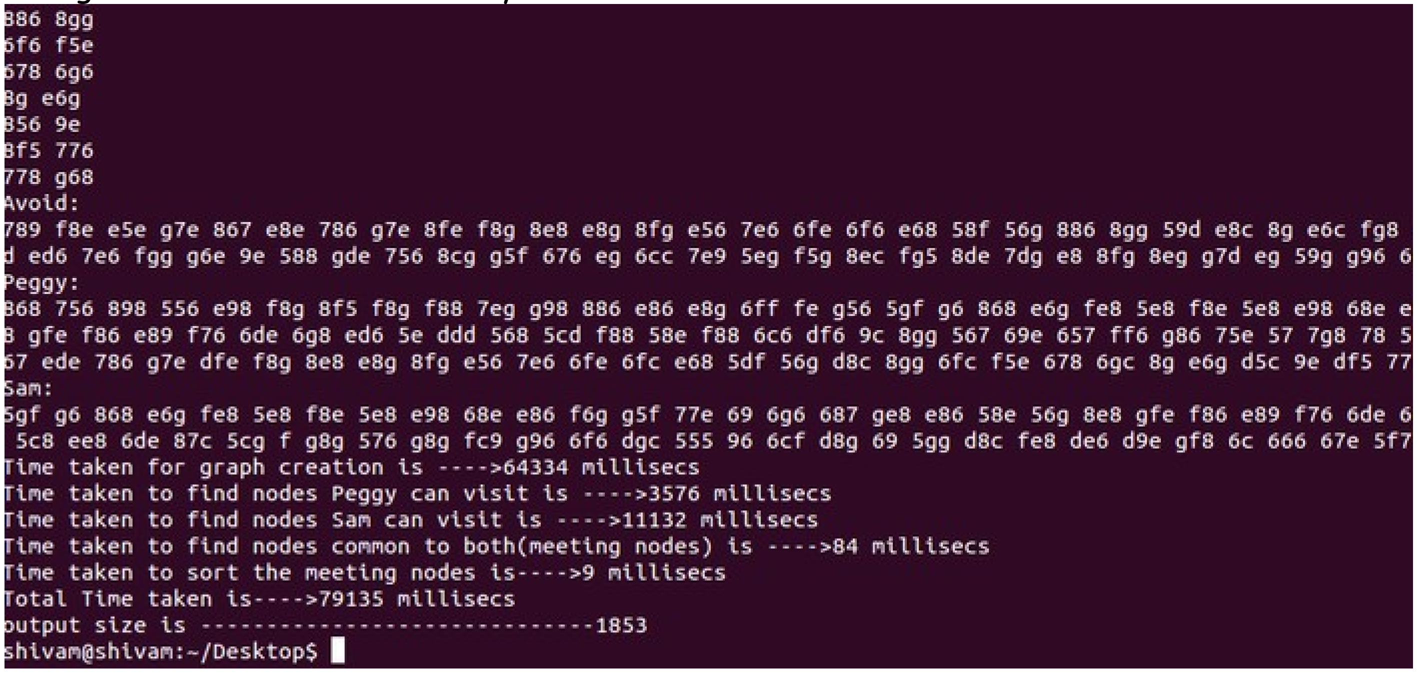
500K edges result is used in the analysis.

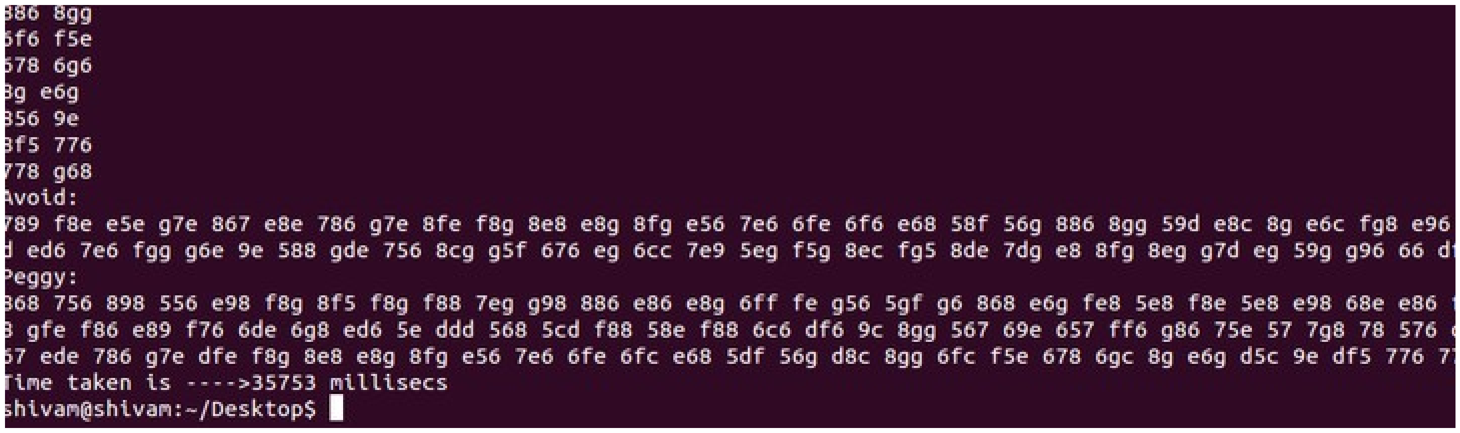


```
886 8gg
6f6 f5e
678 6g6
8g e6g
856 9e
8f5 776
778 g68
Avoid:
789 f8e e5e g7e 867 e8e 786 g7e 8fe f8g 8e8 e8g 8fg e56 7e6 6fe 6f6 e68 58f 56g 886 8gg 59d e8c 8g e6c fg8
d ed6 7e6 fgg g6e 9e 588 gde 756 8cg g5f 676 eg 6cc 7e9 5eg f5g 8ec fg5 8de 7dg e8 8fg 8eg g7d eg 59g g96 6
Peggy:
868 756 898 556 e98 f8g 8f5 f8g f88 7eg g98 886 e86 e8g 6ff fe g56 5gf g6 868 e6g fe8 5e8 f8e 5e8 e98 68e e
8 gfe f86 e89 f76 6de 6g8 ed6 5e ddd 568 5cd f88 58e f88 6c6 df6 9c 8gg 567 69e 657 ff6 g86 75e 57 7g8 78 5
67 ede 786 g7e dfe f8g 8e8 e8g 8fg e56 7e6 6fe 6fc e68 5df 56g d8c 8gg 6fc f5e 678 6gc 8g e6g d5c 9e df5 77
Sam:
5gf g6 868 e6g fe8 5e8 f8e 5e8 e98 68e e86 f6g g5f 77e 69 6g6 687 ge8 e86 58e 56g 8e8 gfe f86 e89 f76 6de 6
 5c8 ee8 6de 87c 5cg f g8g 576 g8g fc9 g96 6f6 dgc 555 96 6cf d8g 69 5gg d8c fe8 de6 d9e gf8 6c 666 67e 5f7
Time taken for graph creation is ---->64334 millisecs
Time taken to find nodes Peggy can visit is ---->3576 millisecs
Time taken to find nodes Sam can visit is ---->11132 millisecs
Time taken to find nodes common to both(meeting nodes) is ---->84 millisecs
Time taken to sort the meeting nodes is---->9 millisecs
Total Time taken is---->79135 millisecs
output size is -----------------------------1853
shivam@shivam:~/Desktop$ █
```

The code consists of 5 parts:
1) the first part scans all the input,it creates the internal graph representation. Nodes are created representing vertices of the graph,the node object has two lists downodes and upnodes,keeping the list of nodes that lie on downstream and upstream respectively.
The lists consisting of nodes to avoid,starting position of peggy and starting position of sam are also created.

Complexity of this part is:O(edge X n/2)=O(n^3)-because there are n vertices,so possibly there can be n^2 edges and for both the nodes in each edge they have to be checked whether they already exist in nodes list or not,so O(n/2).

It is the most time consuming of all the steps. Approximately 82% time is used in this step.64.3 seconds used in total 79.13 seconds.
In actuality a lot of the time is taken to read the input rather than logic,according to my calculation approximately-35 seconds were taken plainly to read the input.
So time taken in this step's logic for 35k edges is 64.3-35=34.3 seconds.



```
386 8gg
5f6 f5e
578 6g6
8g e6g
856 9e
8f5 776
778 g68
Avoid:
789 f8e e5e g7e 867 e8e 786 g7e 8fe f8g 8e8 e8g 8fg e56 7e6 6fe 6f6 e68 58f 56g 886 8gg 59d e8c 8g e6c fg8 e96
d ed6 7e6 fgg g6e 9e 588 gde 756 8cg g5f 676 eg 6cc 7e9 5eg f5g 8ec fg5 8de 7dg e8 8fg 8eg g7d eg 59g g96 66 d
Peggy:
868 756 898 556 e98 f8g 8f5 f8g f88 7eg g98 886 e86 e8g 6ff fe g56 5gf g6 868 e6g fe8 5e8 f8e 5e8 e98 68e e86
8 gfe f86 e89 f76 6de 6g8 ed6 5e ddd 568 5cd f88 58e f88 6c6 df6 9c 8gg 567 69e 657 ff6 g86 75e 57 7g8 78 576
57 ede 786 g7e dfe f8g 8e8 e8g 8fg e56 7e6 6fe 6fc e68 5df 56g d8c 8gg 6fc f5e 678 6gc 8g e6g d5c 9e df5 776 77
Time taken is ---->35753 millisecs
shivam@shivam:~/Desktop$ █
```

Code for above is:

```
import java.util.*;
import java.util.Collections;


class A{
public static void main(String args[]){
Scanner reader = new Scanner(System.in);
long startTime = System.currentTimeMillis();
int count=465847;
while (count>1)
{

        String nextline=reader.nextLine();
        count=count-1;

}
long endTime = System.currentTimeMillis();

long duration1 = (endTime - startTime);
System.out.println("Time taken is ---->"+duration1+" millisecs");

}
}
```

2)In this part all the nodes peggy can reach are calculated. First a queue is initialized with the starting nodes of peggy,then each node is accessed and checked whether it is not included in avoid nodes list or not and whether it has been already included in the nodes peggy can visit list or not(both the queue and final list).If it passes both the test,than all the nodes which are present in downstream-nodes list of this node are put in the queue and this node is moved to the final list of nodes peggy can visit. This process is repeated until the queue is empty.

Complexity:
(n/2 + |queue| +|nodes to avoid|)*|queue|~O(n/2*(some fraction of n))
Time taken in this step was:3.567 sec(total-79.13) which is 4.5 % of total time taken.
The time taken by this step is extremely variable depending on how many nodes are there which peggy can visit and branching factor of nodes

3)In this part all the nodes sam can reach are calculated,it is exactly same as part2 just starting nodes of sam are considered and for each node upnodes are put in the queue. First a queue is initialized with the starting nodes of sam,then each node is accessed and checked whether it is not included in avoid nodes list or not and whether it has been already included in the nodes sam can visit list or not(both the queue and final list).If it passes both the test,than all the nodes which are present in upstream-nodes list of this node are put in the queue and this node is moved to the final list of nodes sam can visit. This process is repeated until the queue is empty.

Complexity:
(n/2 + |queue| +|nodes to avoid|)*|queue|~O(n/2*(some fraction of n))
Time taken in this step was:11.13 sec(total-79.13) which is 14% of total time taken.
The time taken by this step is extremely variable depending on how many nodes are there which samy can visit and branching factor of nodes.

4)In this part the list of nodes which are common between nodes that can be visited peggy and the nodes that can be visited by sam is created. One node that peggy can visit is matched with all nodes sam can visit,if it matches it is added to final output list. I could have used binary search and other efficient techniques but for it,i have to sort both
the lists and it will take-up advantage of using binary search in place of linear search,and also its contribution to total time is very small.

Complexity:
Its complexity is O(|peggy possible loc list|*|sam possible loc list|/2)
This step takes 84 milliseconds which is less than .0011% of total computational time.(79.13 sec)

5)In this part the list consisting of all the valid nodes that peggy and sam can visit is sorted. I have used java's inbuilt collection.sort in place of efficient algos. taking O(n*logn) like quicksort because I implemented it and difference was coming of mere 2-3 millisec.

Complexity:|output list|^2
time taken by this step is 9 millisec which is .00011% of total time.(79.13 secs)

Efficient code by code explanation is given in code_with_comments folder's file Solution.java.
I have attached snapshots of time taken by inputs consisting of total edges-original input,2.5k,20K,100K,500K.