CS207 Design and Analysis of Algorithms

Sajith Gopalan

Indian Institute of Technology Guwahati sajith@iitg.ac.in

March 22, 2022

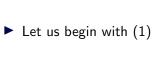
Backtracking

- ► Useful for constraint satisfaction problems (CSPs)
- ► (A CSP has a set of variables, each with a domain of values, and a set of constaints on these variables specified through relations on them. We need to find values for all variables, so that all constraints are satisfied. E.g., Sudoku)
- Incrementally builds candidates to the solutions
- Abandons a candidate ("backtracks") as soon as it becomes clear that the candidate cannot be completed into a valid solution

- ▶ Given to you are an $n \times n$ chess board, and n queens
- You are required to place the quuens on the board so that no two attack each other
- ▶ The board will have no other piece save these *n*-queens
- ► That means no two queens are on the same row, the same column or the same diagonal
- ► In particular, every row has exactly one queen
- ▶ Let Q_i be the queen of the i-th row

	Q_1		
			Q_2
Q_3			
		Q_4	

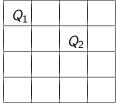
- \blacktriangleright We denote this solution by the tuple (2,4,1,3)
- ▶ Means: Q_1 is in column 2, Q_2 is in column 4 etc.
- ightharpoonup (2) is a partial candidate solution; only Q_1 has been placed
- ightharpoonup (2,4) is a partial candidate solution; only Q_1 and Q_2 have been placed
- ▶ (2,4,1) is a partial candidate solution; only Q_1 , Q_2 and Q_3 have been placed
- \triangleright (2,4,1,3) is a full solution
- ► Say, we need to find all the solutions





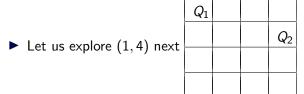
 \blacktriangleright (1,1) and (1,2) are invalid

ightharpoonup Let us consider (1,3)



▶ (1,3,i) is invalid for every i; so, (1,3) cannot be completed into a full solution

ightharpoonup We backtrack from (1,3) to (1)



 \blacktriangleright (1,4,1) is invalid; let us consider (1,4,2)

Q_1		
		Q_2
	Q_3	

(1,4,2,i) is invalid for every i; so, (1,4,2) cannot be completed into a full solution; backtrack to (1,4)

- ightharpoonup (1,4,3) and (1,4,4) are invalid; (1,4) has failed
- \blacktriangleright We backtrack from (1,4) to (1)
- ▶ All extensions of (1) have been explored; we backtrack to ()
- ► Let us explore (2) next
- \blacktriangleright (2,1), (2,2), (2,3) are invalid
- ► Let us consider (2,4)
- \blacktriangleright First, let us look at (2,4,1)
- \triangleright (2,4,1,1) and (2,4,1,2) are invalid

 \blacktriangleright (2,4,1,3) is a solution

	Q_1		
			Q_2
Q_3			
		Q_4	

- \blacktriangleright (2, 4, 1, 4) is invalid
- \blacktriangleright We backtrack from (2,4,1) to (2,4)
- \triangleright (2,4,2), (2,4,3), (2,4,4) are invalid
- ▶ We backtrack from (2,4) to (2), and then to ()
- And so on continues the backtracking algorithm
- ▶ By left-right symmetry we know that (3) and (4) are the mirror images of (2) and (1) respectively
- ightharpoonup So (3,1,4,2) is a solution as well
- ► There are only two solutions

	Q_1		
			Q_2
Q_3			
		Q_4	

		Q_1	
Q_2			
			Q_3
	Q_4		