Eight Queens Problem

- Placing 8 queens in nonattacking postions on a chess board requires $\binom{64}{8} = 4426165368$ different configurations to be checked if exhaustive search is used.
- But we know that we don't have to see all configurations as no two queens can be placed in the same row.
- This reduces the chess board representation to a vector of eight elements. Each corresponds to a different column.
- So an exhaustive search based algorithm will be:

Eight Queens Problem

```
for (i_i = 1; i_1 <= 8; i_1++)
    for (i_2 = 1; i_2 <= 8; i_2++)
        ...
            for (i_8 = 1; i_8 <= 8; i_8++)
                sol = [i_1, i_2, ..., i_8]
                if (solution(sol)) {
                    write sol;
                    stop;
                }
```

- Number of position to be checked is $8^8 = 16777216$.
- Of course solution will stop after considering 1299852 positions.

Eight Queens Problem

- Since we used domain knowledge of chess to reduce number of postions to be checked we can as well take it a bit further.
- We know that no two queens can be placed in same column.
- So, the solution is a vector of 8 distinct elements $\{1, 2, \ldots, 8\}$.
- This means we need to check all possible permutations of 8 elements.

```
sol = initialPermutation;
while ((sol != finalPermutation) &&
          (not solution(sol)))
    sol = nextPermutation;
if (solution(sol)) write sol;
else
    write "there is no solution";
```


Eight Queens Problem

- The number of possible positions now reduces to 8!=40320.
- We can generate permutations of n integers as follows
- $\mathbf{T}$ initialized to $\{1,\ 2,\ \ldots, 8\}$, and call `permutation(1)` for generating the permutations.

```
permutation(i) {
    if (i==n) use(T)
    else
        for (j=i; j≤n) {
            exchange(T(i), T(j));
            permutation(i+1);
            exchange(T(i), T(j));
        }
}
```


Eight Queens Problem

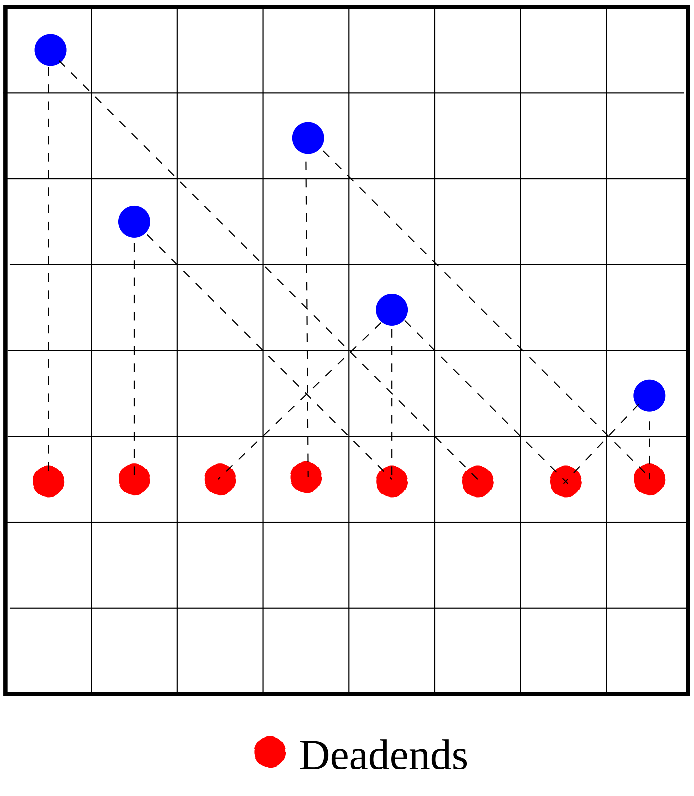
- We first reformulate 8-queen problem as a tree searching problem.
- A vector $V[1 \ldots k]$ is called k-**promising** for $0 \leq k \leq 8$, if none of the k-queens placed in positions $(1, V[1]), (2, V[2]), \ldots, (k, V[k])$ attack each other.
- That is V is k-**promising** if for every pair of intgers $i \neq j$
  - $V[i] - V[j] \notin \{i - j, 0, j - i\}$.
  - For $k \leq 1$ any vector is k-**promising**
- Solution to 8 queens problem corresponds to vectors that are 8-**promising**.

k-Promising Vectors

- Let N be the set of k-**promising** vectors, $0 \leq k \leq 8$.
- Let $G = (N, E)$ be the directed graph such that $(U, V) \in E$ if and only if there exists an integer k, $0 \leq k < 8$. that
  - U is k-**promising**
  - V is $(k+1)$-**promising**
  - $U[i] = V[i]$ for every $i \in [1 \ldots k]$
- Graph G is a tree. Its root is the empty vector corresponding to $k = 0$.

k-Promising Vectors

- Its leaves are either solutions of 8 queen problems or postions which will never lead to a solution ($k = 8$). Essentially deadends ($k < 8$) like [1, 4, 2, 5, 8].



k-Promising Vectors

- We know due to third conditions k-**promising** vector are extensions of their $k-1$ counter-parts.
- So with a $(k-1)$-**promising** vector available, only one needs to check positions where last queen should be placed.
- The checking can be made faster provided
  - With each promising node we associate the set of columns, positive diagonal and negative diagonals controlled by others queens already on the board.

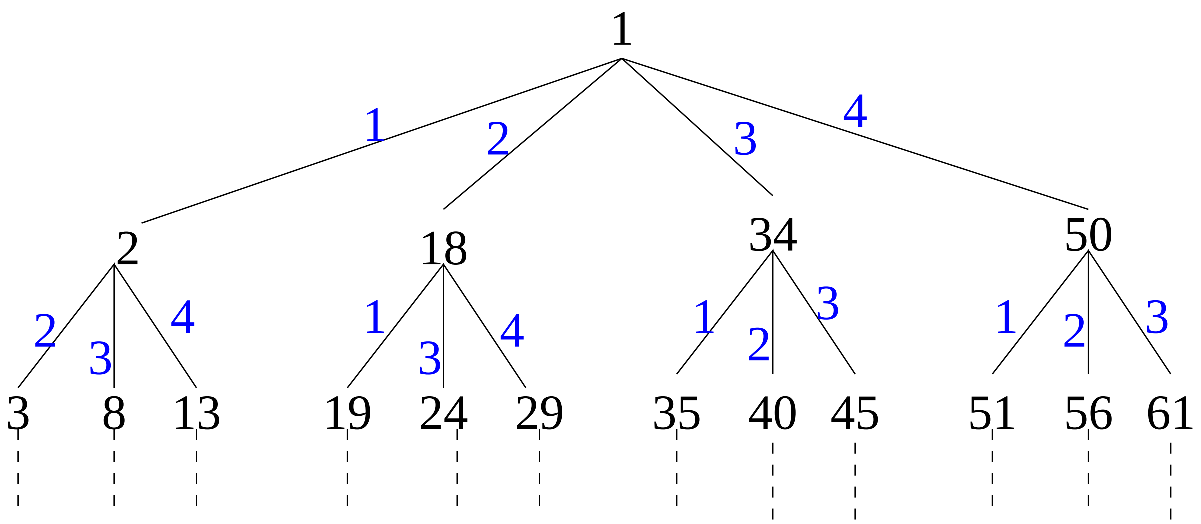
Eight Queens Problem

```
eightQueens(k, col, D⁻, D⁺) {
   if (k==8) print sol;
   else
      for(j=1; j≤ 8; j++)
         if ((j ∉ col) && (j-k ∉ D⁺)
               && (j+k ∉ D⁻)){
            sol[k+1] = j;
            eightQueens(k+1, col∪{j},
                           D⁺∪{j-k},
                           D⁻∪{j+k})
         }
}
```


Four Queen Solution Space

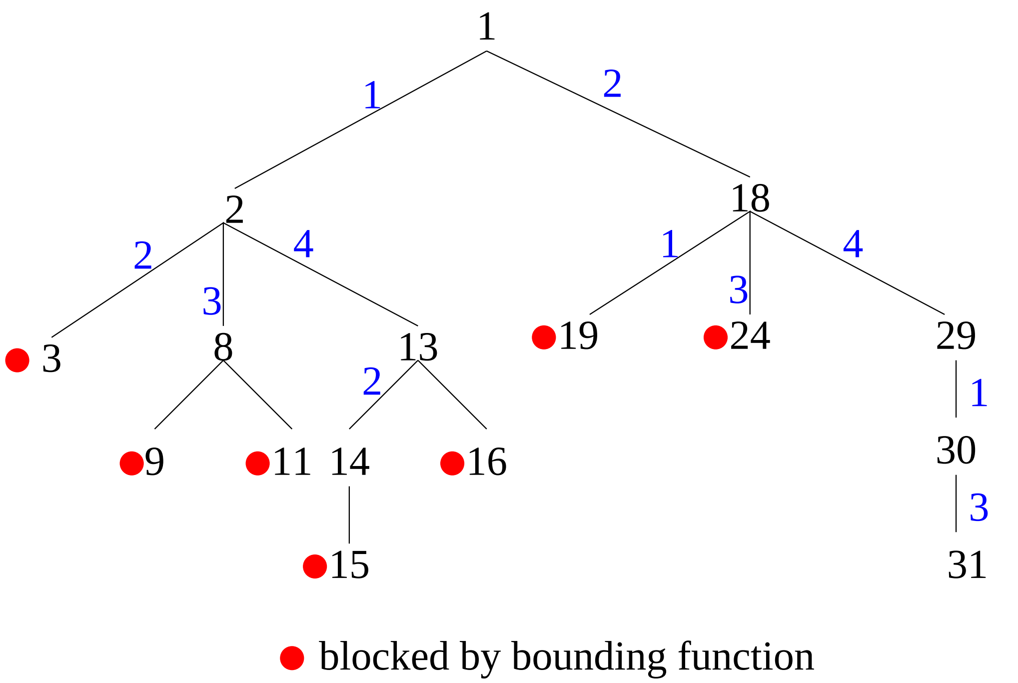
- Full solution space (implicit tree) generated by 4-queen problem is provided below.
- The problem constraint that no two queens can belong to
  - the same row, or
  - to the same column

  have been used for generation of the tree, no additional bounding function have been used here.



Generated by Backtrack

- The state space is reduced by applying bounding function as indicated.



Estimate of Efficiency

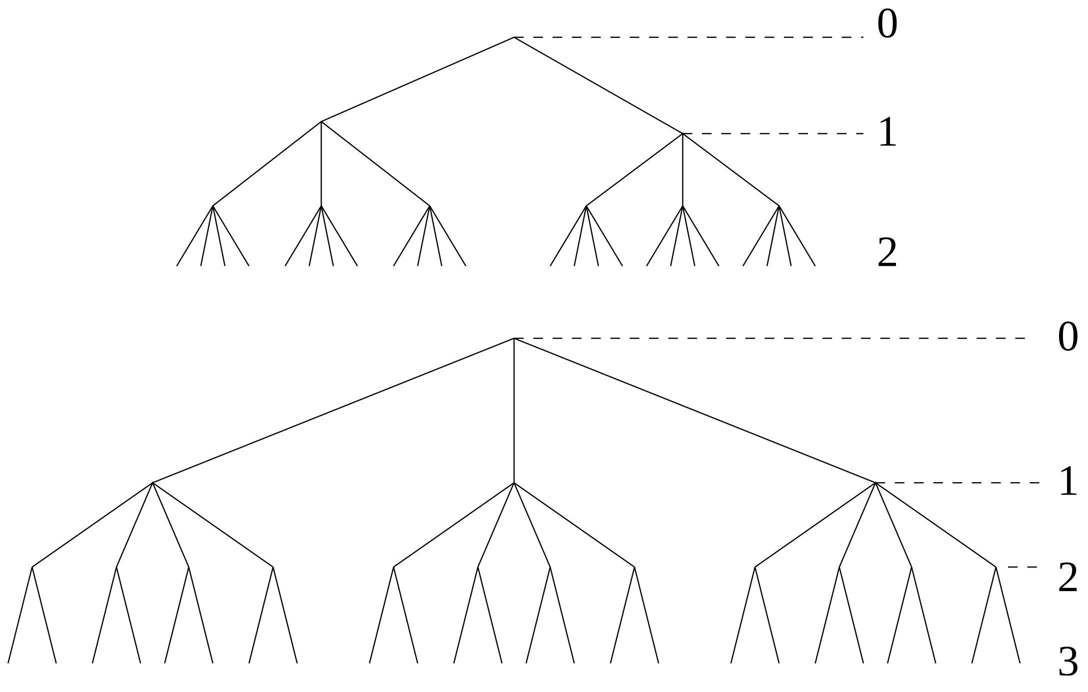
- Depends of four factors:
  1. Time to generate next choice.
  2. The number of choices satisfying the constraints,
  3. Time for the bounding function.
  4. Number of choices satisfying the bounding function.
- Bounding function is considered as good if it helps to eliminate substantial number of nodes in solution space.
- But bounding function may take substantial time to evaluate.
- The requirement should be reduction in overall computation time.

Estimate of Efficiency

- General principle is to rearrange the search space.
- In many problems, sets S_i - the set from which next element x_i can be chosen - can be take in any order.
- If that be the case then select the set with fewest elements to pick the next element.
- Effectiveness of rearrangement can be understood from the example of solution state spaces shown in figure which appear next.

Estimate of Efficiency

- In first case if we are able to eliminate 1 node at level 1 then it is can eliminate 12 nodes at level 2.
- Whereas if 1 node in level 1 is eliminated in 2nd case then only 8 nodes are eliminated.



Estimate of Efficiency

- But, as we noticed, it does not work in n-queen problem.
- There may be many such situation where selecting set with fewest elements to pick the next won't work.
- Suppose we have static bounding function that leaves m_i nodes at level i for $i = 1, 2, \ldots$.
- Also assume that the nodes at same level have same degree.
- Then number of nodes generated by backtrack

$$1 + \sum_{i=1}^{k} \left(\prod_{j=1}^{j=i} m_j \right)$$

Eight Queens Problem

```
estimate() {
   m=1; r=1;
   for (i=1; i≤k; i++) {
      T_k = {x[k]| x[k]∈
            T(x[1], ..., x[k-1]) and
            B_k(x[1], ..., x[k])}
      if (size(T_k)==0) exit();
      r = r*size(T_k);
      m = m+r;
      x[k] = choose(T_k);
   }
   return m;
}
```

Estimate of Efficiency

- Using the static bounding function based estimate the number of nodes in state space for 8-queen problem is

$$1 + \sum_{j=0}^{8} \left(\prod_{i=0}^{j} (8 - i) \right) = 69,281$$

- Whereas suppose 5 following configurations 8×8 of chess board are generated through `estimate()` function. The number of nodes in each trial is indicated below.

$$\boxed{\begin{array}{rcr} (2, 4, 1, 3, 5) & = & 1649, \\ (4, 6, 3, 5, 7, 1) & = & 769, \\ (1, 8, 6, 5, 7, 2, 4) & = & 1401, \\ (1, 3, 5, 2, 4) & = & 1977, \\ (3, 6, 2, 7, 1, 4, 8, 5) & = & 2339. \end{array}}$$

- Above 5 trials give an average of 1625 nodes.

