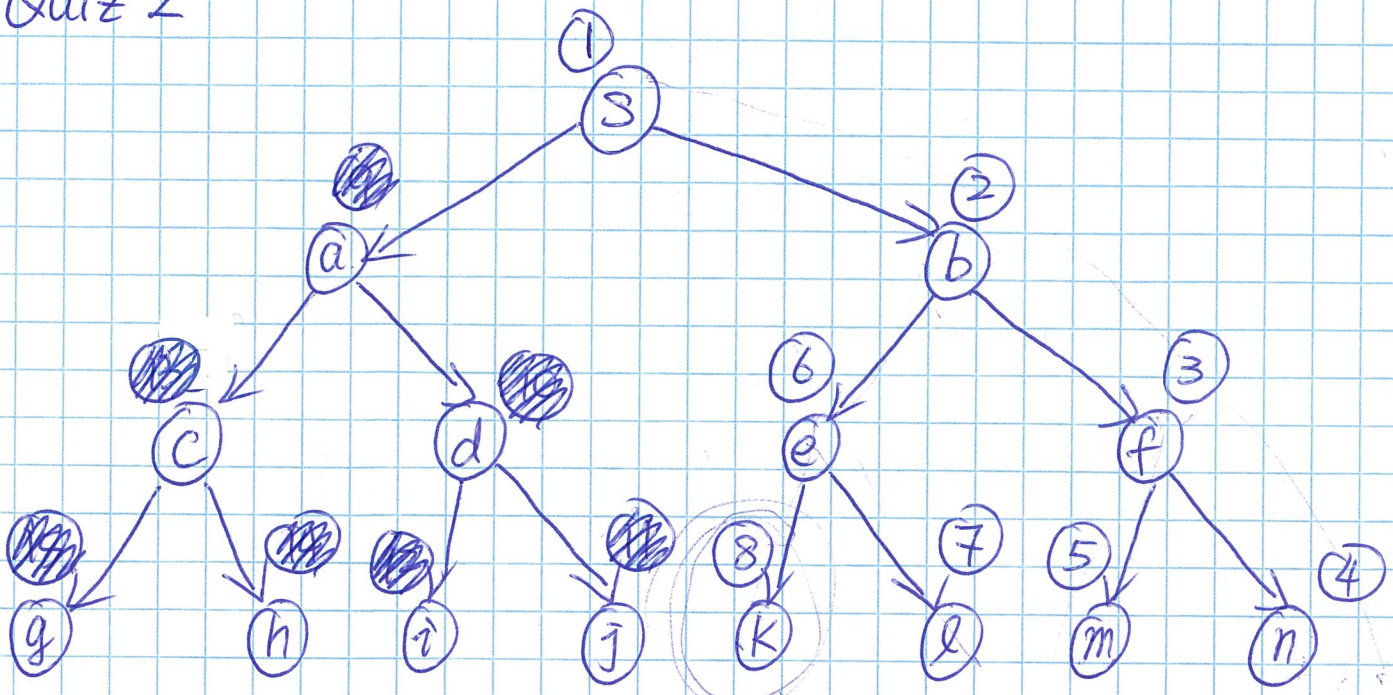
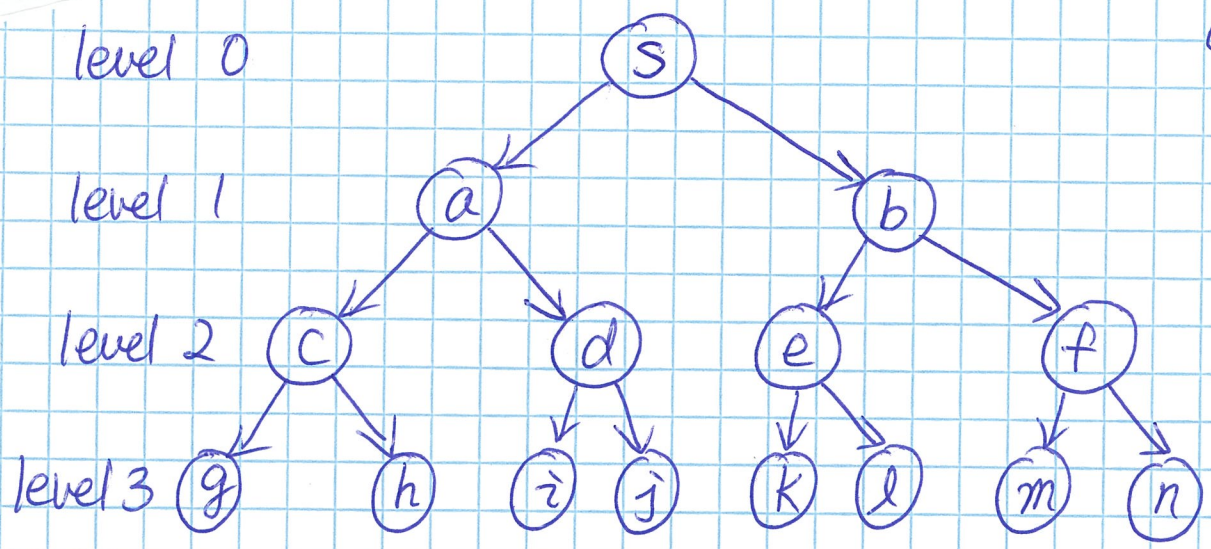
Quiz 2



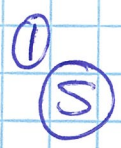
The nodes expanded by Depth-First Search.

We stop at (k) because it's a goal node.

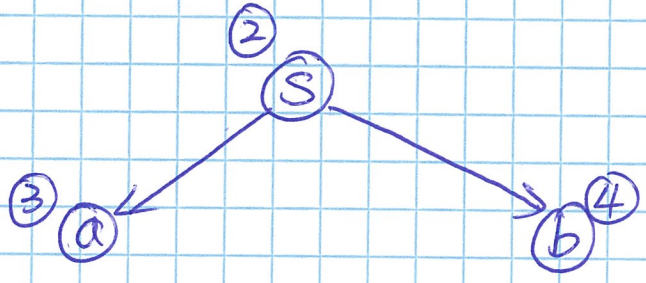
Quiz 2



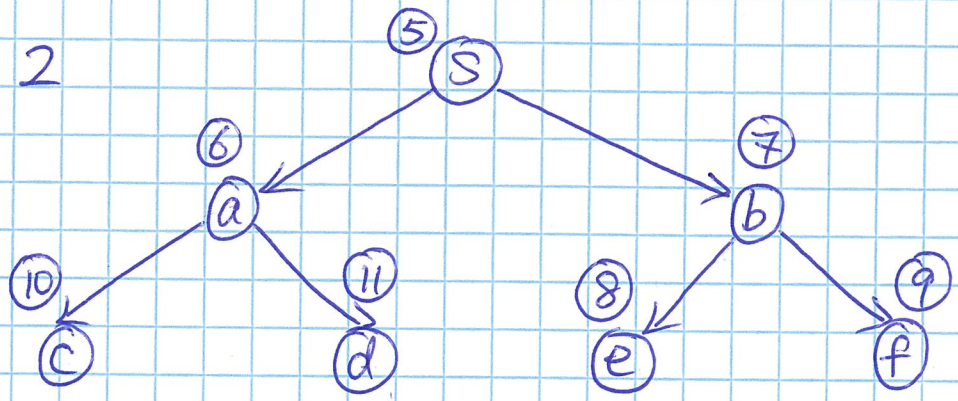
depth limit = 0



depth limit = 1



depth limit = 2



The nodes added to the frontier by Iterative-Deepening Search

The Backtracking Search Algorithm

Algorithm 1 BACKTRACK(assignment, csp)

```
1:  if assignment is complete then return assignment
2:  var ← SELECT-UNASSIGNED-VARIABLE(csp)
3:  for all value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
4:      if value is consistent with assignment then
5:          add {var = value} to assignment
6:          inferences ← INFERENCE(csp, var, value)
7:          if inferences ≠ failure then
8:              add inferences to assignment
9:              result ← BACKTRACK(assignment, csp)
10:             if result ≠ failure then
11:                 return result
12:             end if
13:         end if
14:     end if
15:     remove {var = value} and inferences from assignment
16: end for
17: return false
```

→ Choose an unassigned variable.

→ try all possible values of the variable.

→ violates a constraint.

→ forward checking
- reduces the domains.

→ keep the reduced domains.

undo any change we did

restore domains.

Backtracking Search 4 Queens Problem Lecture 7

current assignment $\{\}$

variable x_0 value 0

$\{x_0 = 0\}$

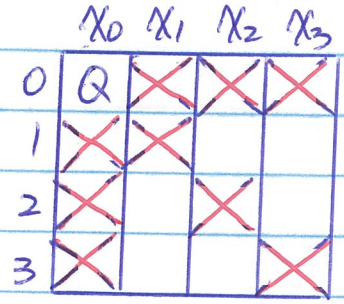
forward checking.

remove 1, 2, 3 from dom(x_0)

remove 2, 3 from dom(x_1)

remove 0, 2 from dom(x_2)

remove 0, 3 from dom(x_3)



current assignment $\{x_0 = 0\}$

$x_1 \in \{2, 3\}$ $x_2 \in \{1, 3\}$ $x_3 \in \{1, 2\}$

variable x_1 value 3 (least-constraining-value heuristic)

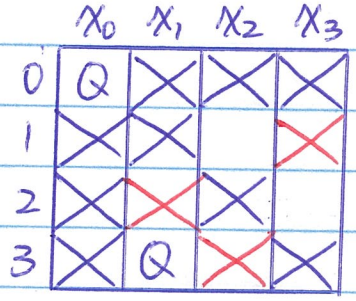
$\{x_1 = 3\}$

forward checking.

remove 2 from dom(x_1)

remove 3 from dom(x_2)

remove 1 from dom(x_3)



current assignment $\{x_0 = 0, x_1 = 3\}$

$x_2 \in \{1\}$ $x_3 \in \{2\}$

variable x_2 value 1

$\{x_2 = 1\}$

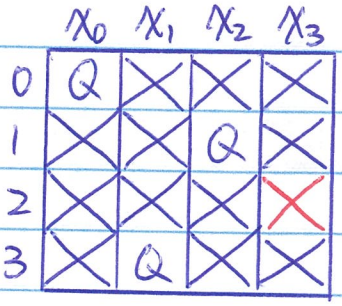
forward checking

remove 2 from dom(x_3)

dom(x_3) is empty. failure!

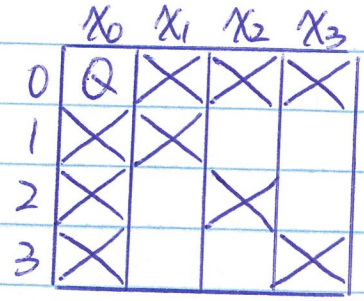
no other value of x_2 to try.

backtrack to x_1.

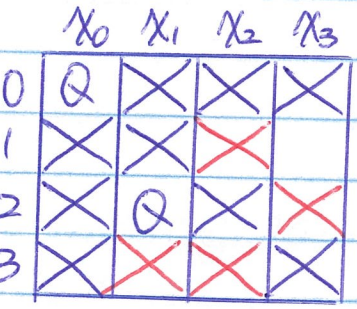


Backtracking Search 4-Queens Problem

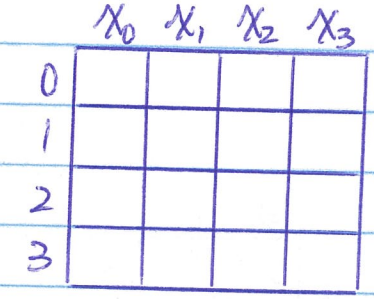
before trying another value of X_1
current assignment $\{X_0 = 0\}$.
$X_1 \in \{2,3\}$, $X_2 \in \{1,3\}$, $X_3 \in \{1,2\}$



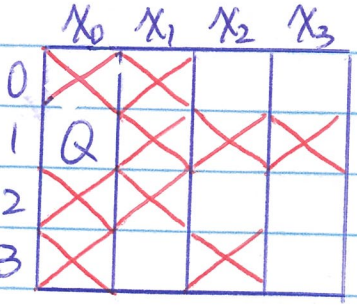
Variable X_1 Value 2
$\{X_1 = 2\}$
forward checking.
remove 3 from dom(X_1)
remove 1, 3 from dom(X_2)
remove 2 from dom(X_3)
dom(X_2) is empty. failure!
no other value of X_1 to try.
backtrack to X_0.



before trying another value of X_0
current assignment $\{\}$
$X_0 \in \{0,\ldots,3\}$ $X_1 \in \{0,\ldots,3\}$
$X_2 \in \{0,\ldots,3\}$ $X_3 \in \{0,\ldots,3\}$



Variable X_0 Value 1
$\{X_0 = 1\}$
forward checking.
remove 0, 2, 3 from dom(X_0)
remove 0, 1, 2 from dom(X_1)
remove 1, 3 from dom(X_2)
remove 1 from dom(X_3)
current assignment $\{X_0 = 1\}$
$X_1 \in \{3\}$, $X_2 \in \{0,2\}$ $X_3 \in \{0,2,3\}$



Backtracking Search 4-Queens Problem

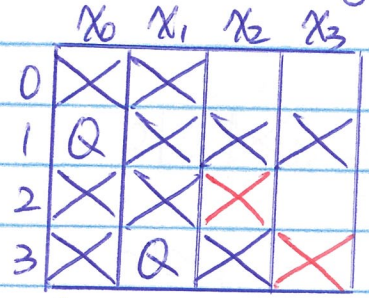
variable x_1 value 3 (minimum-remaining-values heuristic)

$\{x_1 = 3\}$

forward checking

remove 2 from dom(x_2)

remove 3 from dom(x_3)



current assignment $\{x_0 = 1, x_1 = 3\}$

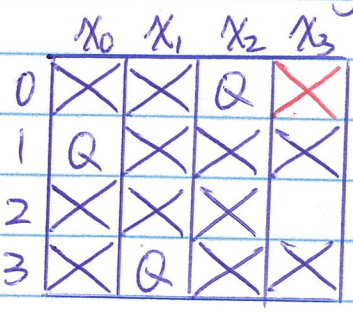
$x_2 \in \{0\}$ $x_3 \in \{0, 2\}$

variable x_2 value 0 (minimum-remaining-values heuristic)

$\{x_2 = 0\}$

forward checking

remove 0 from dom(x_3)

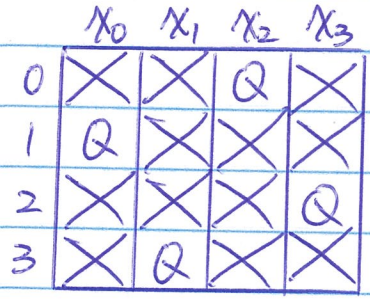


Current assignment $\{x_0 = 1, x_1 = 3, x_2 = 0\}$

$x_3 \in \{2\}$

variable x_3 value 2

$\{x_3 = 2\}$



solution: $\{x_0 = 1, x_1 = 3, x_2 = 0, x_3 = 2\}$

4Queens Problem Local Search

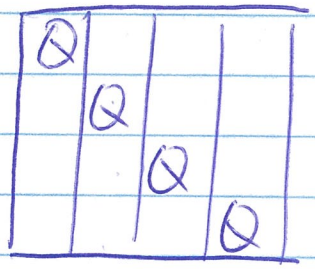
Variables: x_0, x_1, x_2, x_3 where x_i is the row position of the queen in column i. One queen per column.

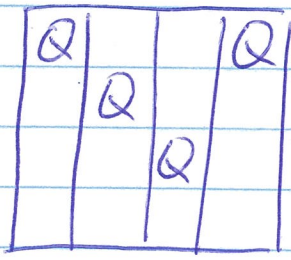
Domain: $\{0, 1, 2, 3\}$.

Constraints: —

Neighbour relation:

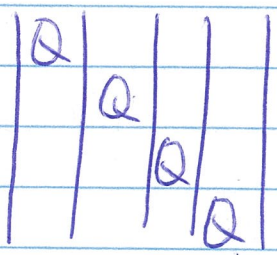
Version A: move a single queen to another square in the same column.

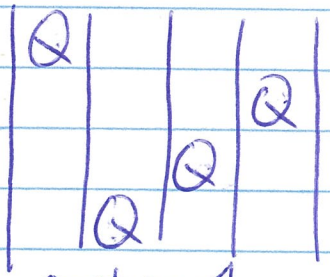




$3 * 4 = 12$.

Version B: swap the row positions of two queens.





cost = 4.

$$\binom{4}{2} = \frac{4*3}{2*1} = 6.$$

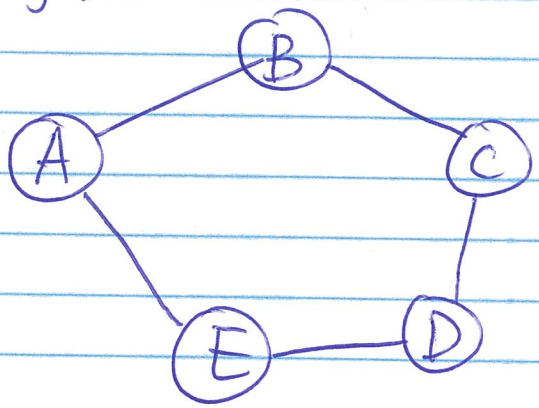
Cost function: The number of pairs of queens attacking each other directly or indirectly.

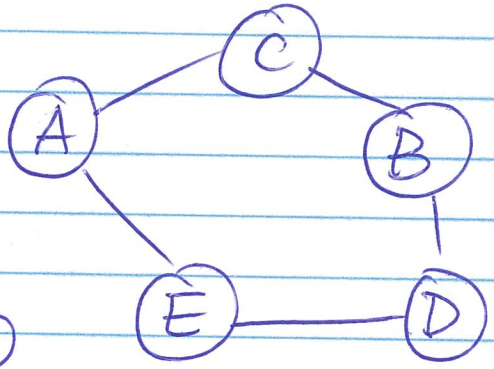
Traveling Salesperson Problem

State: a complete tour of all the cities starting from city A and ending with city A. The tour visits each city exactly once.

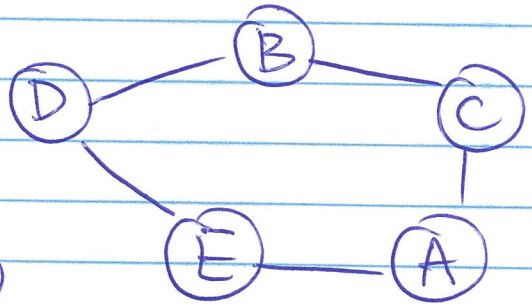
Cost function: The sum of the costs of all the edges on the tour.

Neighbour relation:

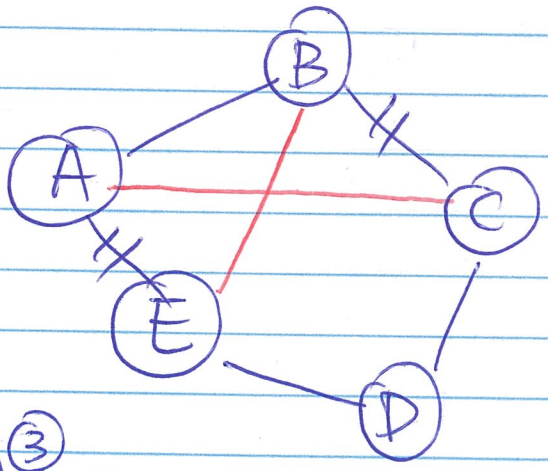




① swap neighbouring nodes

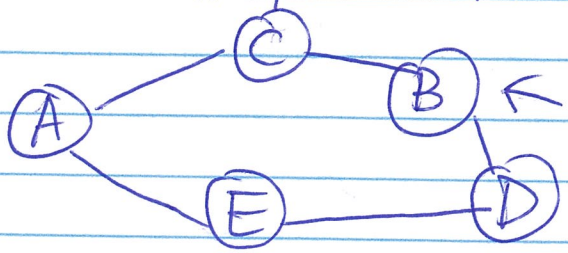


② swap any 2 nodes.



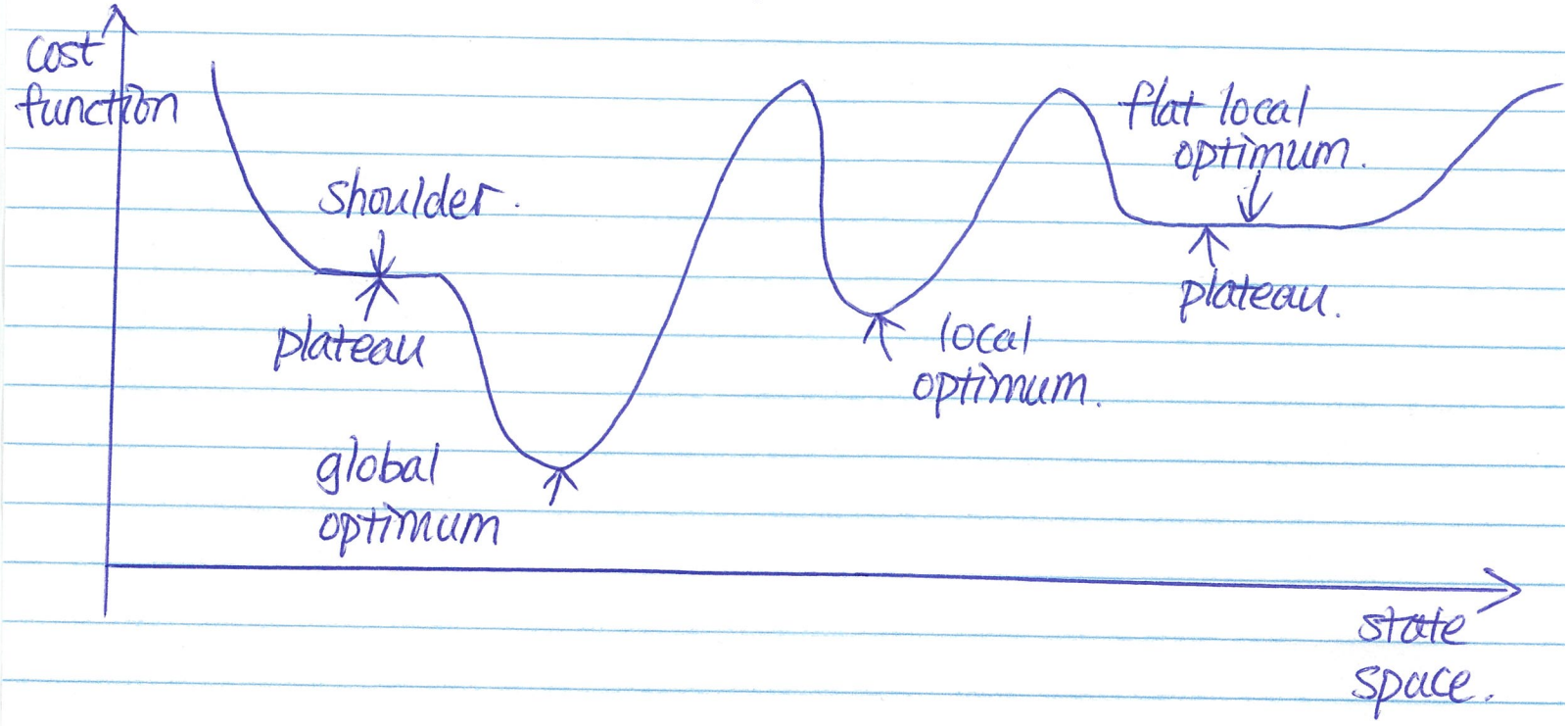
③ 2-swap.
reconnect 2 edges

④ move one node to a different position in the tour



← moved B from between A & C to between C & D.

A one-dimensional state space.



Local optima: A state s^* is locally optimal if $c(s^*) \leq c(s)$ for every neighbour ~~of~~ s of s^*.

A Plateau is a local optimum.