Selected solutions Module 8

Exercise 15.7.

(a) First note that you will never walk through a corridor twice in the same direction. So, in finite time, you will either find an exit or get stuck.

Suppose you get stuck. Then you must be back at your starting point, because for each other room you can choose a different exit each time you enter that room.

Number the rooms in order they are first visited. We prove by induction on i that all corridors connected to room i have already been traversed in both directions.

For $i = 1$ (the starting room) this is certainly true because we are stuck here, meaning that we have exited the room through each corridor, and so we have entered it through each corridor.

Now assume that we have traversed all corridors of room i in both directions. Then we have also traversed the corridor to room $i + 1$ in both directions. Hence, we have exited room $i + 1$ through an exit marked with $\circ$. Hence we have exited room $i+1$ through each corridor. Hence, we have entered room $i + 1$ through each corridor.

It now follows by induction that all corridors connected to room i have been traversed in both directions, for all i. Hence, all corridors that are reachable from the starting room have been traversed in both directions. So if there were a route to an exit, we would have found it.

(b) You traverse each corridor at most twice. Each time you traverse a corridor, you enter a room and need to find an exit without $\times$. The number of exits of each room is at most equal to the number of rooms (minus one). Hence, the running time is $O(r \cdot c)$, with r the number of rooms and c the number of corridors.

Exercise 15.8

A possible algorithm (Depth First Search) is as follows:

- Put s in a list Q.
- Repeat the following steps until Q is empty:
  - consider the last vertex v in list Q;
  - if $v = t$, output YES and stop;
  - remove v from Q;
  - if v is not marked as visited:
    * add each neighbour of v to list Q (at the end);
    * mark v as visited.
- output NO.

Correctness of the algorithm can be shown similarly as in the previous exercise. The running time is $O(|E|)$ because each edge is considered at most once, each vertex reachable from s is considered at most once and there are at most $|E| + 1$ such vertices.