CS340 - Registrar Project

Greg Van Aken, Conor Stuart Roe, Russell Gerhard

October 4, 2018

1 Checkpoint 1

1.1 Description

Let M denote an nxn 2-D matrix, where n is the number of classes. Initialize M such that all values are 0. Iterate over the teacher-class list. For each pair of classes (c, c') taught by the same teacher, set $M[c][c']$ to *Infinity*. Iterate over each student's preference list. For each class pair (c_i, c_j) in a given student's preference list, (where i and j represent indices/preferences) increment $M[c_i][c_j]$ by ((4-i) + (4-j)). In other words, compute the effective *cost* to each student of overlapping two courses. If i is 0 and j is 1, the student prefers both of those classes highly and their overlap will be a cost of 7, much higher than if i were 2 and j were 3 (cost of 3). Classes taught by the same teacher mustn't, by definition, occur at the same time, so their overlap cost is infinite. Assemble each class into its own disjoint set in a collection of sets S. While the size of $S >$ the number of class times, merge sets with the minimum overlap cost, according to M. When a merge takes place between two sets s and s' to form set s'', for each class pair c and c' in s'', for each class c'' in $M[c]$, increment $M[c][c'']$ by ($M[c][c'] + M[c''][c']$) and for each class c''' in $M[c']$, increment $M[c'][c''']$ by ($M[c][c'] + M[c'''][c]$). Distribute classes such that merged sets occur during the same time slot.

1.2 Pseudocode

Function `genMatrix`(*Preferences, numClasses, Teachers*)
  let n = len(numClasses); initialize M = n x n 2-D array of all 0's;
  M = setTeach(M, $Teachers$);
  M = setCost(M, $Preferences$);

```
Function setTeach(M, Teachers)
    sort Teachers by teacher id; // Teachers is a list of class-teacher pairs
    initialize last = Teachers[0];
    while i = 1 < len(Teachers) do
        let teach = Teachers[i];
        if teach[1] == last[1] then
            set M[teach[0]][last[0]] to infinity; // teacher teaches both classes
        end
        set last = Teachers[i]; i++;
    end
    return M
```

```
Function setCost(M, Preferences)
    for student in Preferences do
        // Assume Preferences = [[p1,p2,p3,p4],[p1,p2,p3,p4],...]
        for i in range(0:4) do
            for j in range(i+1:4) do
                let c1 = student[i]; let c2 = student[j];
                M[c1][c2] += ((4-i) + (4-j))
            end
        end
    end
```

1.3 Time Analysis

Time Analysis goes here. $O(n^2)$

1.4 Discussion

Collaboration

Name1, Name2, ...