IT 300 Laboratory problems

Problems on stable matching algorithm

1. Implementation of a stable marriage problem.

   The Stable Marriage Problem states that given N men and N women, where each person has ranked all members of the opposite sex in order of preference, marry the men and women together such that there are no two people of opposite sex who would both rather have each other than their current partners. If there are no such people, all the marriages are “stable”.

   Consider the following example.
   Let there be two men m1 and m2 and two women w1 and w2.
   Let m1‘s list of preferences be {w1, w2}
   Let m2‘s list of preferences be {w1, w2}
   Let w1‘s list of preferences be {m1, m2}
   Let w2‘s list of preferences be {m1, m2}

   The matching { {m1, w2}, {w1, m2} } is not stable because m1 and w1 would prefer each other over their assigned partners. The matching {m1, w1} and {m2, w2} is stable because there are no two people of opposite sex that would prefer each other over their assigned partners.

2. Given a set of preferences among hospitals and medical students, implement a stable matching admissions process.

   Let's consider the Unstable pair for Hospital h and student s form an unstable pair if both:

   - h prefers s to one of its admitted students
   - s prefers h to assigned hospital.

   Stable assignment: Assignment with no unstable pairs.
   Input: A set of n hospitals H and a set of n students S.

   - Each hospital h $\in$ H ranks students.
   - Each student s $\in$ S ranks hospitals.

Hospital	1st	2nd	3rd
Delhi	Deepak	Manoj	Pankaj
Mumbai	Manoj	Deepak	Pankaj
Banglore	Deepak	Manoj	Pankaj

Hospital preferences list

Student	1st	2nd	3rd
Deepak	Mumbai	Delhi	Banglore
Manoj	Delhi	Mumbai	Banglore
Pankaj	Delhi	Mumbai	Banglore

Student preferences list

A perfect matching M : {Delhi-Pankaj, Mumbai-Manoj, Bangalore-Deepak}

3. Implementation of propose and reject algorithm using Gale-shapley approach.

- Let's Men propose to women in decreasing order of preference. Once a woman is matched, she never becomes unmatched;
- Algorithm terminates after at most n^2 iterations of the while loop. Each time through the while loop a man proposes to a new woman. There are only n^2 possible proposals.

Men	1st	2nd	3rd	4th	5th
V	A	B	C	D	E
W	B	C	D	A	E
X	C	D	A	B	E
Y	D	A	B	C	E
Z	A	B	C	D	E

Men's preferences list

Women	1st	2nd	3rd	4th	5th
A	W	X	Y	Z	V
B	X	Y	Z	V	W
C	Y	Z	V	W	X
D	Z	V	W	X	Y
E	V	W	X	Y	Z

Women's preferences list

Perfect matching M: {V-E, W-A, X-B, Y-C, Z-D}

Problems on Divide and Conquer algorithm

Q1. You have a shopping list, and your friend is telling you to grab them in 20 minutes. He also gives you priorities, so you need to grab them first.

Shopping list (**Item/Priority**);

1. Eggs (4)
2. Bread (2)
3. Milk (6)
4. Water (3)
5. Meat (1)
6. Detergent (5)

For small lists, it is easy to seek through with eyes and follow priority numbers. But think about whether He gives you a list of items with a count of 128? What are you going to do? Would you check the entire list to find the next item? NO!

So you will be reordering the list by priority, by comparing them on the first element which is eggs. Implement the above procedure.

Solution: quick sort approach

STEP 1: Move more priority elements before eggs and less priority ones after eggs.

1. bread (2) - Group A
2. water (3) - Group A
3. meat (1) - Group A
4. eggs (4) - Pivot
5. milk (6) - Group B
6. detergent (5) - Group B

Then you are doing same thing for each group:

STEP 2: Group A

1. meat (1)
2. bread (2) - Pivot
3. water (3)

Group A is done!

STEP 3: Group B

4. detergent (5)
5. milk (6) - Pivot

Group B is done.

So what do we have?

1. Group A ordered
2. Pivot (in place where it needs to be)
3. Group B ordered

STEP 4: Write them all

1. meat (1)
2. bread (2)
3. water (3)
4. eggs (4)
5. detergent (5)
6. milk (6)

For 6 items, only with 4 steps you sorted your shopping list by priority.

If it would have 128 items, there will be average 128*(7) = 896 iterations to sort them all. Not bad if you compare the typical selection sort which is 128^128 which is terrible.

Q2. Suppose you had 20 years of stock market data in 300 files. you wanted to combine the files and remove duplicate data. Some of the files were not properly sorted by time. Each file was about 150MB, so you could not load all of the data into RAM at once. you only have 32GB of ram. you could use virtual memory, but it would cause lots of swapping and bring the system to its knees. How are you going to implement it?

Solution: Merge sort approach. Actually you should first sort each file individually using in-memory quicksort and then recursively merge them pairwise, removing duplicates with each merge. Done in a few seconds. Not strictly a merge sort, but based on the same concept.

Q3. If you want to divide a long loaf of bread into 8 or 16 equal pieces, generally people cut it into two equal halves first and then cut each half into two equal halves again, repeating the process until you get as many pieces as you want - 8, 16, 32, or whatever. Almost nobody tries to divide the loaf into 8 pieces all at once - people can guess halves much better than eighths. Implement this.

Solution: Divide and conquer approach

Q4. Use the divide-and-conquer integer multiplication algorithm to multiply the two binary integers 10011011 and 10111010.