1. Assume that you are organizing a party for N people and have been given a list L of people who, for social reasons, should not sit at the same table. Furthermore, assume that you have C tables (that are infinitely large). Write a function layout(N,C,L) that can give a table placement (ie. a number from 0 . . . C − 1) for each guest such that there will be no social mishaps. For simplicity we assume that you have a unique number 0 . . . N − 1 for each guest and that the list of restrictions is of the form [(X,Y), ...] denoting guests X, Y that are not allowed to sit together. Answer with a dictionary mapping each guest into a table assignment, if there are no possible layouts of the guests you should answer False.
   USN-1NT18CS133

In [1]:
```
def backtrack(x,enemy_list,domain,assigned):
    if -1 not in assigned:
        return x
    v = 999
    for i in range(len(domain)):
        if v>len(domain[i]) and assigned[i]!=1:
            v = i
    order=[]
    for i in domain[v]:
        mini = 1000
        for j in enemy_list[v]:
            temp = len(domain[j])
            if i in domain[j]:
                temp-=1
            if temp<mini:
                mini = temp
        order.append((i,mini))
    order = sorted(order,key=lambda x:x[1],reverse=True)
    ordered = [i[0] for i in order]
    for i in ordered:
        newdomain = [ [j for j in i] for i in domain]
        for j in enemy_list[v]:
            if i == x[j]:
                continue
        x[v] = i
        assigned[v] = 1
        newdomain[v] = [z for z in newdomain[v] if z==i]
        temp = []
        for j in range(len(newdomain)):
            if j!=v and j in enemy_list[v]:
                newdomain[j] = [z for z in newdomain[j] if z!=i]
        res = backtrack(x,enemy_list,newdomain,assigned)
        if res!=0:
            return res
    x[v] = ""
    assigned[v] = -1
    return 0
people = int(input("Enter the number of people"))
tables = int(input("enter the number of tables"))
edges = []
line = input("enter elements of list L(people who should not sit together) till an empty newline charac
ter. ").split()
while(line):
     edges.append((int(line[0]),int(line[1])))
     line = input().split()
x = ["" for i in range(people)]
enemy_list = [[] for i in range(people)]
for i in edges:
    enemy_list[i[0]].append(i[1])
    enemy_list[i[1]].append(i[0])
for i in range(people):
    j = list(set(enemy_list[i]))
    enemy_list[i] = j
assigned = [-1 for i in range(people)]
domain = [[x for x in range(tables)] for i in range(people)]
res = backtrack(x,enemy_list,domain,assigned)
if res == 0:
    print('False')
else:
    for i in range(len(res)):
        print(' {} :'.format(i),res[i])
```

```
Enter the number of people8
enter the number of tables3
enter elements of list L(people who should not sit together) till an empty newline character. 0 2
0 3
0 4
1 4
1 7
2 3
2 6
3 7
3 4
4 7
5 6

 0 : 0
 1 : 1
 2 : 2
 3 : 1
 4 : 2
 5 : 1
 6 : 0
 7 : 0
```

In [ ]: