col: an integer
;;; Purpose:
;;;   return the range of values in a column
;;; Produces:
;;;;  result, a string
;;; Preconditions:
;;;   col is an integer between 1 and 14 inclusive
;;; Postconditions:
;;;   the range is a string in the form "minValue-maxValue"
(define range
  (lambda (col)
    (let ([data (filter number? (reduce append
                        (map list
                            (map (section list-ref <> col) data-selected))))])
      (letrec ([maxlst (lambda (lst)
                        (if (equal? 1 (length lst))
                            (car lst)
                            (max (car lst) (maxlst (cdr lst))))
                        )]
               [minlst (lambda (lst)
                        (if (equal? 1 (length lst))
                            (car lst)
                            (min (car lst) (maxlst (cdr lst))))
                        )])
      (string-append (number->string (minlst data)) "-"
                     (number->string (maxlst data)))
    ))))
```

Column	Range of Data
Name	n/a
Applicants Total	6142-72676
Admissions Total	5521-35815
SAT Critical Reading 25th percentile score	370-720
SAT Critical Reading 75th percentile score	450-800
SAT Math 25th percentile score	350-770
SAT Math 75th percentile score	450-800
SAT Writing 25th percentile score	480-720
SAT Writing 75th percentile score	600-800
ACT Composite 25th percentile score	15-33
ACT Composite 75th percentile score	19-35
Estimated freshman undergraduate enrollment, total	1104-10241
Tuition and fees, 2013-14	7182-49138
Total price for in-state students living on campus 2013-14	21849-64988
Total price for out-of-state students living on campus 2013-14	27441-64988
State	n/a

Description of algorithm(s)

Description of each algorithm

- `select-cols`: select the columns in a row of a table specified by the indexes provided
- `gen-size`: to determine the size of univeristies in the table.
- `gen-sat`: to calculate the 75th percetile SAT score of univeristies in the table.
- `gen-accpt`: to calculate the acceptance rate of univeristies in the table.
- `gen-cost`: to calculate the total cost of univeristies in the table.
- `sort-accpt`: sort table according to the acceptance rate.
- `fil-size`: take lists of colleges based on desired size and sort table according to the acceptance rate.
- `fil-sat`: take lists of colleges based on given SAT score and sort table according to the acceptance rate.
- `fil-act`: take lists of colleges based on given ACT score and sort table according to the acceptance rate.
- `fil-cost`: take lists of colleges based on given cost the user can afford and sort table according to the acceptance rate.
- `fil-loc`: take lists of colleges based on desired location and sort table according to the acceptance rate.
- Main Algorithm ->`search-college`: provide recommended lists of various colleges based on size, sat, act, cost, and loc.

Overview of how each were implemented

There are four steps basically.

- Step one: clean data
    - import the csv file as a table
    - select out columns that are useful
- Step two: process data
    - process the data to generate columns of size, sat score, acceptance rate, cost etc.
- Step three: helper procedures for main algorithm
    - write the filtering procedure for each of the parameters of main algorithm
- Step four: main algorithm
    - putting every thing together

Description of what analysis showed

Complete description of what analysis showed

We paid special attention to the parameters of our main algorithm. For the size of the college, we define small schools as those with enrollment lower than 500, medium schools as those with enrollment between 500 and 1000, and ‘large’ schools as those with enrollment larger than 1000. We choose this definition because we discover that almost 50% of schools in the dataset actually have enrollment lower

than 500. To balance this ratio, we set large schools to the enrollment larger than 1000, even though this threshold might seem too low for most colleges.

For SAT and ACT scores, we provide recommendations for schools whose SAT 75% percentile score is lower than that of the user. Though filtering out those schools with higher expected SAT score seems arbitrary, after balancing the choice of “safe” and “reach” school choices, we decide to recommend schools that are more secured according to the user’s score.

For the cost section, we are assuming that the user is searching for school as an out of state student. We are also not considering financial aid or scholarships.

Addresses any objections or bias that may have affected results

There could be many objections to this algorithm due to several of our decisions in designing this algorithm.

First, our definition of sizes for schools might seem too low when considering those large public colleges. However, since our dataset consists of almost 50% of schools with enrollment lower than 500, we need to balance that when we are designing the algorithm.

Second, people do get into colleges with a 75% percentile SAT score high than that of the student, which makes it arbitrary to exclude those colleges at once. After considering this problem, we decide to only recommend those safe schools.

Third, sorting the schools according to the acceptance rate might not be always helpful, but again, we are only recommending those safe school choices.

Instructions for running code

Running this algorithm is pretty simple. The only procedure called is `search-college`. By specifying the criteria for size, sat, act score, cost, and location, the algorithm will provide the 10 most secure school choices ranked according to acceptance rate.

This algorithm gives the user the freedom to ignore certain criteria by specifing the requirement with #f. For instance, if I would like to find a school with small size and in New York, I would simpy command `(search-college small #f #f #f "New York")`