Final Project

Josh Seligman

joshua.seligman1@marist.edu

October 25, 2022

1 Hospitals and Residents Stable Matching

1.1 The Algorithm

In the hospitals and residents stable matching problem, the goal is to assign residents to hospitals given the preferences of both sides so that all assignments are stable. In this context, the term "stability" means that for each resident, there is no hospital that is available that is higher on a resident's list compared to that resident's current assignment. The reason stability is in the terms of the residents is because the residents propose to the hospitals on their preference lists and the hospitals have the ability to either provisionally accept or reject the residents based on their resident preferences and current capacity. In other words, hospitals may have a preferred resident that is available, but that resident may not want to go to that hospital and, therefore, will end up elsewhere.

1.2 Asymptotic Analysis

The pseudocode for the hospitals and residents stable matching algorithm is provided in Algorithm 1. The algorithm starts off by assigning all residents and hospitals to be completely free, which are $O(r)$ and $O(h)$ operations, respectively. Next, line 8 is the condition for a while loop that runs until the residents list is empty. Since residents are being picked off one-by-one as done in line 9, the loop will run for each resident, which makes it run on average r times. Line 10, similar to line 8, also defines a while loop. This time, however, it is iterating over the resident's preferences, which means the while loop runs h times for each iteration of the outer loop. The if-statement block on lines 12-16 is inside of the inner loop, and contains a statement to get the least preferred assigned resident, which is an $O(r)$ operation as it may have to loop through all of the residents at worst case. The condition and other 2 assignments in the if-statement block are constant time assignments. The assignment on line 17 sets the assignment for the resident, which runs in constant time. Next, just like line 12, the check if the hospital is full a constant time check. Line 19 is the same as line 13 and is an $O(r)$ operation. Next, there is a loop defined on line 20 that iterates through the remaining residents in the list of hospital preferences, which at worst case is about r iterations. It also has an $O(r)$ operation that is executed once to get the starting index. Lastly, there is a constant time assignment on line 21, and the removal function calls are $O(h)$ and $O(r)$ operations, respectively, as the hospitals and residents are being iterated through in each call. Overall, when putting it all together, the runtime of the original stable matching problem for residents and hospitals is $r*h*(r+r+r*(h+r)) = rh*(2r+rh+r^2) = 2r^2h+r^2h^2+r^3h = O(r^2h+r^2h^2+r^3h)$,

Algorithm 1 Hospitals and Residents Stable Matching Algorithm

```
1: procedure StableMatchOriginal(residents, hospitals)
2:     for r of residents do
3:         r.assignment ← null    // Residents start off unassigned
4:     end for
5:     for h of hospitals do
6:         h.assignments ← [ ]    // Hospitals initially have no assignments
7:     end for
8:     while !residents.isEmpty() do
9:         r ← residents.dequeue()    // Get the next resident in line to be assigned
10:        while r.assignment == null && !r.preferences.isEmpty() do
11:            h ← r.preferences.dequeue()    // Try the resident's next top preference
12:            if h.isFull() then
13:                r′ ← h.getLeastPreferredAssignedResident()
14:                r′.assignment ← null    // Set the least preferred assigned resident to be free
15:                residents.enqueue(r)    // Add the resident back to the list to be reassigned
16:            end if
17:            r.assignment ← h    // Provisionally assign r to h
18:            if h.isFull() then
19:                s ← h.getLeastPreferredAssignedResident()
20:                for i ← h.preferences.indexOf(s) + 1, len(h.preferences) − 1 do
21:                    s′ ← h.preferences[i]
22:                    s′.preferences.remove(h)    // Remove h from preferences of s′
23:                    h.preferences.remove(s′)    // Remove s′ from preferences of h
24:                end for
25:            end if
26:        end while
27:    end while
28: end procedure
```

which can be simplified to $O(r^3)$ because there are typically a lot more residents than hospitals, so r^3 becomes the dominant term in the expression.

2 Appendix

2.1 Linear Search