

**CS263**

Lecturer : Dr.Ashish Phophalia



# End-Sem

# Project

# JEE Seat Allocation

**Indian Institute of Information Technology Vadodara - ICD**

---

Alex Tamboli - 202011071

Tarun Mali - 202011073

Vivek Borole- 202011018

Suyash Rajput - 202011070

Hari Jarupla - 202011024

---

---

## CONTENT

- I. Problem statement
  - II. Approach
    - A. Generating the college data
    - B. Seat class
    - C. Snippet of “seat allocation on category”
    - D. Making the candidate preference array for all the seats
    - E. Important note
    - F. Snippet of “Build the preference list of the College seats”
    - G. Candidate class
    - H. Snippet of “Candidate setup” method
    - I. Initializing the seatPrefList[] associated with each candidate
    - J. Stable Marriage Problem
    - K. Gale-Shapley Algorithm
  - III. Output
    - A. Seat allocation on “AIR” alone
    - B. Seat allocation on category
  - IV. Gale-Shapley Algorithm
    - A. Algorithm
    - B. Runtime Analysis
    - C. Time complexity
  - V. Inspiration
  - VI. Citations
-

---

**Problem statement :**

JEE Seat Allocation Problem states that given  $N$  students and  $N$  Seats of different Colleges, where each Student has an AIR rank and Category rank. Each Student has a preference Order of Colleges. Algorithms task is to find the right College for individual Student by comparing individual's Rank and preference Order of all the Students.

Consider the following example.

Candidate 1 and Candidate 2 having rank 1 and rank 2 have the best ranks, they can easily get colleges of their first choice (5,4,2,3,1) and (4,5,1,2,3) respectively.

So their Seat allotted would be in College 5 and College 4 respectively.

---

## Approach :

Consider the problem statement Example,

⇒

Here, rank and category of the Student can be generated randomly. So the only input we need to take is the preference order of that Student.

The Actual seat allocation part will be filled by 40%GEN, 30%OBC, 20%SC and 10%ST.

## Generating the college data

We are dealing with N=50 seats distributed equally among 5 colleges. Further these 10 colleges are distributed in 40%-30%-20%-10% among the colleges. We have taken these symmetrical data to make our problem a little more easier.

## The Seat class

```
class Seat{
    int Id;
    int category;// 0 = general, 1 = obc, 2 = sc, 3 = st
    int[] candidatePrefList;
}
```

We are going to give each seat available in the counselling an unique id, other than this it will also have a category and each seat will have an array of candidate preference choice. We have made seats in a such a way that it will easy to find the associated college with the seat id, for example the college 1 have seats which are numbered 0 to 9, the first 4 seats are general seats, the next 3 are obc, the next 2 are sc and the last one is a st seat. We wrote this hard coded code to categorize the seats.

## Snippet of “Seat allocation on category” part of the Code :

```
if((i>=0 && i<4) || (i>=10 && i<14) || (i>=20 && i<24) || (i>=30 && i<34) || (i>=40 && i<44)) {
    seats[i].category = 0;
}
else if((i>=4 && i<7) || (i>=14 && i<17) || (i>=24 && i<27) || (i>=34 && i<37) || (i>=44 && i<47)) {
    seats[i].category = 1;
}
else if((i>=7 && i<9) || (i>=17 && i<19) || (i>=27 && i<29) || (i>=37 && i<39) || (i>=47 && i<49)) {
```

```

        seats[i].category = 2;
    }
    else {
        seats[i].category = 3;
    }
}

```

### Making the candidate preference array for all the seats

For general seats, this task is easy, but for category seats, we have to first sort the candidates in ascending order of their category rank and fill the array. For the remaining candidates, we have to sort them by their general ranks and fill the rest of the candidate preference array, we also have to make sure that none of the category students gets repeated. For example for an obc sort, sort the candidates with their obc rank and fill the preference array, after that sort the other candidates according to their general rank and fill the remaining array, making sure no obc candidate got repeated in the process, do similarly for all the category students.

**Important thing to note** is that we don't need the individual category ranks for building the preference lists of the seats.

Just knowing what is the category of the candidate will allow us to build the preference list, for example, the obc having better general ranks will definitely have better obc rank, so we need not to know the obc ranks.

### Snippet of “Build the preference list of the College seats” part of the code :

```

for(int i = 0; i < seats.length; i++) {
    seats[i].candidatePrefList = new int[seats.length];

    if(seats[i].category == 0) {
        for(int j = 0; j < seats[i].candidatePrefList.length; j++) {
            seats[i].candidatePrefList[j] = j+1;
        }
    }
    else if(seats[i].category == 1) {
        int count = 0;
        for(int j = 0; j < noOfCandidates; j++) {
            if(candidates[j].category == 1) {
                seats[i].candidatePrefList[count] = candidates[j].air;
                count++;
            }
        }
    }
}

```

```

    }
    else continue;
}
int count2 = 0;
for(int j = 0; j < noOfCandidates; j++) {
    if(candidates[j].category == 1) continue;
    else {
        seats[i].candidatePrefList[count+count2] = candidates[j].air;
        count2++;
    }
}
}
else if(seats[i].category == 2) {
    int count = 0;
    for(int j = 0; j < noOfCandidates; j++) {
        if(candidates[j].category == 2) {
            seats[i].candidatePrefList[count] = candidates[j].air;
            count++;
        }
        else continue;
    }
    int count2 = 0;
    for(int j = 0; j < noOfCandidates; j++) {
        for(int j = 0; j < noOfCandidates; j++) {
            if(candidates[j].category == 2) continue;
            else {
                seats[i].candidatePrefList[count+count2] = candidates[j].air;
                count2++;
            }
        }
    }
}
else {
    int count = 0;
    for(int j = 0; j < noOfCandidates; j++) {
        if(candidates[j].category == 3) {
            seats[i].candidatePrefList[count] = candidates[j].air;
            count++;
        }
        else continue;
    }
    int count2 = 0;
    for(int j = 0; j < noOfCandidates; j++) {
        if(candidates[j].category == 3) continue;
        else {
            seats[i].candidatePrefList[count+count2] = candidates[j].air;
            count2++;
        }
    }
}
}

```

## The Candidate class

```
class Candidate{
    int air;
    int category;// 0 = general, 1 = obc, 2 = sc, 3 = st
    int[] seatPrefList = new int[50];
}
```

Each candidate has a unique air (no two candidates have the same air for simplicity). Now we have to initialize the category of all the 50 candidates, we will do this randomly through the function **candidatesSetUP()**

### Snippet of “Candidate setup” method of the code :

```
private static void candidatesSetUP() {
    //giving random category to candidates
    for(int i = 0; i < candidates.length; i++) {
        candidates[i] = new Candidate();
        candidates[i].air = i+1;

        Random random = new Random();
        int buffcat = random.nextInt(4);

        candidates[i].category = buffcat;
    }
}
```

We are allocating categories randomly by using the

```
random.nextInt();
```

### Initializing the seatPrefList[] associated with each candidate :

We can generate the order manually through input or randomly, in any case, the seats are filled in the seatPrefList after that, for example the order of college we took is 5 3 1 2 4, then the array will look like

#### Preference list of Air 1

90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63

Once we have generated all Students rank ,category and have their preference order of Colleges, we can start comparing Each student's preference order to that of others and their category ranks to assign Colleges in a way that would satisfy everyone's preference order.

This is similar to the **Stable-Marriage Problem**, where the “Student’s preference order” role is played by Man and “College Seat” role is played by women.



### → Stable Marriage Problem :

Assume that there are the same number of Men and Women. Each man has a preference list which contains the order in which he likes the Women. We have the same list for each woman.

Now, we want to pair the Men and Women in such a way that there is no unstable pair.

What is an unstable pair?

Suppose man B1 and woman G1 are paired together.

Also, assume that B2 and G2 are paired together.

Now, if B1 likes G2 more than G1 and also, G2 likes B1 more than B2, then B1 and G2 will run away.

The **Gale-Shapley Algorithm** attempts to find a stable matching - a matching with no unstable pairs.

### → Gale-Shapley Algorithm :

The idea is to iterate through all free men while there is any free man available.

Every free man goes to all women in his preference list according to the order.

For every woman he goes to, he checks if the woman is free, if yes, they both become engaged. If the woman is not free, then the woman either says no to him or dumps her current engagement according to her preference list.

So an engagement done once can be broken if a woman gets a better option.



The modified explanation would be iterating through every Students preference list and checking if a particular college seat is available for his/her category rank.

Time Complexity of the Gale-Shapley Algorithm is  $O(n^2)$ .

---

---

**Testing Example** : All students have same preference list, without category

**Expected output** : College Seat allocation will be on AIR alone.

---

**Screenshot** : as we can see if we remove category variable then Seat allocation will be based on AIR alone.



---

**Example 1 :** All students have same preference list, with category

**Expected output :** College Seat allocation will be on Category.

---

**Screenshot :**

```
Enter the number of candidates you want to enter preference list manually
For the rest, the preference list will be generated automatically
=> 2
no of input: 2
Enter the order in which you want to choose the colleges for the candidate 1
1 for IIIT Vadodara
2 for IIIT Delhi
3 for IIITV ICD
4 for IIIT Allahabad
5 for MNIT Allahabad
5 4 2 3 1
Enter the order in which you want to choose the colleges for the candidate 2
1 for IIIT Vadodara
2 for IIIT Delhi
3 for IIITV ICD
4 for IIIT Allahabad
5 for MNIT Allahabad
4 5 2 1 3
```

**Explanation of this input and output:**

As the candidate 1 and candidate 2 having rank 1 and rank 2 have the best ranks, they can easily get colleges of their first choice which is college 5 and 4 respectively.

AIR	College
1	5
2	4
3	3
4	1
5	3
6	5
7	2
8	1
9	1
10	2
11	5
12	4
13	4
14	3
15	5
16	1
17	1
18	2
19	1
22	5
23	1
24	3
25	3
26	5
27	4
28	3
29	5
30	3
31	3
32	1
33	3
34	3
35	2

---

## Algorithm : Gale-Shapley

---

start

```
1: while there is an unpaired man do
2:   pick an unpaired man X and the first woman w on his list
3:   remove w from his list so it won't be picked again
4:   if w is engaged then
5:     if w prefers X more than her current partner Y then
6:       set X-w as married
7:       set Y-w as unmarried so now Y is unpaired
8:     else
9:       X is still unpaired since w is happier with Y
10:  end if
11: else
12:  the woman was not previously paired so accept immediately, X-w, as married
13: end if
14: end while
```

end

---

---

### Runtime Analysis :

The algorithm begins with  $\Theta(n^2)$  time to set up the needed data structures, then run the main proposal loop until there are no more free women. The remaining runtime analysis will be for this loop.

⇒ There are no more than  $n^2$  proposals during a run of the Gale-Shapley algorithm.

*Proof.* No woman proposes to a man more than once. If all  $n$  women propose to all  $n$  men, there would be  $n^2$  proposals. Since no woman proposes to a man more than once, there can't be any additional proposals.

⇒ Each proposal can be performed in  $O(1)$  time.

*Proof.* This can be achieved by using certain data structures to maintain the needed information. Comparing preferences could take  $O(n)$  time if we only use the preference lists, but by storing an array of preferences indexed by people this can be performed in constant time. Setting up this array took  $\Theta(n^2)$  time during setup, but results in a faster runtime within this loop.

Since the setup takes  $\Theta(n^2)$  and there are at most  $n^2$  proposals each taking  $O(1)$  time, the Gale-Shapley algorithm has a runtime of  $O(n^2)$ .

---

### Time Complexity : $O(n^2)$

#### Some associated probabilities :


Since there are  $n!$  different lists, the probability that a man will get a particular sequence is  $1/n!$ .

---

---

## Inspiration :

### What is the DA algorithm that is being applied for the seat allocation in JoSAA and how exactly is the process working?



**Aman Goel**, B.Tech Computer Science and Engineering, Indian Institute of Technology, Bombay (2017)  
Answered Jul 21, 2015 · Upvoted by Rishabh Vatsa, B Tech Instrumentation and Control & Robotics, Dr. B. R. Ambedkar National Institute of Technology, Jalandhar... and Shail Daswani, Undergraduate Student at IIT Kharagpur

Originally Answered: What is the DA Algorithm that is being applied for the seat allocation in JoSAA and how exactly is the process working?

Nice question.

I did a project in my third semester to design the entire JEE seat allocation portal. The algorithm that works here is Gayle Shapley Algorithm.


Let's talk about the algorithm in an altogether different context.


Assume that there are some number of boys and there are same number of girls. Each boy has a preference list which contains the order in which he likes the girls. We have the same list for each girl.


Now, we want to pair the boys and girls in such a way that there is no unstable pair. Now what is an unstable pair?


Suppose boy B1 and girl G1 are paired together. Also, assume that B2 and G2 are paired together. Now, if B1 likes G2 more than G1 and also, G2 likes B1 more than B2, then B1 and G2 will run away :P


The Gale Shapley algorithm attempts to find a stable matching - a matching with no unstable pairs.


 386



 4

 15





We have seen this Quora answer by Aman Goel (one of the famous mentors for Jee and making a career in the CS field) where he has described this idea, unfortunately he didn't provide the source code for the same. So, we have come up with the complete code from scratch.

---

---

## ***Citations :***

### **Stable Marriage Problem - GeeksforGeeks**

*Stable Marriage Problem - GeeksforGeeks* (2013).

Available at: <https://www.geeksforgeeks.org/stable-marriage-problem/>

(Accessed: 10 December 2021).

### **Anon**

(2021) *Youtube.com*.

Available at: [https://www.youtube.com/watch?v=0m\\_YW1zVs-Q&t=341s](https://www.youtube.com/watch?v=0m_YW1zVs-Q&t=341s)

(Accessed: 10 December 2021).

### **Anon**

(2021) *Community.wvu.edu*.

Available at: <https://community.wvu.edu/~krsbramani/courses/fa01/random/lecnotes/lecture5.pdf>

(Accessed: 10 December 2021).

### **What is the DA algorithm that is being applied for the seat allocation in JoSAA and how exactly is the process working?**

*What is the DA algorithm that is being applied for the seat allocation in JoSAA and how exactly is the process working?* (2021). Available at:

[https://www.quora.com/What-is-the-DA-algorithm-that-is-being-applied-for-the-seat-allocation-in-JoSAA-and-how-exactly-is-the-process-working/answer/Aman-Goel-9?ch=10&oid=14193868&share=e48a083a&srid=hrT0tK&target\\_type=answer](https://www.quora.com/What-is-the-DA-algorithm-that-is-being-applied-for-the-seat-allocation-in-JoSAA-and-how-exactly-is-the-process-working/answer/Aman-Goel-9?ch=10&oid=14193868&share=e48a083a&srid=hrT0tK&target_type=answer)

(Accessed: 10 December 2021).

---