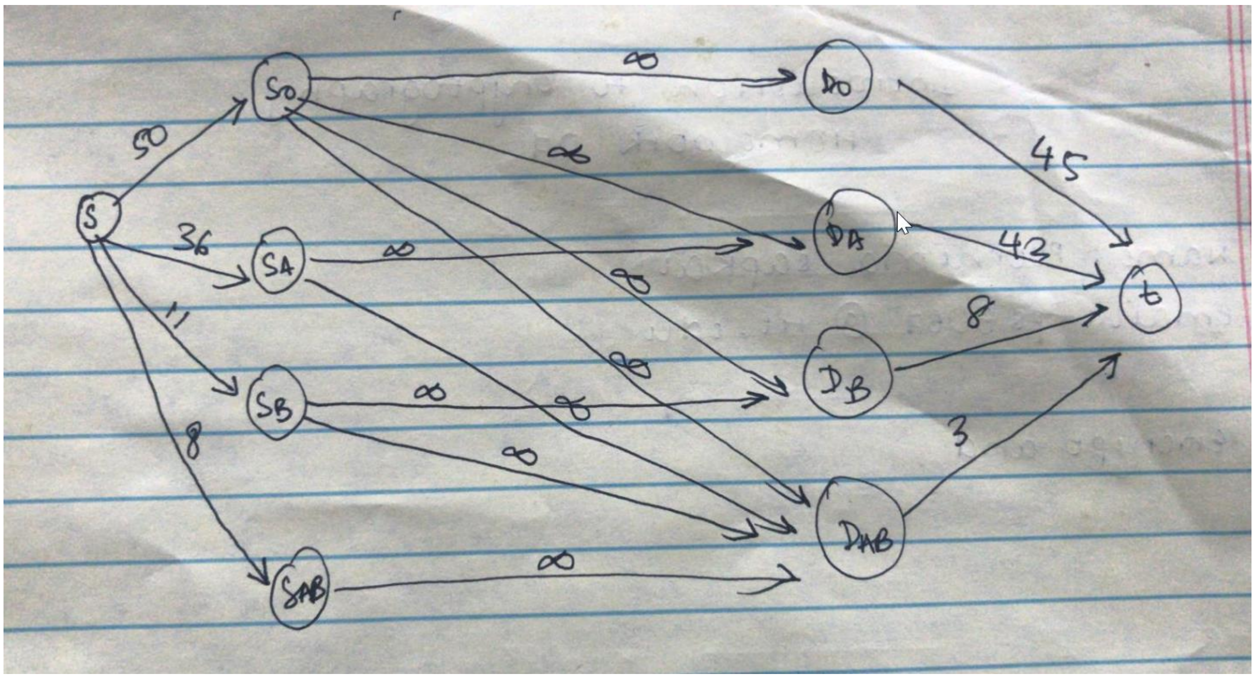
Summary:

We create two nodes s and t and connect them to SO(representing blood type 0), SA(representing blood type A), SB(representing blood type B) and SAB(representing blood type AB). Here the vertex s which is created represents the supply and the vertex t created represents the demand. Individual demands DO,DA,DB,DAB represents the individual demands for type O,A,B,AB which each individually connect to the sink t. We here have to clearly note which supply can be connected to which demand. So can be connected to DO,DA,DB,DAB as it's a universal doner.SA can be connected to DA and DAB. SB can be connected to DB and DAB and SAB can be only connected to DAB because of the demand. Here these connections are made to infinity flow but in ou code are taken as Integers max value.



Algorithm:

A) Algorithm

Procedure : FindingMaxFlow(G)

Input : Graph G(V,E)

Output : Max Flow

Step 1: Read the input which is the graph as mentioned in the question

Step 2: Perform Breadth first search on the graph to be able to find augmenting paths for each

Step 3: For every augmenting path that you get find identify the max flow of that path

Step 4: Subtract the flow of every edge in that path with the max flow that you found.

Step 5: Keep repeating this until there is no path left to traverse from the source to the sink.

B)

Input table from textbook of Supply/Demand

blood type	supply	demand
O	50	45
A	36	42
B	11	8
AB	8	3

It is not possible to meet the full demand in this graph. The total demand that we have it 98 which can be calculated using = (45+42+8+3)=98. We have achieved the max flow = 97 so we know that the value of the max flow is less than the demand and also equal to the min cut which we achieved to be 97 =(50+36+8+3). This proves that the blood at hand can not suffice the projected need.

B ii) Find an allocation that satisfies the maximum possible number of patients. Use an argument based on a minimum-capacity cut to show why not all patients can receive blood. Also, provide an explanation for this fact that would be understandable to the clinic administrators, who have not taken a course on algorithms. (So, for example, this explanation should not involve the words flow, cut, or graph in the sense we use them in this book.)

Since we have proved that the full demand can not be met, the resulting flow will tell us how to meet this demand in clinical terms :

- We use 45 units of blood type O to transfuse with other O-type blood
- We use 5 units of blood of type O to transfuse with Type A blood
- We use 36 units of blood type A to transfuse with Type A blood
- We use 8 units of blood type B to transfuse with Type B blood
- We use 3 units of blood type AB to transfuse with Type AB blood

45+5+36+8+3 =97 which is equal to our max flow.

Complexity: O(maxflow*(E+V))

How to execute the code:

Javac BloodGroup.java

Java BloodGroup

Result

Graph After every iteration

```
*****************************
   0   50   36   11    8    0    0    0    0    0
   0    0    0    0    0 2147483647 2147483647 2147483647 2147483647    0
   0    0    0    0    0    0 2147483647    0 2147483647    0
   0    0    0    0    0    0    0 2147483647 2147483647    0
   0    0    0    0    0    0    0    0 2147483647    0
   0    0    0    0    0    0    0    0    0   45
   0    0    0    0    0    0    0    0    0   42
   0    0    0    0    0    0    0    0    0    8
   0    0    0    0    0    0    0    0    0    3
   0    0    0    0    0    0    0    0    0    0
Graph After every iteration
*****************************
   0   50   36   11    8    0    0    0    0    0
   0    0    0    0    0 2147483647 2147483647 2147483647 2147483647    0
   0    0    0    0    0    0 2147483647    0 2147483647    0
   0    0    0    0    0    0    0 2147483647 2147483647    0
   0    0    0    0    0    0    0    0 2147483647    0
   0    0    0    0    0    0    0    0    0   45
   0    0    0    0    0    0    0    0    0   42
   0    0    0    0    0    0    0    0    0    8
   0    0    0    0    0    0    0    0    0    3
   0    0    0    0    0    0    0    0    0    0
Graph After every iteration
*****************************
   0   50   36   11    8    0    0    0    0    0
   0    0    0    0    0 2147483647 2147483647 2147483647 2147483647    0
   0    0    0    0    0    0 2147483647    0 2147483647    0
   0    0    0    0    0    0    0 2147483647 2147483647    0
```

0 0 0 0 0 0 0 0 2147483647 0

0 0 0 0 0 0 0 0 0 45

0 0 0 0 0 0 0 0 0 42

0 0 0 0 0 0 0 0 0 8

0 0 0 0 0 0 0 0 0 3

0 0 0 0 0 0 0 0 0 0

Graph After every iteration

****

0 50 36 11 8 0 0 0 0 0

0 0 0 0 0 2147483647 2147483647 2147483647 2147483647 0

0 0 0 0 0 0 2147483647 0 2147483647 0

0 0 0 0 0 0 0 2147483647 2147483647 0

0 0 0 0 0 0 0 0 2147483647 0

0 0 0 0 0 0 0 0 0 45

0 0 0 0 0 0 0 0 0 42

0 0 0 0 0 0 0 0 0 8

0 0 0 0 0 0 0 0 0 3

0 0 0 0 0 0 0 0 0 0

Graph After every iteration

****

0 50 36 11 8 0 0 0 0 0

0 0 0 0 0 2147483647 2147483647 2147483647 2147483647 0

0 0 0 0 0 0 2147483647 0 2147483647 0

0 0 0 0 0 0 0 2147483647 2147483647 0

0 0 0 0 0 0 0 0 2147483647 0

0 0 0 0 0 0 0 0 0 45

0 0 0 0 0 0 0 0 0 42

0 0 0 0 0 0 0 0 0 8

0 0 0 0 0 0 0 0 0 3

```
   0    0    0    0    0    0    0    0    0    0
The Min Cut is along the below mentioned edges
0 - 1
0 - 2
7 - 9
8 - 9
The max flow is
97

Process finished with exit code 0
```