Backtracking

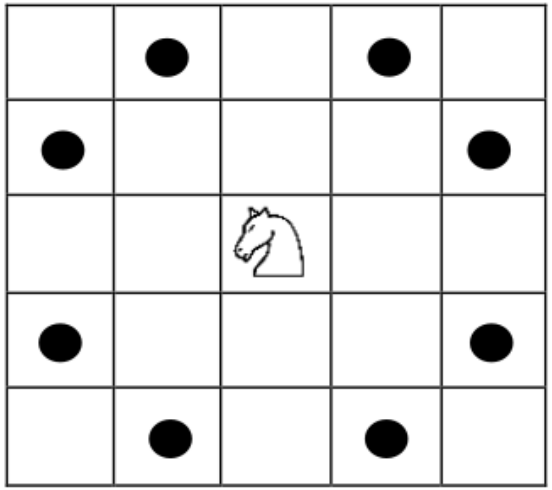
Simeon Monov

Backtracking

- There are problems that can be solved only with full solution search (NP-Complete and NP-Hard problems).
- One way to solve such problems is by using backtracking
- Backtracking is a method where the solution is constructed in sequence (in steps).
- Each step is constructed by extending the current solution with all other possible solutions and abandons a candidate ("backtracks") as soon as it determines that the candidate cannot possibly be completed to a valid solution.

Knight tour problem

- Problem definition:
  - Given a N*N board with the Knight placed on the first block of an empty board. Moving according to the rules of chess knight must visit each square exactly once. Print the order of each cell in which they are visited.
- Straight forward solution with backtracking



Knight tour backtracking

- The algorithm:
  1) The board is defined as two-dimensional (matrix) array NxN – **board[,]**
  2) Knight start position is defined as **startX, startY**
  3) We define two helper arrays offesetX and offsetY, defining the offsets of different movements of the knight:

```
private int[] offsetX = new int[] { -2, -2,  1, -1,  2,  2,  1, -1 };
private int[] offsetY = new int[] {  1, -1,  2,  2,  1, -1, -2, -2 };
```

  4) Create two-dimensional array marking visited positions **visited[,]**
  5) Initialize current as 1
  6) Execute recursion (on next slide)

Knight tour backtracking

6) Recursion starts from **x=startX**, **y=startY** and **current=1**:
   1) If **current==N*N**, means we found a solution (made N*N steps) – return true.
   2) Mark current position as visited: **visited[x, y] = current** (keep track of visiting sequence).
   3) Try to visit (recursively executed same method) all next positions (solutions). We can visit another position (newX, newY) only if it is inside the board borders (newX and newY are between 0 – (N-1)) and was not visited.
   4) If visiting next solution returns true, return true in order to exit the recursion.
   5) Mark current positions as not visited: **visited[x, y] = 0**.
   6) Return false.