Akarsh Kumar
ak39969
EE-360 Algorithms Assignment 1
Dr. Santacruz

---

Gale-Shapley Pseudocode:

---

This implementation of the Gale-Shapley algorithm puts students in a queue and goes through their preferred internship list until they find an internship that will accept them. They are temporarily paired until all students are matched or have reached the end of their list, in which case they are left unemployed. An internship accepts a student if it has room for a student or the student is better than their current worst student (tracked using a min heap).

```
// let M be the matching
// let M.m and M.n be the number of internships and students respectively
// let M.internshipSlots be the number of slots in each internship
// let M.studentPref be the student preference list
// let M.internPref be the internship preference list

Gale-Shapley(M):
      O = Optimization(M)

      studentQueue = [0, 1, 2, ... , M.n] // add all students
      studentMatching = [-1, -1, -1, ... , -1] // -1 for all students

      // the current index of internship preference the student is at
      studentCIPI = [0, 0, 0, ... , 0] // 0 for all students

      internshipMatchings = []
      for internship in (0, M.m):
                  internshipMatchings.add(new InternshipMatching(M, O,
internship))

      while studentQueue is not empty:
                  student = studentQueue.removeMin()
                  internshipIndex = studentCIPI[student]

                  matched = false

                  while not matched:
                              if internshipIndex is out of range of
internships:
                                    matched = true
                                    studentMatching[student] = -1
                              else:
                                    internship =
M.studentPref[student][internshipIndex]
```

```
                                    if
internshipMatchings[internship].canAdd(student):
                                                rejectedStudent =
internshipMatchings[internship].add(student)
                                                if rejectedStudent is not -
1:

      studentQueue.add(rejectedStudent)
                                                studentMatching[student] =
internship

      return studentMatching


class InternshipMatching:
      studentHeap = []

      init(M, O, internship):
                  initilize studentHeap as min heap to prioritize least
preferred students

      canAdd(student):
                  if studentHeap.size < number of slots in internship:
                              return true
                  else if studentHeap.peek() is less preferred than
student:
                              return true
                  else:
                              return false

      addStudent(student):
                  rejectedStudent = -1

                  studentHeap.add(student)

                  if studentHeap.size >= number of slots in internship
                              rejectedStudent = studentHeap.removeMin()

                  return rejectedStudent


//optimizes access times for finding the preference index in a preference
list
Optimization(M):
      O.invStudentPref = Invert all lists (M.studentPref)
      O.invInternPref = Invert all lists (M.internPref)
      return O

// inverts the given list and maps the value at an index to the index

Invert(list):
      invertedList = list
      for i in (0, list.length):
                  invertedList[list[i]] = i
```

```
      return invertedList


-------------------------------------------------------------------------------
--------------------

      ***Overall Time Complexity***
-------------------------------------------------------------------------------
--------------------

The overall time complexity of this program is:
O(m * n * log k) where m, n, k are the number of internships, students,
and the max number of slots an internship has respectively.

This complexity simplifies to O(m * n) if k << m, n.


-------------------------------------------------------------------------------
--------------------

      Time Complexity Explanation:
-------------------------------------------------------------------------------
--------------------
      - Invert takes O(n), given list size n
      - Optimization takes n*m + m*n = O(m*n), given number of students
and internships, n, m
      - Creating a InternshipMatching takes O(1)
      - GaleShapley takes O(Optimization) + O(3*n) + O(m) + O(main
matching loop)
      - main matching loop takes O(m * n * log k), where m is the number
of internships, n is the number of students, and k the largest number of
slots an internship has.
                  - this is because, at the worst case, the loops could
iterate through each student and their list of internship preferences (m *
n)
                  - within each of these, we are finding the the least
preferred student for a given internship. Using a min heap to keep track
of the least preferred students, this process will take log k time, where
k is the size of the heap, and worst case is equal to the number of slots
the internship has available.
                  - multiplying it out, we have that the main loop takes
O(m * n * log k) in time
```