Lanxin Yang
ly5592

1 Pseudocode for University Optimal

Algorithm 1 University Optimal G-S

```
Initially all universities u ∈ U and students s ∈ S are unmatched
while ∃u whose available opening(s)> 0 do
    while u's opening > 0 do                                          ▷ O(n)
        Let s be the favorite student in u's preference list to whom u has not sent an offer to
        if s didn't receive any offer then
            s is temporarily admitted by u                            ▷ Instability
            available opening of u decrease by 1
        else s was temporarily admitted by u'
            if s prefer u' than u then
                number of openings of u remain the same
            else s prefer u than u'
                s is temporarily admitted by u                        ▷ Instability
                available opening of u decrease by 1
                available opening of u' increase by 1
            end if
        end if
    end while
end while
for students s ∈ S do
    if s is matched with universities u then
        fill u to s's position in set S
    else students s is unassigned
        Fill −1 to the set S
        return S
    end if
end for
```

2 Run time of University Optimal Algorithm

The worst case is that each university has the same preference list, which is $[s_0, s_1, ..., s_n]$, and students have the same preference list $[u_m, u_{m-1}, ..., u_0]$. It will take mn operations to "reverse" students' preference list by following the same procedure we discussed to create a $invpref()$ for women's preference, in this case, we can compare students' preferences of universities in O(1). For the worst case, assume the available opening for university i is p_i, $\sum_{i=0}^{m} p_i = n$, while $p_0 \leq p_1 \leq p_2 \leq ... \leq p_m$, therefore, u_0 will be reject by $n - p_0$ times, u_1 will be reject by $n - p_0 - p_1$ times..., u_m won't be reject. The total runtime for matching will be:

$$(n - p_0) + (n - p_0 - p_1) + ... + (n - \sum_{i=0}^{m} p_i) < m(n - p_0) < mn$$

Therefore the **Big-O** run time for University Optimal Algorithm will be $O(mn)$

3 Pseudocode for Student Optimal

Algorithm 2 Student Optimal G-S

Initially all universities $u \in U$ and students $s \in S$ are unmatched
while $\exists u$ haven't been admitted by any university and has not applied to every university **do**
    Let u be the favorite university in s's preference list to which s has not applied to
    if u still have openings **then**
        s is temporarily admitted by u ▷ Instability
        available opening of u decrease by 1
    else$u$ openings are fully filled
        let s' be the least favorite student in u's current temporarily filled openings list
        if u prefer s than s' **then**
            s is temporarily admitted by u
            s' is rejected ▷ Instability
            sort u current admitted students by preference list and find the least preferred one s'
        else u prefer s' than s
            s is rejected ▷ Instability
        end if
    end if
end while
for students $s \in S$ **do**
    if s is matched with universities u **then**
        fill u to s's position in set S
    else students s is unassigned
        Fill -1 to the set S
        return S
    end if
end for

4 Run time of Student Optimal Algorithm

The worst case is that each university has the same preference list, which is $[s_n, s_{n-1}, ..., s_0]$, and students have the same preference list $[u_0, u_1, ..., u_m]$, while the opening for each university is 1. It will take mn operations to "reverse" universities' preference list by following the same procedure we discussed to create a $invpref()$ for women's preference, in this case, we can compare universities' preferences of students in O(1). After each time admitting a new student, it will take $O(n)$ to find the least preferred student. Therefore, for each student, it will at most be compared and rejected by m times. The total runtime for matching will be:

$$mn + 2m(n - m) + 2(m - 1) + 2(m - 2) + ... + 2(m - m) < mn + 2m(n - 1) < 3mn - 2m$$

Therefore the **Big-O** run time for University Optimal Algorithm will be $O(3mn - 2m) = O(mn)$