Try all possibilities for next decision. Recursively make remaining decisions.
**RecursiveNQueens(Q[1..n], r):**

```
if r = n + 1
    print Q
else
    for j ← 1 to n
        legal ← TRUE
        for i ← 1 to r - 1
            if (Q[i] = j) or (Q[i] = j + r - i) or (Q[i] = j - r + i)
                legal ← FALSE
        if legal
            Q[r] ← j
            RecursiveNQueens(Q[1..n], r + 1)
```

*Figure 2.1. Laquière's backtracking algorithm for the n-queens problem.*
\begin{figure}
\begin{algorithm}
\textbf{PLAYAnyGAME}(X, player):
  \begin{algorithmic}
  \STATE if player has already won in state X
  \STATE \quad return GOOD
  \STATE if player has already lost in state X
  \STATE \quad return BAD
  \STATE for all legal moves X \rightarrow Y
  \STATE \quad if \text{PLAYAnyGAME}(Y, \neg player) = BAD
  \STATE \quad \quad return GOOD
  \STATE return BAD
  \end{algorithmic}
\end{algorithm}
\caption{How to play any game perfectly.}
\end{figure}
How do we describe subproblem?

Fix original input as "global" variable $A[k..n]$.
Specify subproblem using indices

Specification:
Let $\text{SPLITTABLE}(k) =$ True iff $A[k..n]$ can be split into English words False o/w
\[ \text{SPLITTABLE}(k): \]
\[
\begin{align*}
\text{if } k &\geq n \\
& \quad \text{return TRUE} \\
\text{for all } j &\leq k \text{ to } n \quad \text{// end of first word} \\
\text{if ISWORD}(k,j) \\
& \quad \text{if SPLITTABLE}(j+1) \\
& \quad \quad \text{return TRUE} \\
& \text{return FALSE}
\end{align*}
\]