Programming Assignment - 1 Report
Submitted by :- Tushar Bansal (tb34826)

a) Pseudo Code for high school optimal

Construct a hashmap to store indices of student preferences for each school ie. what school is preferred at what number for each student. This helps reduce complexity of code later on. Now, iterating over the fact that there are some high schools with available spots, we iterate over a second loop to get each student in the preference list of such school. If the student has not been allotted a school, you assign this one and decrease the spots in the school and increase preference pointer by 1 (just in case we might have to come back to this school again). Preference pointer helps keep track of how many students might have been covered in the preference list of the school. If the student is already allotted, you check if the preference of student for that allottment is more than your new school. If the current allotment is preferred, you move to next pointer for this new school, otherwise you switch the two schools & decrease available slots for new, increase for current & move the tracker of preference as well. If the current school was filled right now, you also make sure you add it back to the list of schools with available slots.

b) Hashmap to keep track of preference of school for each student has complexity of $O(mn)$ because first you iterate over each student & then each school. For the main code, we are only doing deletions and insertions inside one loop at indexed locations or using linked lists to add new elements at the end. All of these have a complexity of $O(1)$.

However there are 2 loops running, one iterating over schools and other on students, and since they are nested, the complexity becomes $O(mn)$. It is because the operation is upper bound by having to access each school & then each student which can take $m*n$ turns.
Since Hashmap runs differently than the main code and they both have complexity of $O(mn)$, hence total complexity is $O(mn)$.

c) Pseudo code for student optimal.

In this case as well, I made a HashMap to retrieve priorities a student is assigned by different highschools to compare later on. This reduces the complexity of the code a lot. Initial code is run on students which are left unmatched and for each of these students, making sure that they are unmatched, each highschool is considered which is highest in their preference. If that high school has a spot, then the student is assigned to it and spots in the school are decreased. If not, then we check for all the students that school has, if it prefers this new student, it is assigned and the least preferred one is removed and added to list of students with no matching. If total spots are equal to no. of students, then the ending condition is that all students need to be matched. If they are unequal, then the ending condition is that each student who has not been matched has gone through all the School.

d) Hashmap has complexity $O(mn)$. First we iterate on students and then nested iterations are on schools, just like the case above. Since, in this case we also have to make sure that we know how each school prefers its allocated students, we have to also run through the spots taken up by each school to get least preferred student for comparison with a new student. Hence, that loop also comes in and its worst case iterations will be equal to maximum no. of spots a school has available (ie having to go through all the spots for one school). Hence, the complexity becomes $O(mnp)$ where p denotes maximum no. of slots in a school.

Special case: When there is only 1 school, and it has n spots for n students, this becomes $p = n$ & $O(1 \times n \times n) = O(n^2)$ just like what we discussed in class.