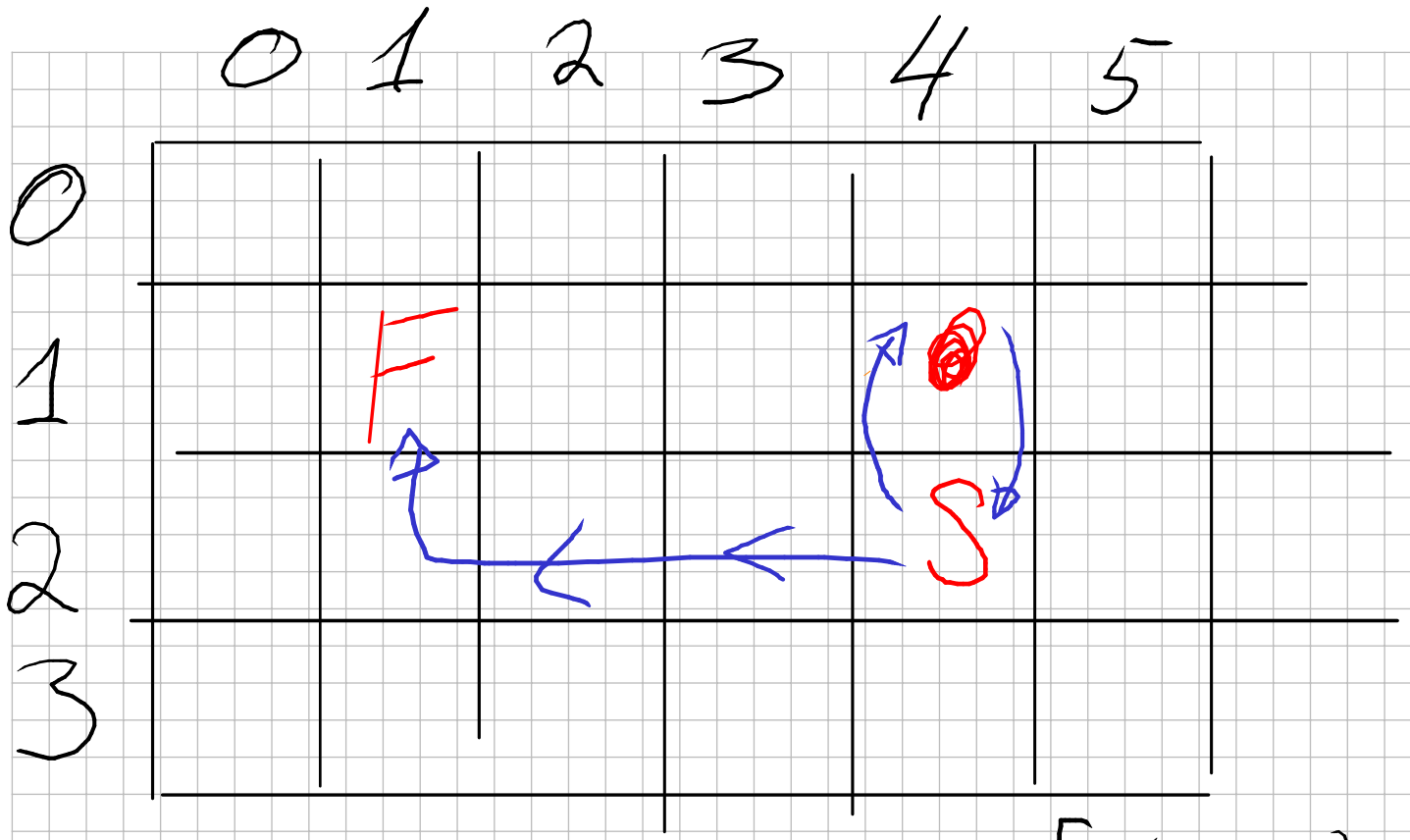


S = (4, 2)
F = (1, 1)

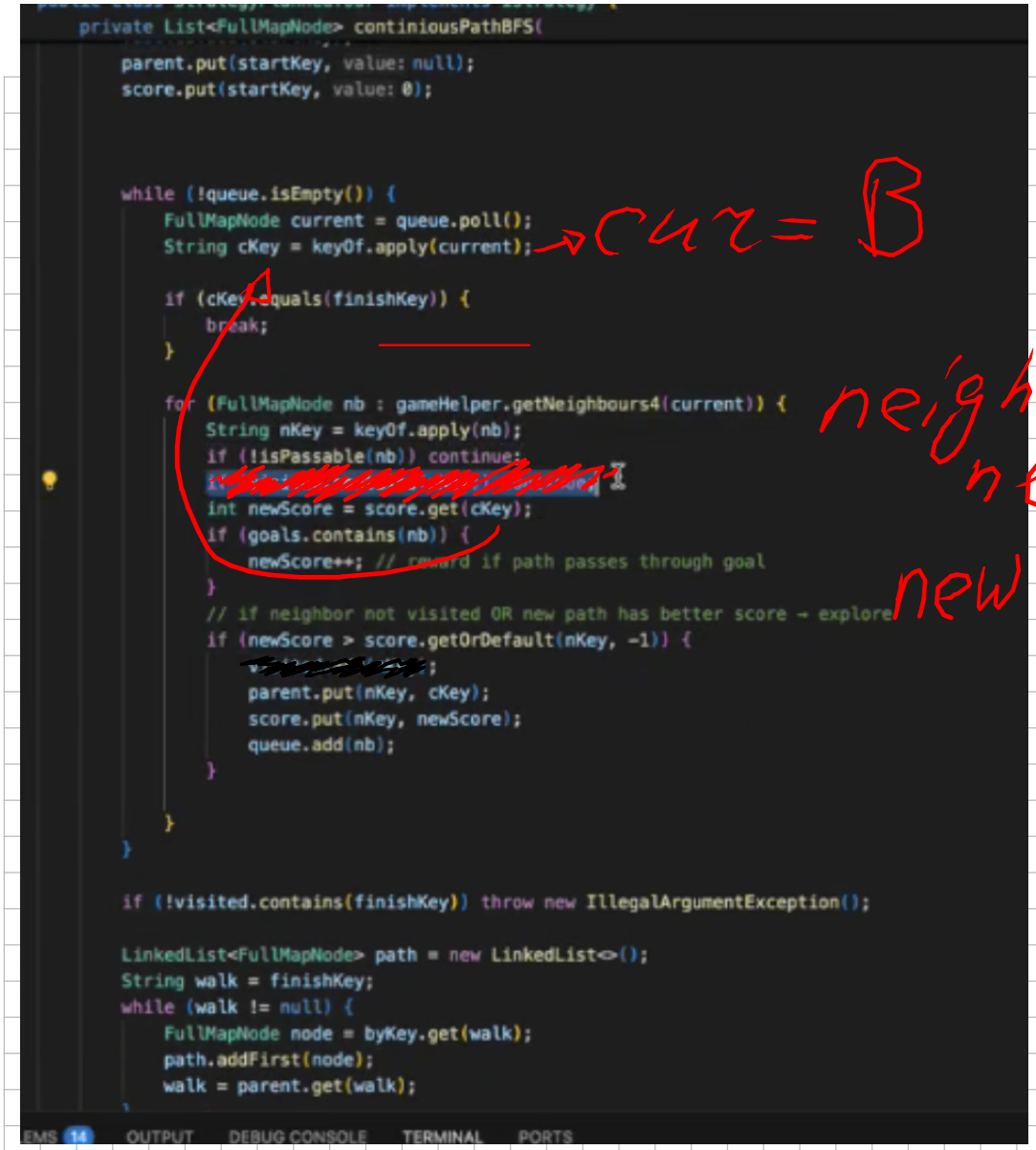
Visited

cur = (4, 1)

queue = [(5,2) (4,3) (3,2)]

(5,1) (3,1) (4,0) (4,2)

(4,2) → (4,1) → (4,2)



cur = B

neighbours = [D A]

nb = D

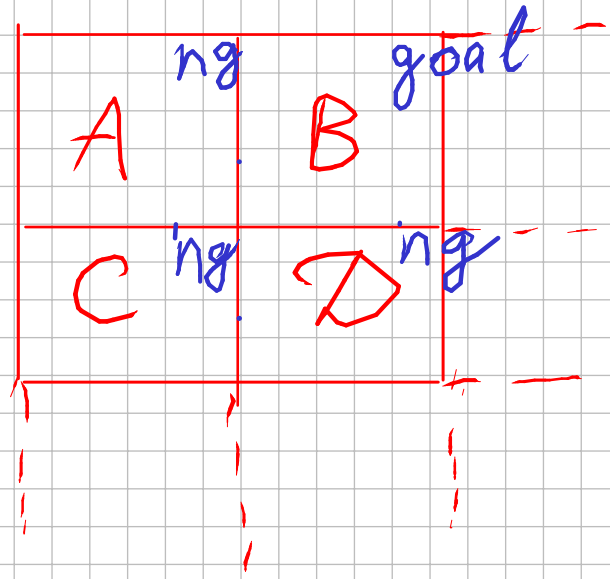
newScore = 1 + 0 = 1

Start = D Finish = A

```
// if neighbor not visited OR new path has better score → explore
if (                                    newScore > score.getOrDefault(nKey, -1)) {
    visited.add(nKey);
    parent.put(nKey, cKey);
    score.put(nKey, newScore);
    queue.add(nb);
}
```

1) 2)

P = [D:null B:D
      C:D   A:C]



Q = [A]

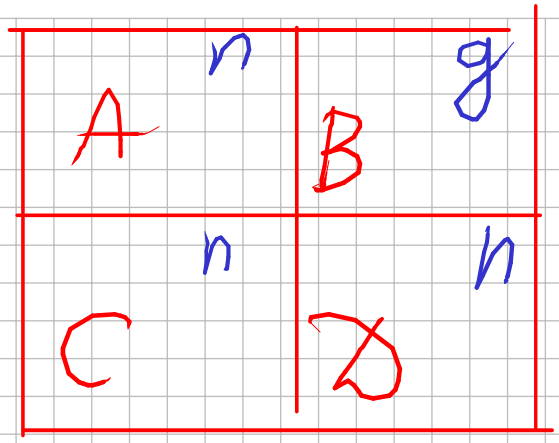
V = [D, C, B, A]

Score = [(D,0) (C,0) (B,1) (A,0)]

cur = B nb = D

Start = D Finish = A

```
228     String startKey = keyOf.apply(start);
229     String finishKey = keyOf.apply(finish);
230     queue.add(start);
231     visited.add(startKey);
232     parent.put(startKey, value: null);
233     score.put(startKey, value: 0);
234
235
236
237     while (!queue.isEmpty()) {
238         FullMapNode current = queue.poll();
239         String cKey = keyOf.apply(current);
240
241         if (cKey.equals(finishKey)) {
242             break;
243         }
244
245         for (FullMapNode nb : gameHelper.getNeighbours4(current)) {
246             String nKey = keyOf.apply(nb);
247             if (!isPassable(nb)) continue;
248             if (visited.contains(nKey)) continue;
249             int newScore = score.get(cKey);
250             if (goals.contains(nb)) {
251                 newScore++; // reward if path passes through goal
252             }
253             // if neighbor not visited OR new path has better score → explore
254             if (newScore > score.getOrDefault(nKey, -1)) {
255                 visited.add(nKey);
256                 parent.put(nKey, cKey);
257                 score.put(nKey, newScore);
258                 queue.add(nb);
259             }
260
261         }
```



nbs = [ ]

Q = [ ]

V = [D, C, B, A]

[Score = D:0
C:0 B:1 A:0]

P = [D:null, C:D, B:D
A:C]

[D → C → A]