Jesse Yang (23307563), Qichong Huang (23311085)

Shallows Explanation

Explanation

The Shallows task can be briefly concluded as ***finding the shortest path***. Though there are few differences between Shallows and normal shortest path problem, the basic idea of solution is almost the same.

Firstly, find the ***starting point***, in this case, will be the origin port. Step1
Then, we ***add lines*** which depart port is the origin port. Step2
Next, we add more lines which connect the lines added in step2. Step3
At the end, we have create ***all possible routines*** that starts with origin port and end with every port. Find out the "shortest path" Step4

While connecting ports, update the information in results array.

```
int i;
int [] results = new int[ports];    //initlialize the results array
for(i=0;i<ports;i++){
  results[i]=0;
}
int j;
int temp = 0;
```

At the beginning of our solution, we create an array results[ ] to store the result. And it will be initialized with 0.

```
for(i=0;i<lanes.length;i++){
  if(lanes[i].depart==origin){
    results[lanes[i].arrive]=lanes[i].depth;
    count++;
  }
}
```

Then, we use a for loop to find every line which depart with origin port. And we store the depth in the corresponding position of results[ ]. And count the number of lines we find in this step.

```java
    int k = 0;
  while(k<ports-count){
    for(i=0;i<ports;i++){
      if(i == origin){
        results[i] = Integer.MAX_VALUE;
        continue;
      }
      if(results[i]==0)
      continue;
      for(j=0;j<lanes.length;j++){
        if(lanes[j].depart==i){
          if(results[i]<lanes[j].depth && results[i]!=0)
          temp = results[i];
          else
          temp = lanes[j].depth;
          if(temp>results[lanes[j].arrive])
          results[lanes[j].arrive]=temp;
        }
      }
    }
    k++;
  }
```

For the example given,

Consider maximumDraughts(5, lanes, 0) with lanes of the form {depart, arrive, depth} as follow:
{0, 1, 9} : Lane from port 0 to port 1 with shallow point 9 units deep
{0, 2, 2} : Lane from port 0 to port 2 with shallow point 2 units deep
{0, 3, 1} : Lane from port 0 to port 3 with shallow point 1 units deep
{1, 2, 7} : Lane from port 1 to port 2 with shallow point 7 units deep
{1, 3, 2} : Lane from port 1 to port 3 with shallow point 2 units deep
{2, 3, 8} : Lane from port 2 to port 3 with shallow point 8 units deep
{4, 2, 9} : Lane from port 4 to port 2 with shallow point 9 units deep

Now we have results[ ] which contain {0,9,2,1,0}

Meaning that:

0-0: 0

0-1: 9

0-2: 2

0-3: 1

0-4: 0

And we define 0-0 as interger.maxsize

Then, we ignore all ***non-existed routines*** (for now) , in this case we use
0-1：9
0-2：2
0-3：1

Next, for port =1, we find every line that connect to port1:
{1, 2, 7} : Lane from port 1 to port 2 with shallow point 7 units deep
{1, 3, 2} : Lane from port 1 to port 3 with shallow point 2 units deep

We now have
0-1-2 = 0-2：7
0-1-3 = 0-3：2

The results[ ] will be updated according to this:
0-1：9
0-2：7
0-3：2

Also, we repeat this process for port = 2, 3.
That's what ***nested for loops*** do in the code above.

If there is a routine which make port 0 connect to a new port (port = 4), we also update that in result array.

And we need a ***big while loop*** to include all of this, which will circle for the number of "non-existed routines". (Considering ***the worst case***) Cause after each time we finish the nested loops, the origin ports might now be able to ***reach to a new port***, meaning that those "non-existed routines" perhaps are now existed.

At the end we return the results[ ].

Time complexity

O(P^2 L)