ACM Lab Week 4

Complete Search

October 8, 2015

Complete Search

Complete Search means solving a problem by generating most/all candidate solutions in order to find one that solves the problem.

- Example: checking if a number n is prime by trying all possible divisors up to $\sqrt{n}$.

Complete Search

Complete Search means solving a problem by generating most/all candidate solutions in order to find one that solves the problem.

- Example: checking if a number n is prime by trying all possible divisors up to $\sqrt{n}$.

Backtracking

Backtracking is an important technique for performing a complete search.

Idea: recursively enumerate a set of partial candidates which eventually includes all solution candidates.

Backtracking

Pseudo-code:

```
1  backtrack(candidate) {
2    if reject(candidate) {
3      return;
4    }
5    if is_solution(candidate) {
6      output(candidate);
7    }
8    for (c ∈ extensions(candidate)) {
9      backtrack(c);
10   }
11 }
```

Backtracking

There are three important design choices:

1. how do I decide whether I should reject a candidate?
2. how do I know whether I have found a solution?
3. what are the extensions of a candidate?

For the performance it is crucial to reject candidates as early as possible.

Backtracking

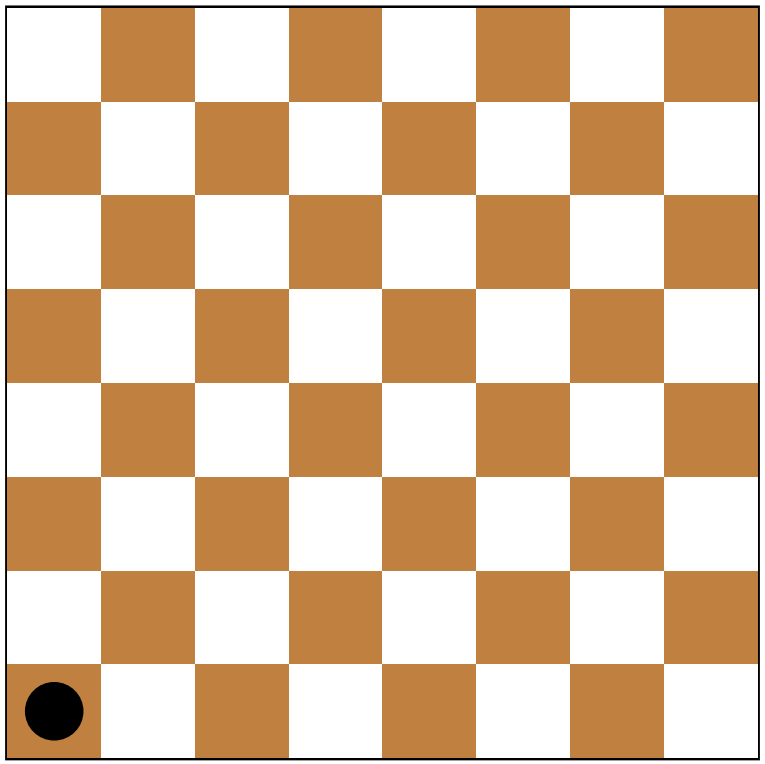
There are three important design choices:

1. how do I decide whether I should reject a candidate?
2. how do I know whether I have found a solution?
3. what are the extensions of a candidate?

For the performance it is crucial to reject candidates as early as possible.

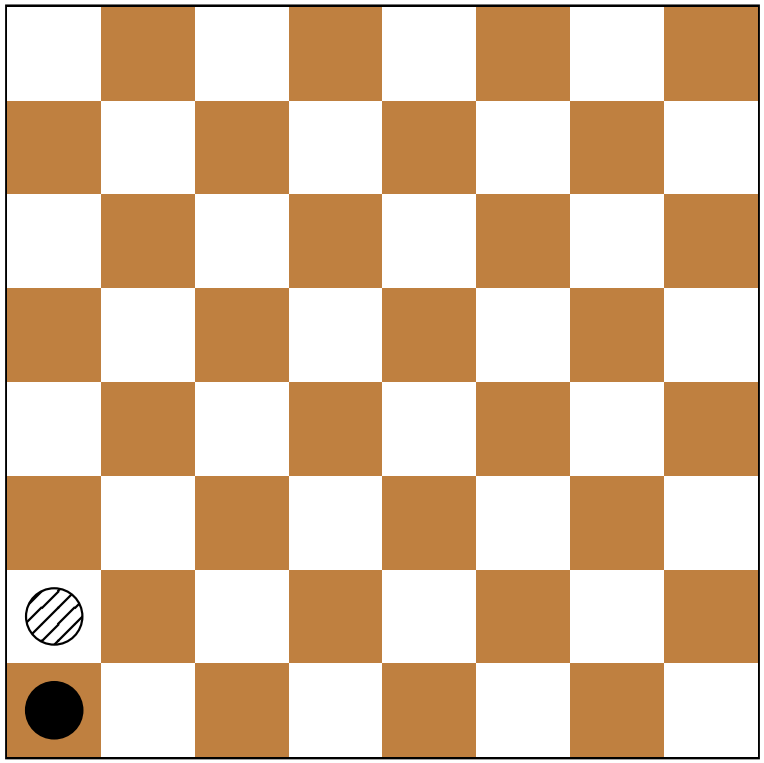
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



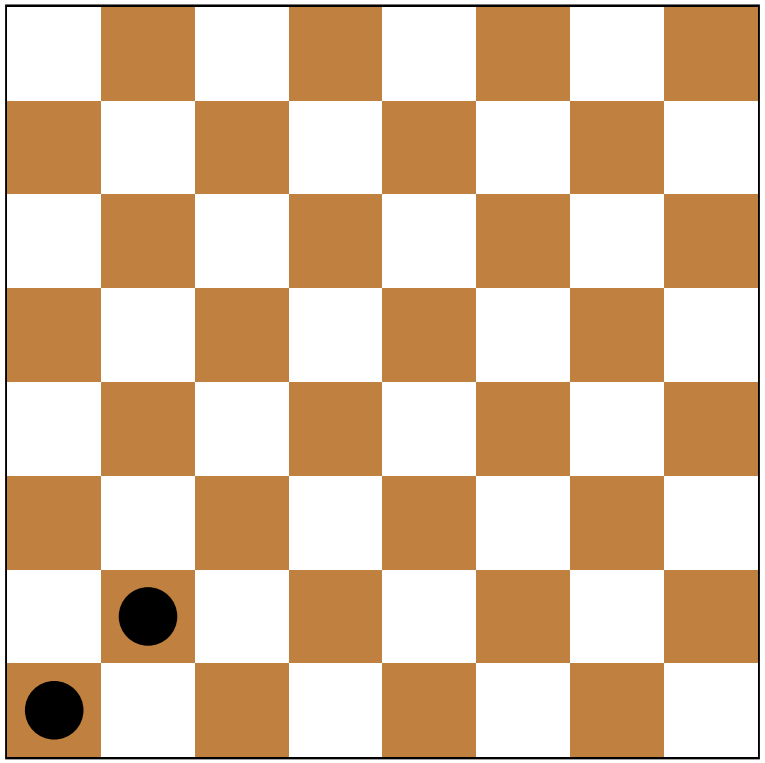
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



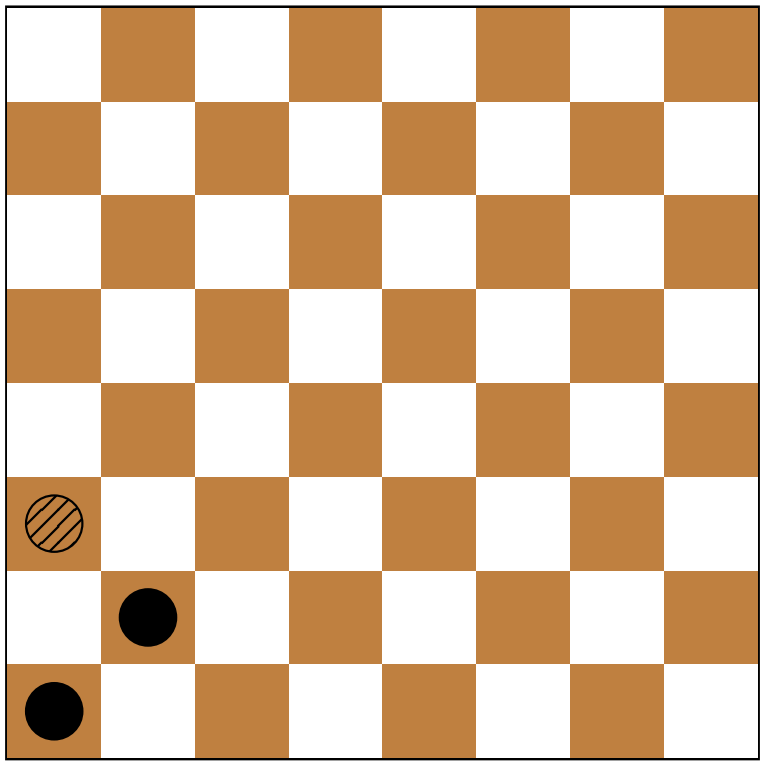
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



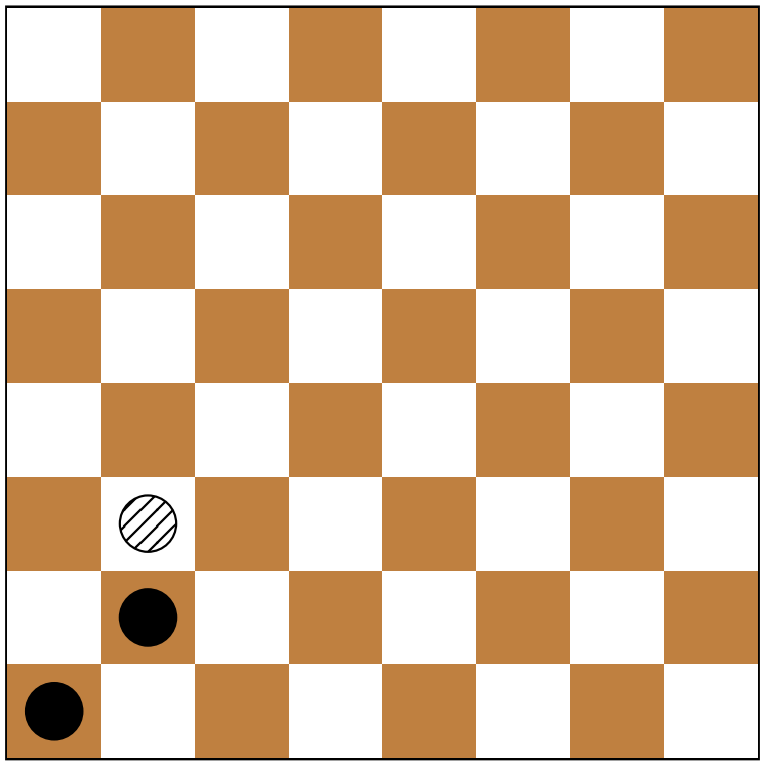
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



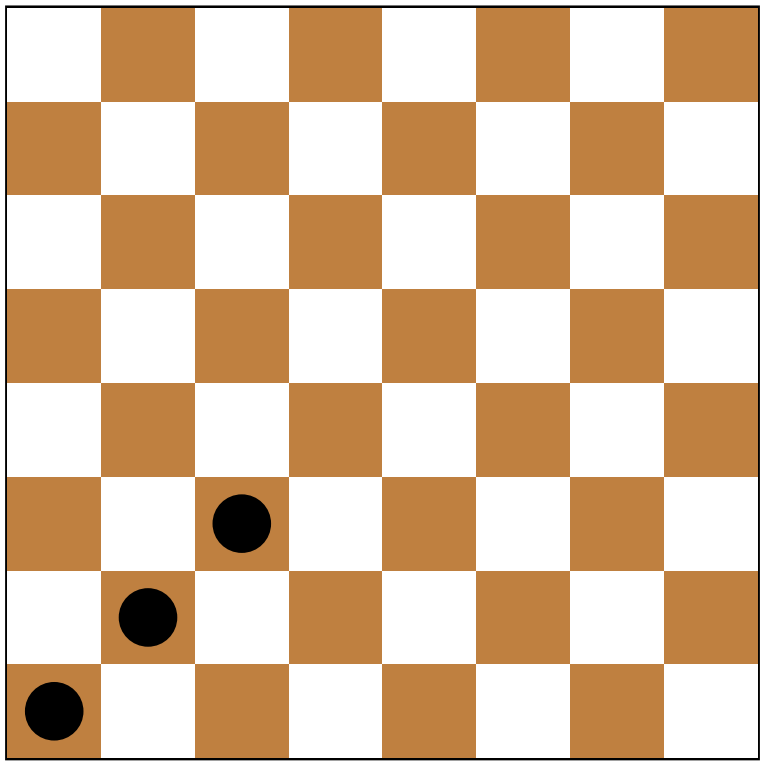
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



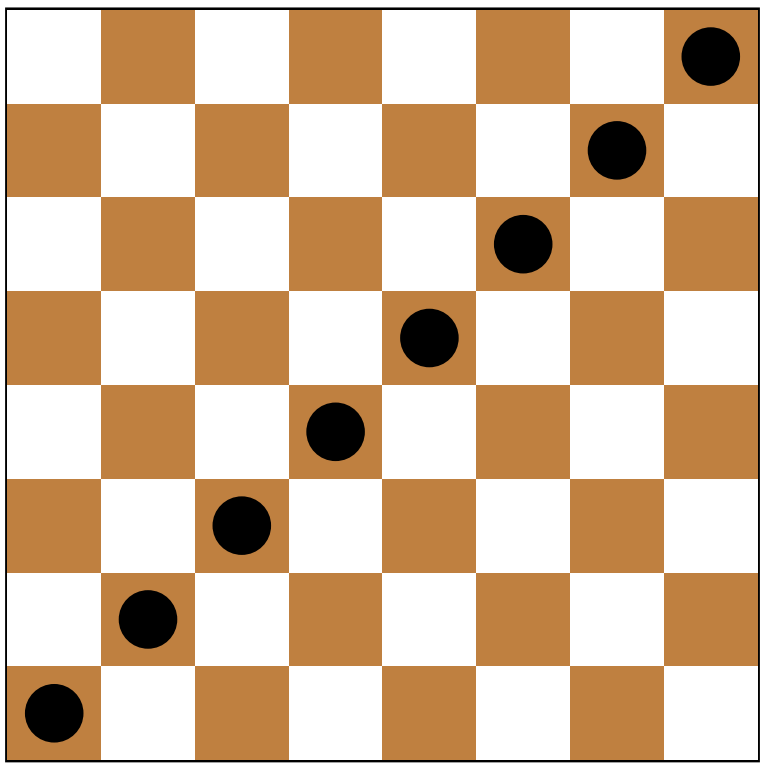
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



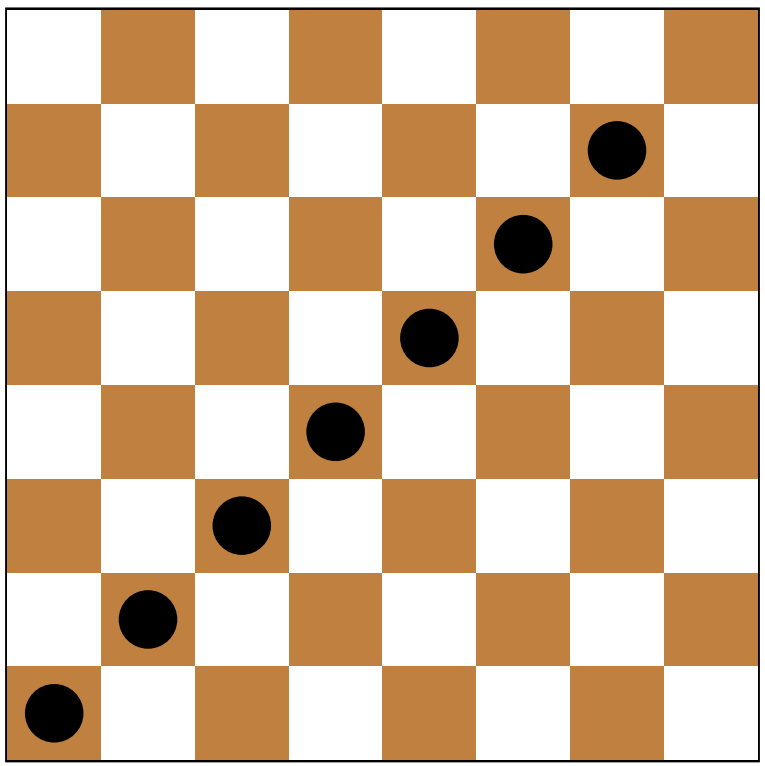
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



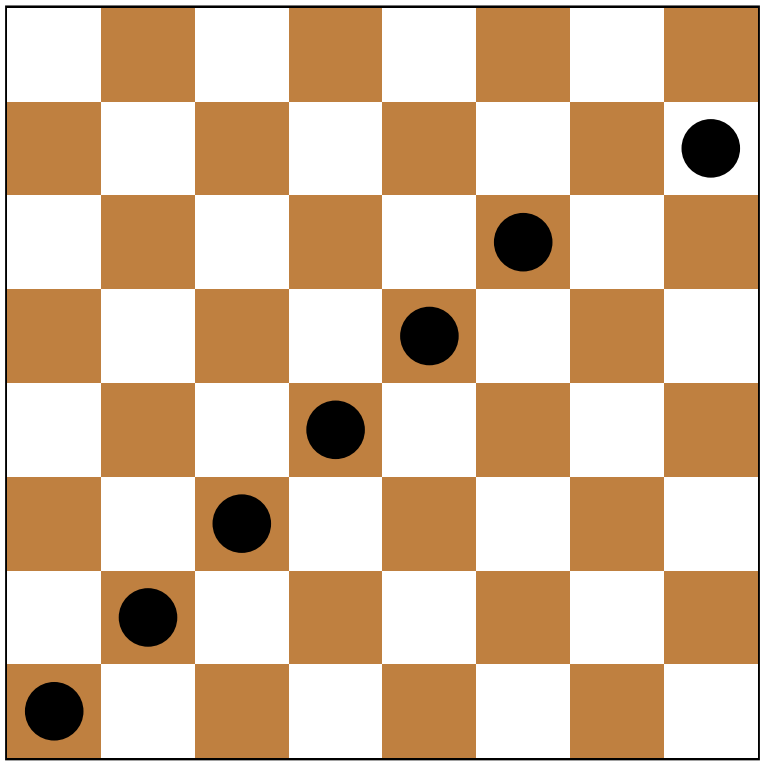
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



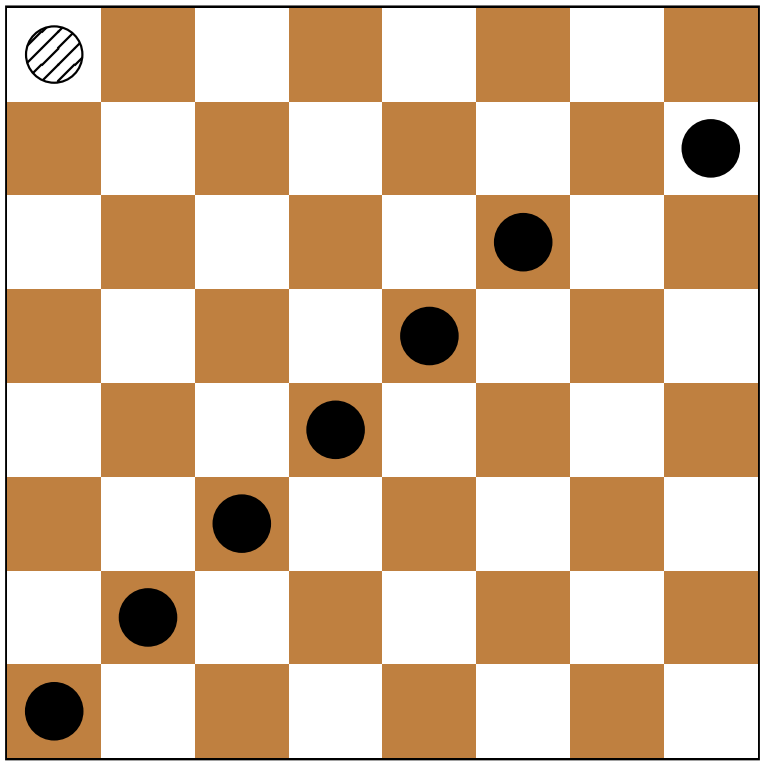
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



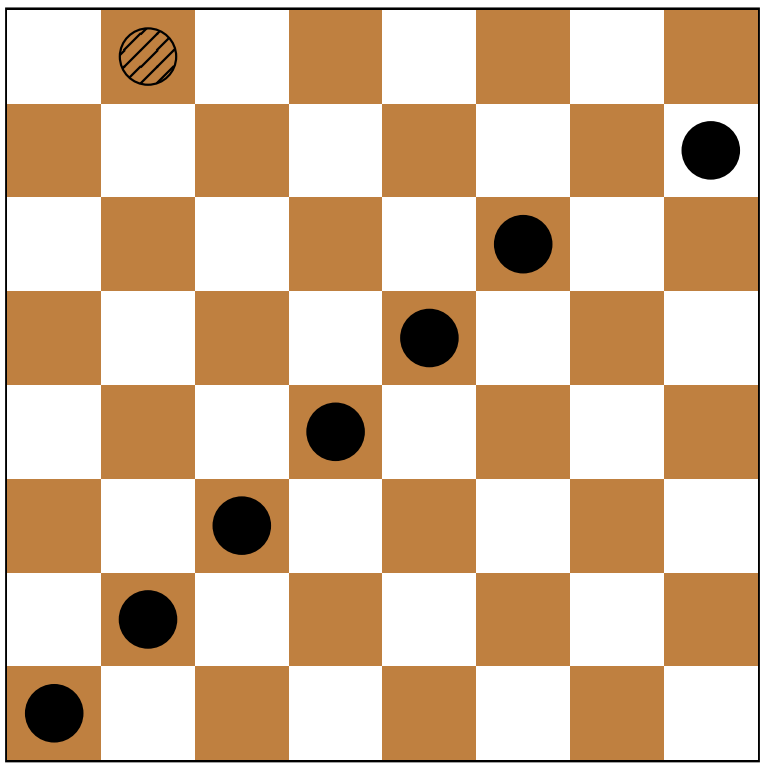
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



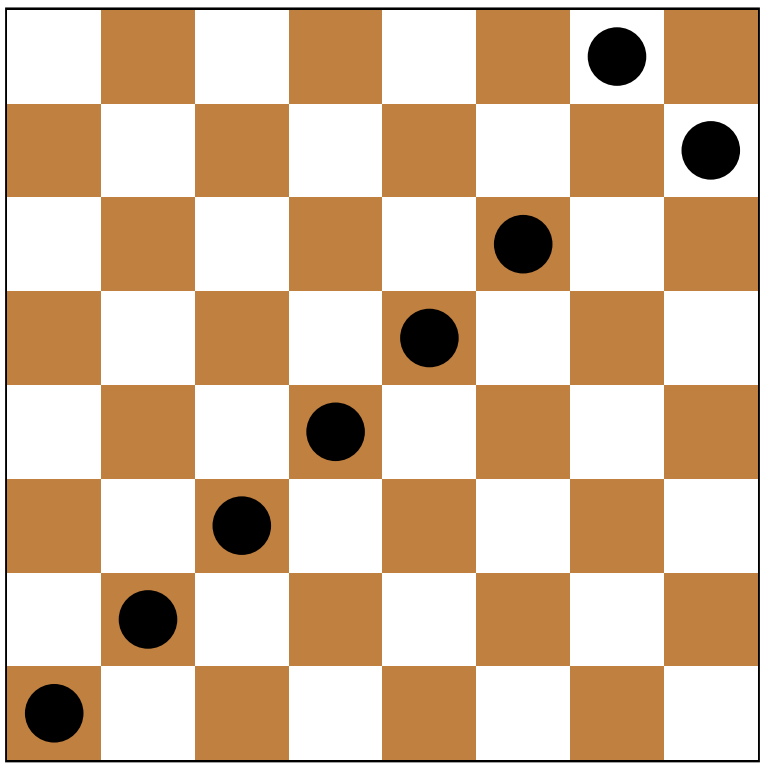
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



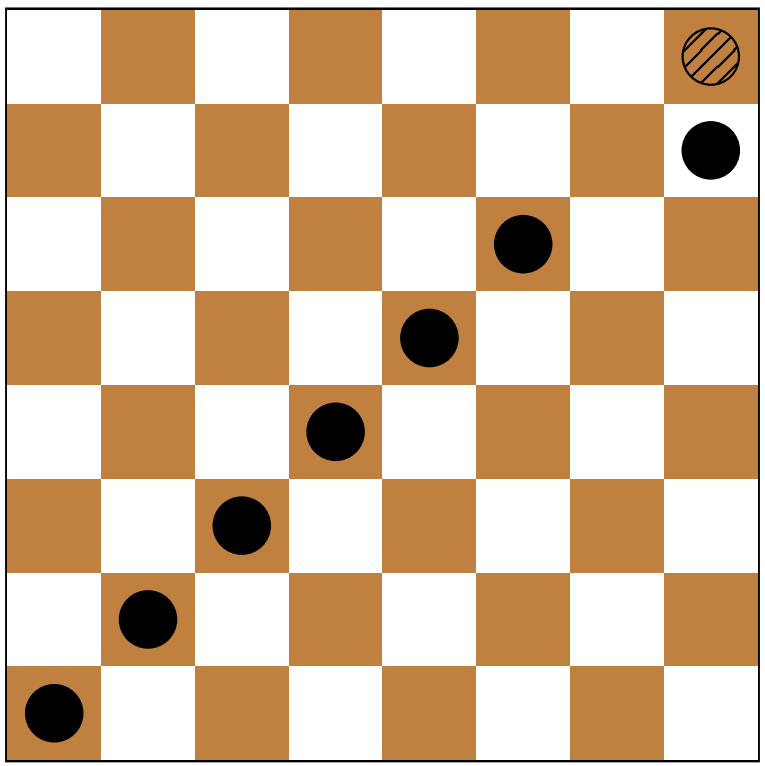
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



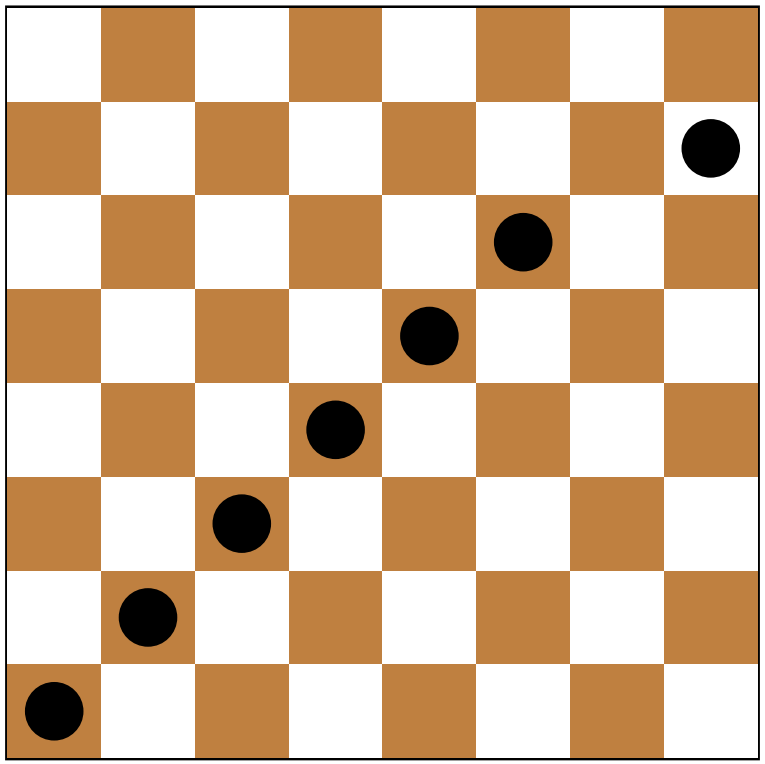
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



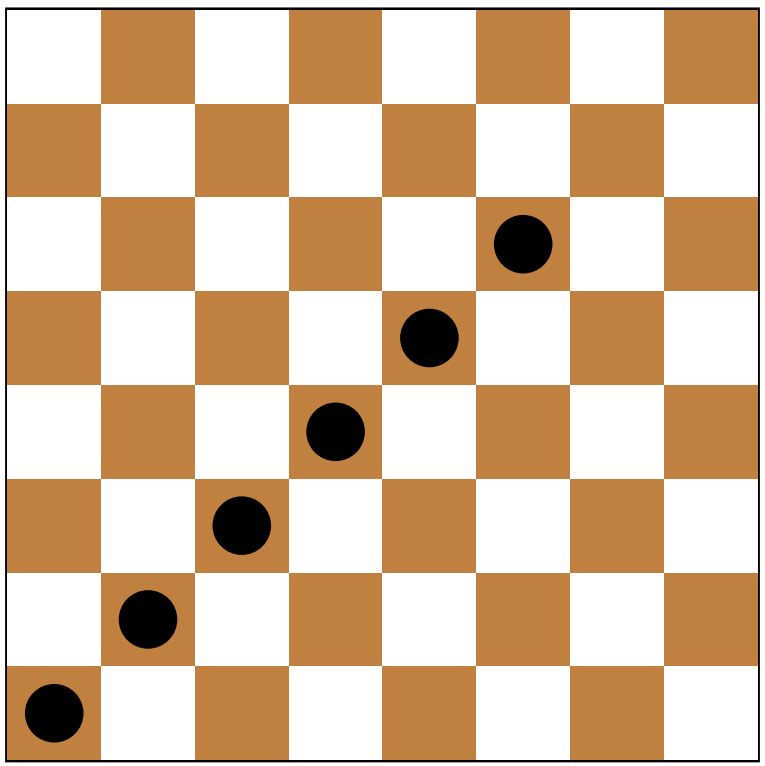
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



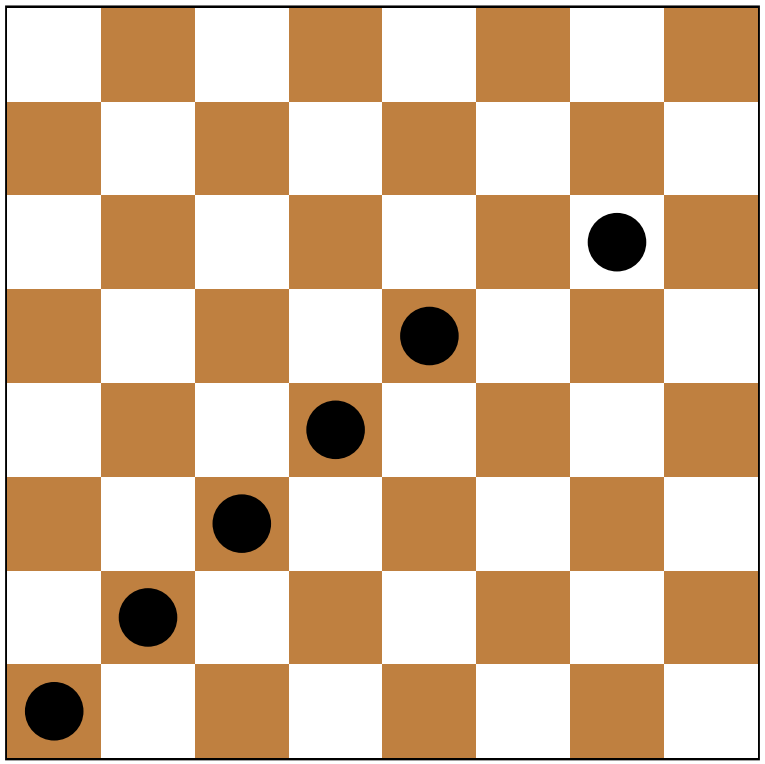
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



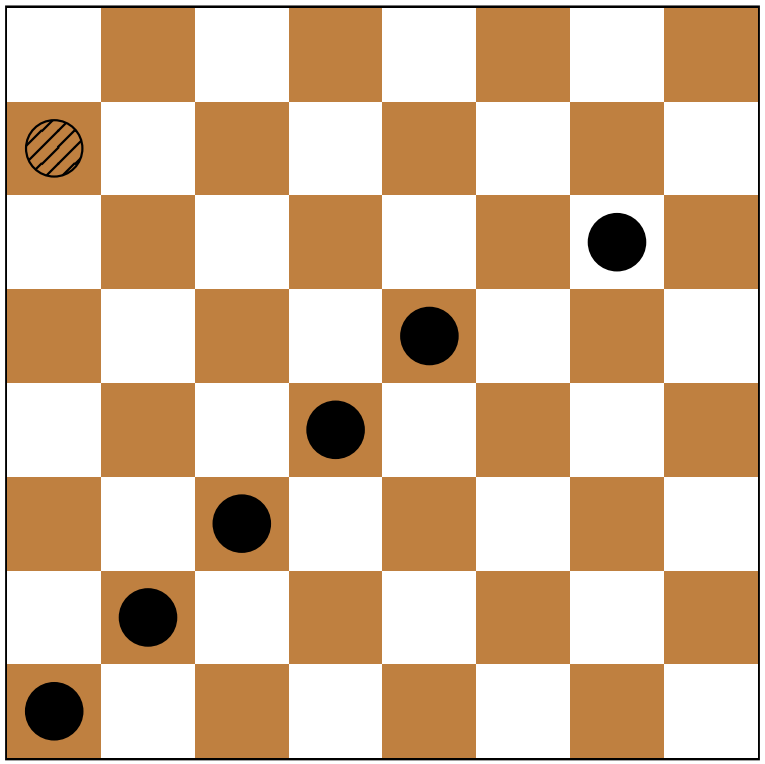
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



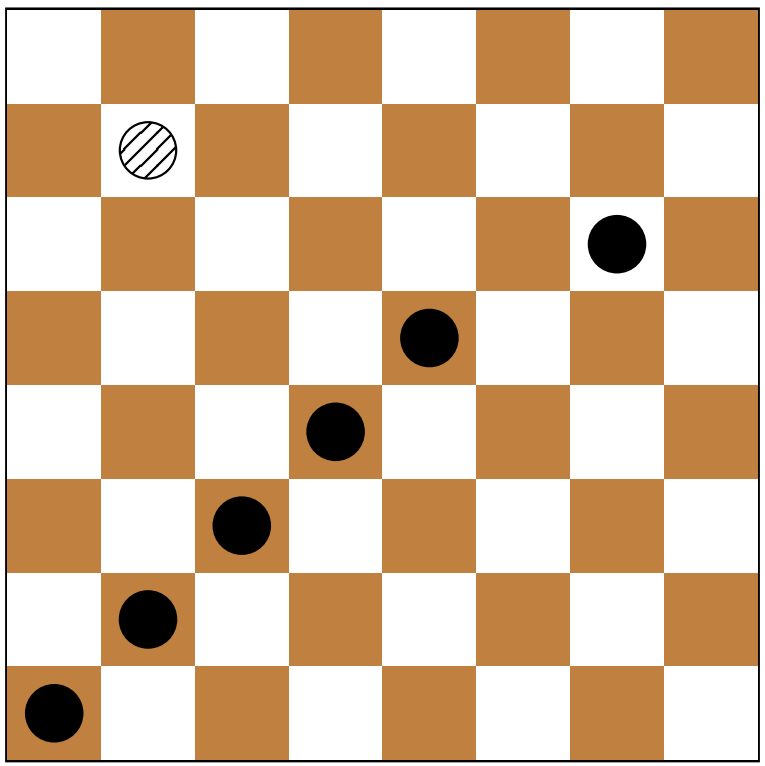
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



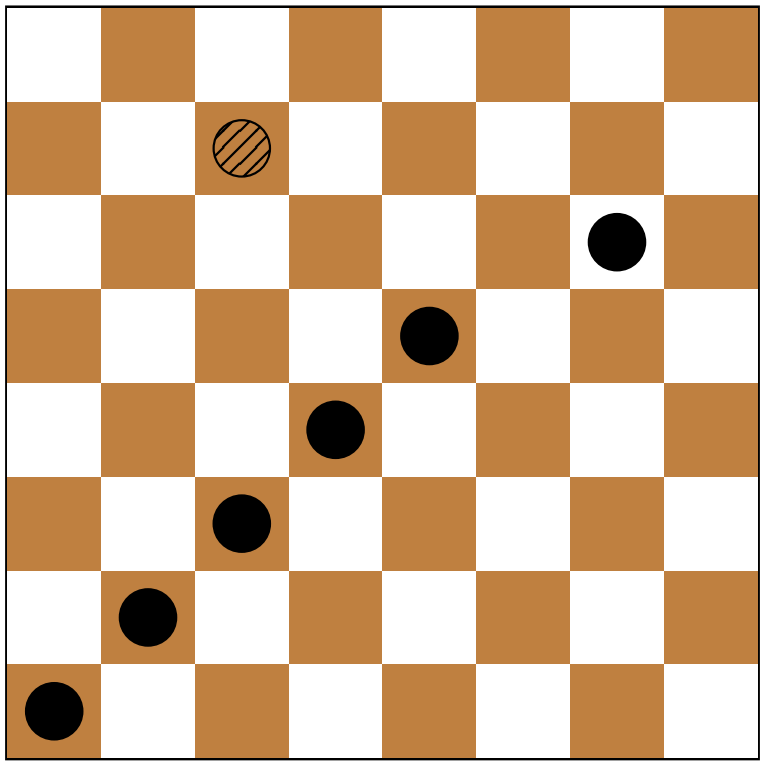
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



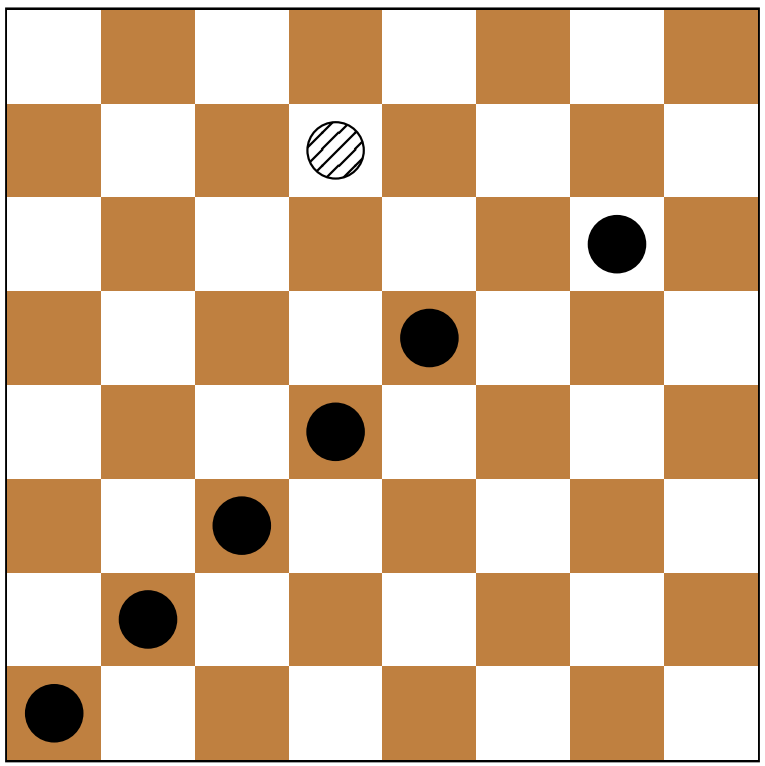
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



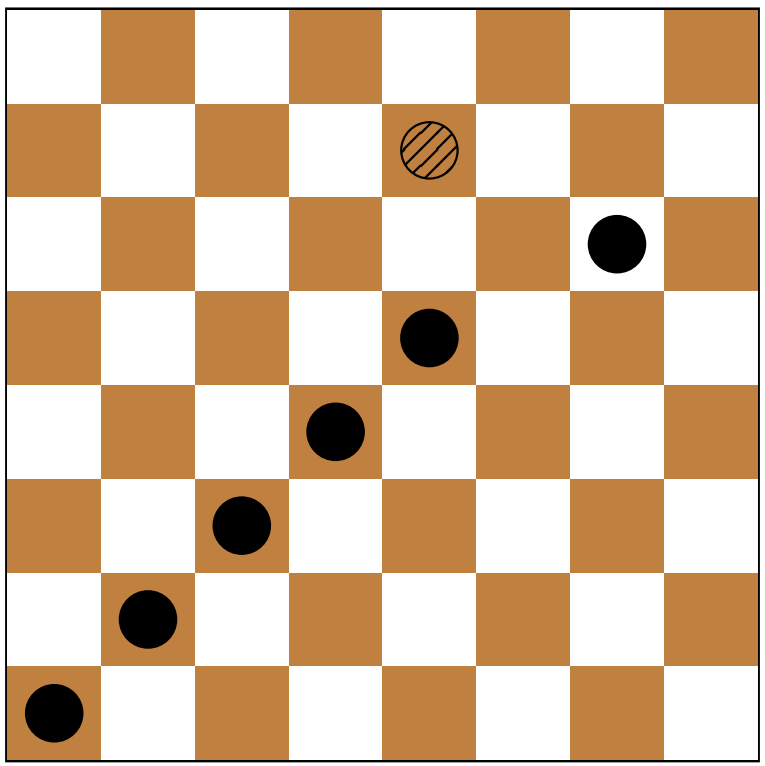
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



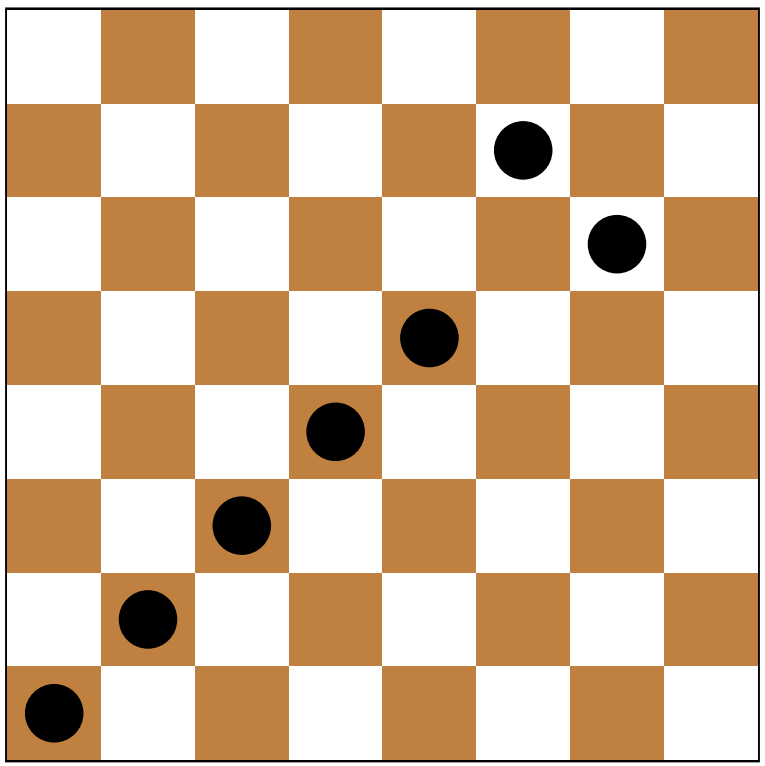
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



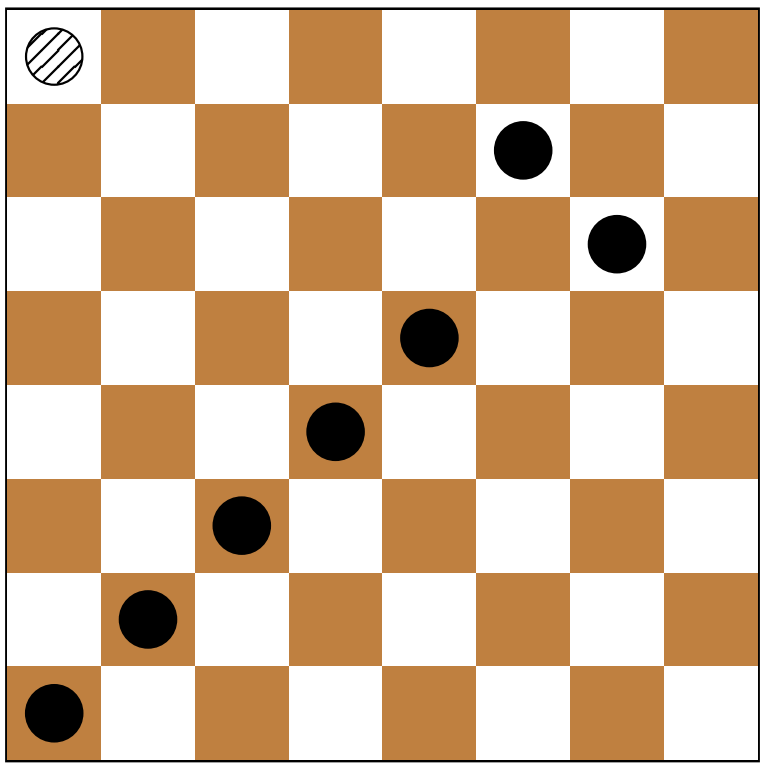
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



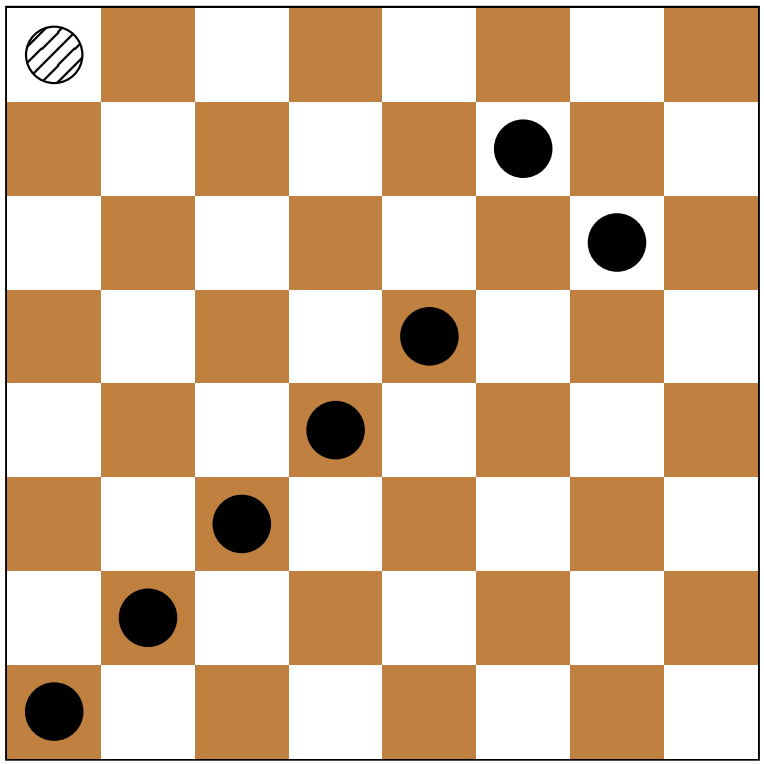
Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



Backtracking: example

Example: generate all ways to place 8 rooks on a chessboard so that none of them is exposed to capture.



And so on. . .

Backtracking

When designing a backtracking algorithm, the goal should always be to restrict the set of extensions as much as possible as early as possible.

Don't be afraid to use heuristics to achieve this goal.

Backtracking

When designing a backtracking algorithm, the goal should always be to restrict the set of extensions as much as possible as early as possible.

Don't be afraid to use heuristics to achieve this goal.

Navigation

Problem

Given two vertices u and v of a graph G, enumerate all paths between u and v in lexicographic order.

Navigation

- The partial candidates are paths starting at u.
- The extensions of a path are obtained by adding one edge at the end. Process the extensions in lexicographic order.
- A path is accepted if it ends at v.
- A path can be rejected if there is no way to reach v from the endpoint (check with DFS).

Navigation

- The partial candidates are paths starting at u.
- The extensions of a path are obtained by adding one edge at the end. Process the extensions in lexicographic order.
- A path is accepted if it ends at v.
- A path can be rejected if there is no way to reach v from the endpoint (check with DFS).

Navigation

- The partial candidates are paths starting at u.
- The extensions of a path are obtained by adding one edge at the end. Process the extensions in lexicographic order.
- A path is accepted if it ends at v.
- A path can be rejected if there is no way to reach v from the endpoint (check with DFS).

Navigation

- The partial candidates are paths starting at u.
- The extensions of a path are obtained by adding one edge at the end. Process the extensions in lexicographic order.
- A path is accepted if it ends at v.
- A path can be rejected if there is no way to reach v from the endpoint (check with DFS).