ITP 20002-03 Discrete Mathematics, Fall 2021

Homework 4

Your name : Seokjun Kim(김석준)

Email address : 21600081@handong.edu

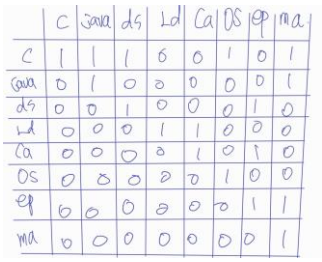
Drawing Hasse diagrams

1. Introduction

This program is a program that draws Hasse diagrams. A given input file that represents a binary relation, a Hasse diagram representing its closure of partial order relation should be output from image file. This binary relation consists of ordered pairs, and since it is a partial order relation, it is reflexive, antisymmetric, and transitive. This is the most important part, and I will continue to focus on this part. Since Hasse diagram already knows the above three characteristics, it is a method of expressing the relation as a height except for the direction (i.e., unordered pair). To draw a Hasse diagram, in the reading group, we created an algorithm, but I thought it was wrong, so I fixed it from the beginning within the overall framework.

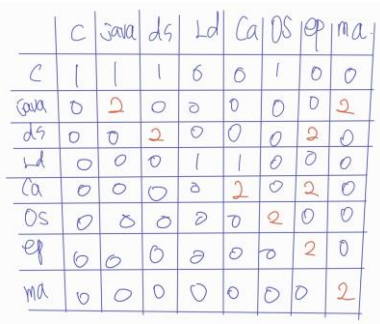
2. Approach

Since the input is a binary relation, we thought it could be expressed in 2d-array. So, if it has a relation, it is expressed as 1, and if it has no relation, it is expressed as 0. In addition, I thought that partial order relation is antisymmetric, so I could change it to an upper triangular matrix to reduce the loop time, but this had a problem. Because binary relation consists of ordered pairs, and if we think of it symmetrically, the order changes. This will be covered in more detail in the discussion part. First, the name is stored in the array while looping in input, and the matrix expressing the relation in that order is made into 2d-array. Then, the matrix is expressed as follows.

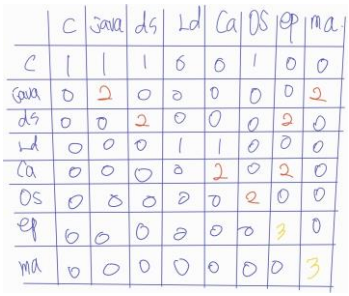


When filling in this matrix, if there is a relation to a non-diagonal, symmetrical portion, and there is a relation in input, then it is not antisymmetric and thus becomes an invalid input. Therefore, the program is terminated. Next, since the closure of the partial order relation of the input file must be considered, the reflexive, that is, the diagonal part, must be made 1, and the transitive part must be found and removed. I used recursions to remove the transitive part, and when a nonzero element is found in the row, transferred the index back to the argument, and took the first index back to the latter and found the first index back to the index. Here, we can find an invalid input, which has a circuit. Circuits cannot exist in partial orders because transitive properties break antisymmetric properties. In addition, if there is a circuit here, an infinite loop occurs in this recursive function. So, at this time, the global variable error was declared as a way to confirm that it was an invalid input, initializing the error to 0 every time the call a recursive function was first started, and +1 every time it was executed so that we could know how many times the function was executed. At this time, if the error exceeds 31*32/2, it means that an infinite loop is occurring and ends the program. Why this is so will also be discussed in more detail in the discussion part.

Except for all of these transitive elements, the height must be expressed to draw the Hasse diagram. First of all, if we know the height and the number of vertices per height in the matrix, we can draw a Hasse diagram. To find the height in the matrix, I came up with the idea of a topological sorting algorithm. First, find the minimal in the matrix. These are the elements on the first floor at the bottom, that is, in the Hasse diagram. As a way to find minimal, there should be no incoming edge except for the reflexive part (i.e., the diagonal part). This means that in the column, all elements are zero except the diagonal part. Then add +1 to the elements except 0 in all rows except for the row with this index. Then all the elements of the row in the vertex on the first floor are 1, and the rest are 2 like below.



At this time, this minimal is stored in a new array, excluding this minimal, and then the above process is repeated. Then, we can find other minerals, and it can be seen that they are on the second floor. If this process is repeated until the number of minimal and the total number of elements are the same, the number of vertices per height and the height of Hasse diagram can be finally known.

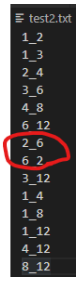


Now all we have left is to draw a Hasse diagram with this matrix. In order to draw the Hasse diagram, I thought of many ways, but when I drew it in a square grid, I thought that there would be a problem that could be overlapped by other lines without a relationship, so each layer was divided into the number of vertices and printed out. First, each height was circled by the number of vertices, and each time the names were written next to each other one by one. At this time, since these names are not in the order of height, a new array was created to arrange them in the order of height again.

After drawing all the vertex and names in this way, all we had to do was draw a line, which was solved in the order of each floor. In the loop in matrix, checked which order the starting vertex was in on that layer, and we checked which order the vertex was in on that layer, drawing a line from which to which on the starting vertex layer of the ending vertex. In this way, a Hasse diagram was made.

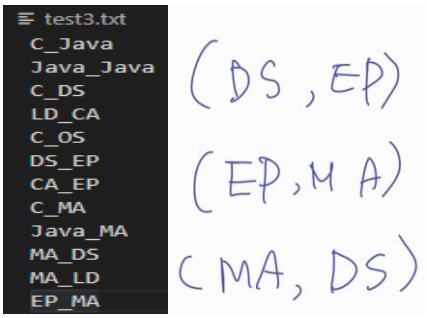
3. Evaluation

To evaluate whether the Hasse diagram is well drawn, several cases can be considered. First, if it is not a antisymmetric, this is when pairs that are not the same in binary relaxation exist symmetrically. In this case, the program must be terminated. We can see that it ends well.



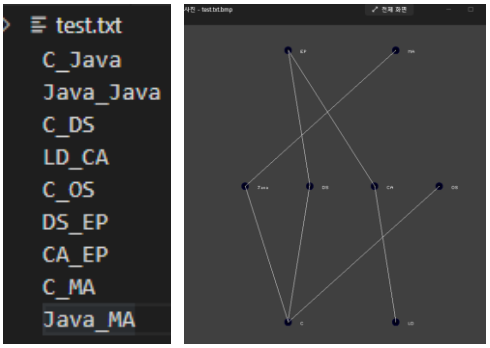

```
PS C:\Users\82108\Desktop\T,F 6 Discrete Mathematics\PA3> ./out test2.txt
Invalid input. This is not a partial order relation.
```

In addition, the program must be terminated in the same way when the circuit exists. This is also because it is not antisymmetric. If we intentionally create a circuit in a normal partial order relation, it can be seen that the program ends as follows.

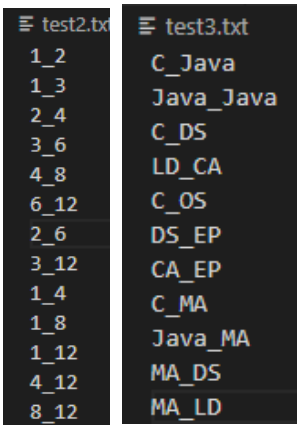


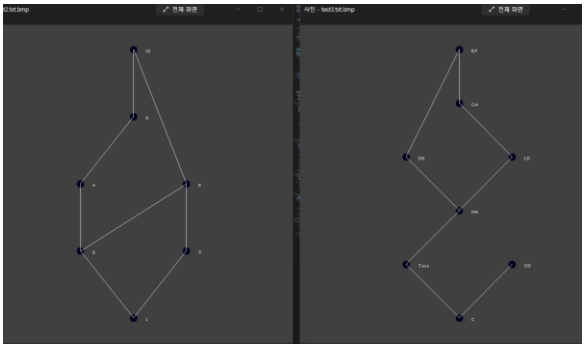

```
PS C:\Users\82108\Desktop\T,F 6 Discrete Mathematics\PA3> ./out test3.txt
Invalid input. This is not a partial order relation.
```

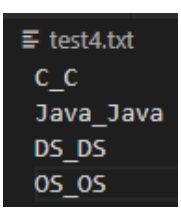
Next, in a normal partial order relation, check whether a Hasse diagram satisfying all binary relations is drawn, and it could be known that it is drawn normally as follows.

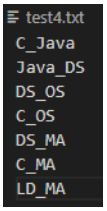


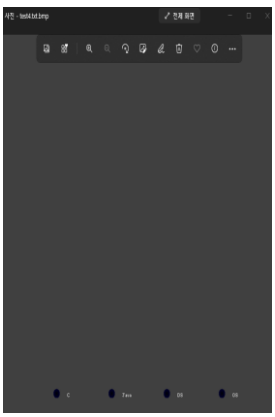
Slightly different examples were also tested.

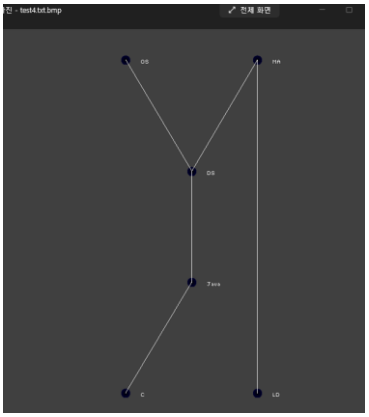












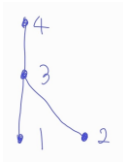
Therefore, it can be said that the above program was well made.

4. Discussion

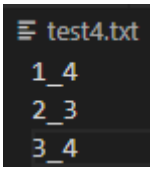
I mentioned several things to consider earlier. First, partial order relation can be expressed in an upper triangular matrix, and let's look at why it should not be expressed in this way. Since it is antisymmetric, if we make the right (index of the column) larger based on the index, it becomes an upper triangular matrix. However, if expressed in this way, the order is changed because it is ordered pair. So, suppose that there are pairs (1,2) (2,1), it is not a poset because it is not antisymmetric, but because all of them are expressed only (1,2) in the matrix, the program is mistaken for a poset. In addition, the order may change, resulting in a wrong picture.

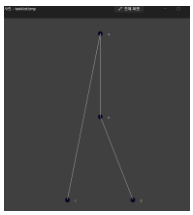
Another point to look at is why there should not be a circuit in the partial order relation. To find the invalid input, the program was terminated when there was a circuit. Suppose there is a circuit. If it is a pair symmetrical to each other (e.g., (1,2) (2,1)), it is clearly not a partial order relaxation because it is not antisymmetric. In addition, suppose there is a circuit of three or more pairs. For example, if there is a circuit that is consisted of {(1,2), (2,3), (3,1)}, the partial order relation is transitive, so (1,3) also belongs to the relation. Since it belongs to both (1,3) and (3,1) relations, it is not antisymmetric and is not a partial order relation. In this process, if there is a circuit, an infinite loop occurs in the program, and the program was terminated when the variable error exceeded 32*31/2. To explain this in more detail, if the recursion can occur to the maximum in the matrix, it would be the case where there is 1 in all the upper triangular matrix even if it is not an upper triangular matrix. Of course, this is also likely to decrease a little after transitive is excluded. However, without considering this, if we think about it as much as possible, the maximum number of elements in the first row is 31, followed by 30… In the last row, 0 total 31*32/2 recursions may occur. To expand this a little bit, if we know the number of elements, will not be able to call a function more than the number added from 1 to the number of elements. Therefore, within the program, the program was implemented to end when the error exceeds element_num * (element_num +1) / 2.

Finally, I would like to give an example that said that drawing in a square grid should not overlap. If there is a binary relation of {(1,4),(2,3),(3,4)}, the following figure is drawn, which is a Hasse diagram representing the relation of {(1,3), (2,3), (3,4)}. The above method was not chosen because there may be many cases of wrongdoing.



If so, considering whether there is an overlapping situation in the method of dividing and distributing each height by the number of vertices, of course, there is a possibility of overlapping when the height elements are the same. However, if we test the above example, you can see that it comes out as follows, and like this example, it is more reasonable because it allows us to avoid overlapping in many cases.





5. Conclusion

Through the process of drawing Hasse diagram from a given binary relation, I learned the characteristics of partial order relation and binary relation, and how they are considered in the program. It was amazing that theoretical characteristics appeared accurately within the program, and it was also amazing that theoretical things could be learned again from the implementation of the program. Other assignments were really fun, but this assignment was also really interesting and has a lot to learn.