8 Queens at Compile Time: A Journey from C++03 to Modern C++

Bloomberg

Engineering

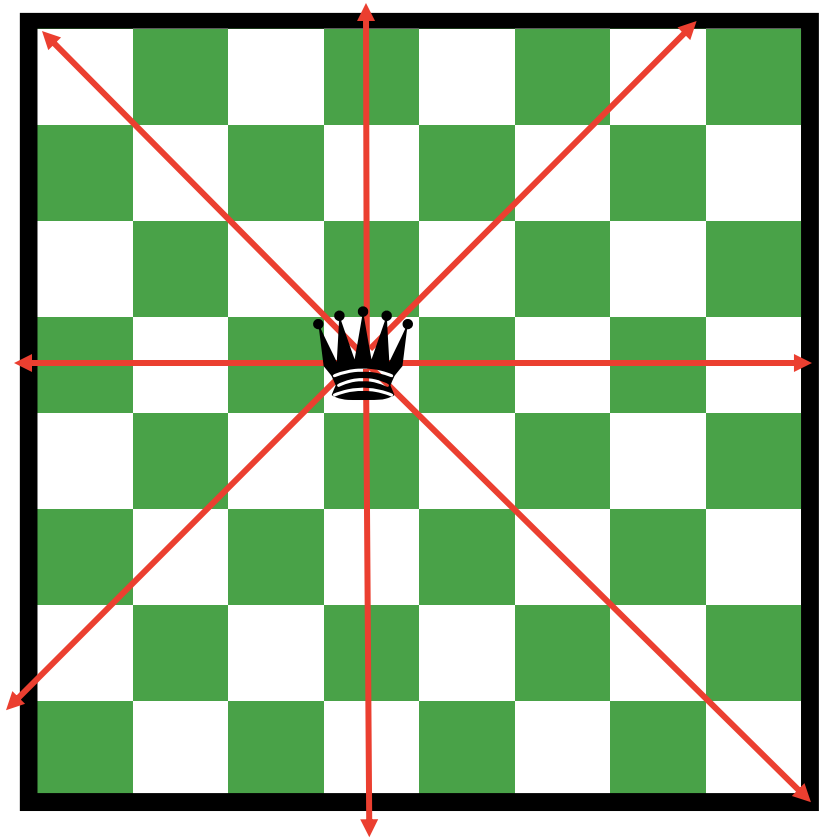
C++ Day 2025
October 25, 2025

Marco Marcello, Team Lead
Jonathan Marriott, Software Engineer

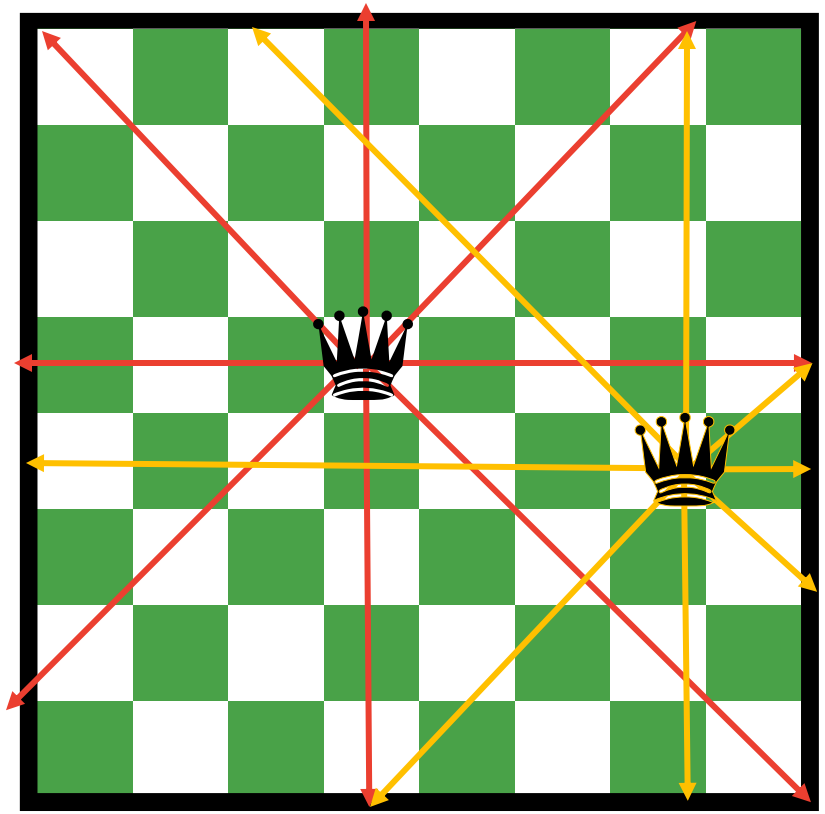
TechAtBloomberg.com

Source of generated image on this slide: ChatGPT

Intro to the Eight Queens Problem

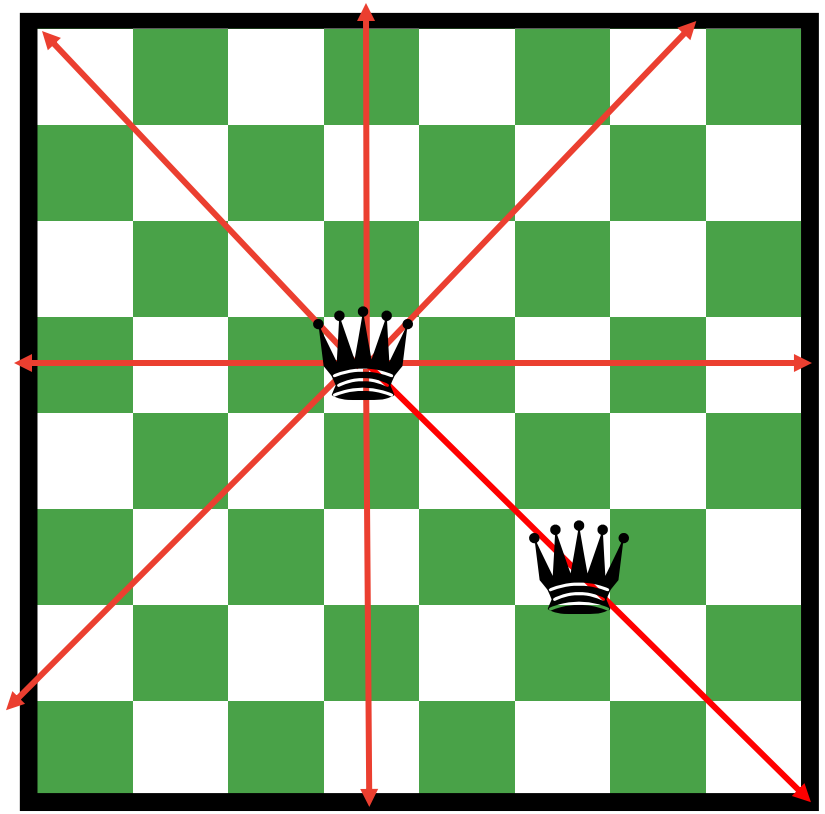


Intro to the Eight Queens Problem



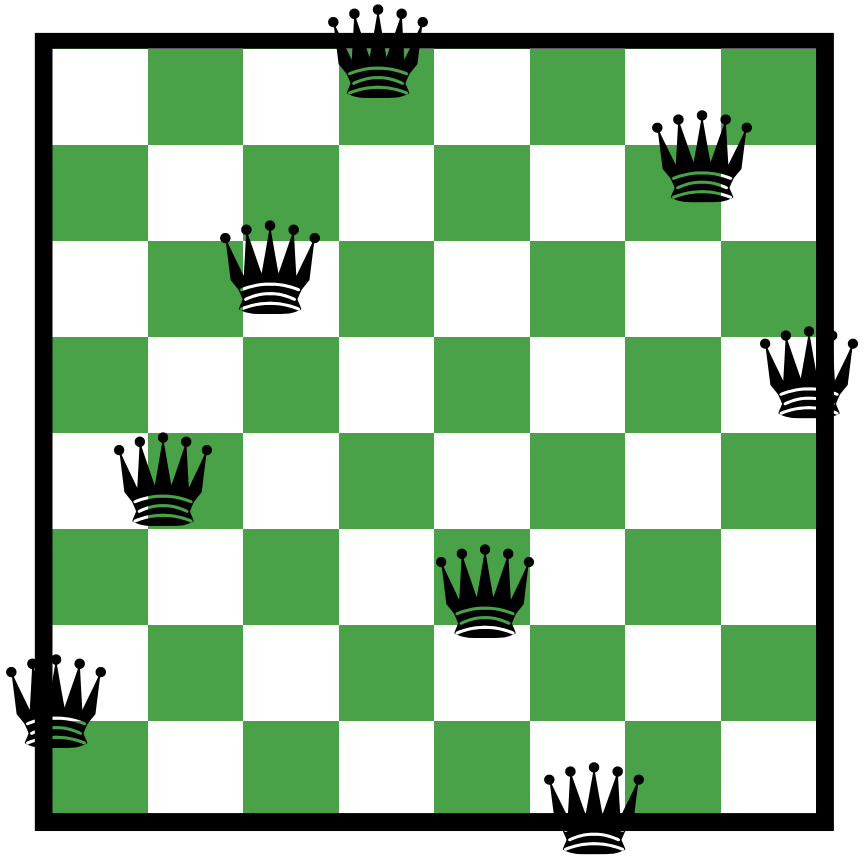
Valid Placement

Intro to the Eight Queens Problem



Invalid Placement

Intro to the Eight Queens Problem



Valid Solution

64 choose 8 =
4,426,165,368 arrangements

92 valid solutions

Source of generated image on this slide: ChatGPT

Template Metaprogramming solution C++03

1. Representation
2. Manipulators
3. Solution lists
4. Merging
5. Attack logic
6. Backtracking
7. Instantiation
8. Run-time logic

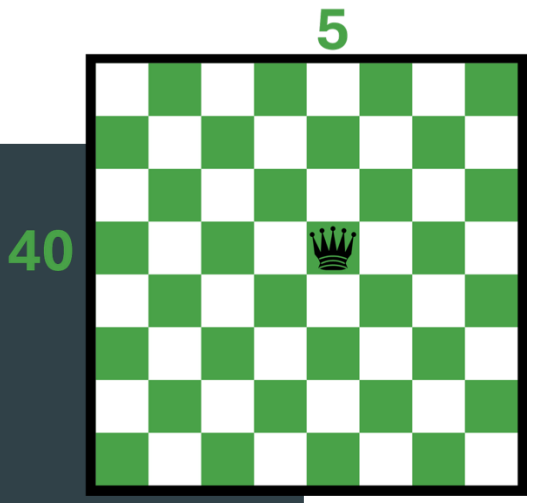
Bloomberg

Engineering

1. Representation

```
 1 template <int VALUE>
 2 struct static_value_square_coordinates {
 3     static const int value = VALUE;
 4
 5     operator int () const{
 6         return VALUE;
 7     }
 8
 9     static const int col = value % 10;
10     static const int row = value - col;
11 };
12
13 #define SQUARE(N) static_value_square_coordinates<N>
```

1. Representation




```
 1 template <int VALUE>
 2 struct static_value_square_coordinates {
 3     static const int value = VALUE;
 4
 5     operator int () const{
 6         return VALUE;
 7     }
 8
 9     static const int col = value % 10;
10     static const int row = value - col;
11 };
12
13 #define SQUARE(N) static_value_square_coordinates<N>
```

1. Representation

```
 1 template <int VALUE>
 2 struct static_value_square_coordinates {
 3     static const int value = VALUE;
 4
 5     operator int () const{
 6         return VALUE;
 7     }
 8
 9     static const int col = value % 10;
10     static const int row = value - col;
11 };
12
13 #define SQUARE(N) static_value_square_coordinates<N>
```

1. Representation

```
 1 struct empty {};
 2
 3 template <
 4     typename T0 = empty,
 5     typename T1 = empty,
 6     typename T2 = empty,
 7     typename T3 = empty,
 8     typename T4 = empty,
 9     typename T5 = empty,
10     typename T6 = empty,
11     typename T7 = empty
12 >
13 struct solution {
14 };
15
16 typedef solution<SQUARE(11), SQUARE(25), SQUARE(38), SQUARE(46),
17   SQUARE(53), SQUARE(67), SQUARE(72), SQUARE(84)> SampleSolution;
18
19 typedef solution<SQUARE(11), SQUARE(25), SQUARE(38)> SampleSolutionPartial;
```

1. Representation

```
 1 template <size_t N, typename SOLUTION>
 2 struct typeat;
 3
 4 template <typename T0, typename T1,
 5           typename T2, typename T3,
 6           typename T4, typename T5,
 7           typename T6, typename T7>
 8 struct typeat<0,
 9               solution<T0,T1,T2,T3,T4,T5,T6,T7> > {
10     typedef T0 type;
11 };
12 //... all specializations for N=0..7
```

1. Representation

```
1 template<typename T>
2 struct is_empty_type {
3     static const bool value = false;
4 };
5
6 template<>
7 struct is_empty_type<empty> {
8     static const bool value = true
9 };
```

1. Representation

```
 1 template <typename CONT, int N>
 2 struct first_N_elems_not_empty {
 3     static const bool value =
 4       !is_empty_type<typename typeat<N-1,CONT>::type>::value &&
 5       first_N_elems_not_empty<CONT, N-1>::value;
 6 };
 7
 8 template <typename CONT>
 9 struct first_N_elems_not_empty<CONT, 0> {
10     static const bool value = true;
11 };
```

1. Representation

```
 1 template <typename CONT, int N>
 2 struct first_N_elems_not_empty {
 3     static const bool value =
 4       !is_empty_type<typename typeat<N-1,CONT>::type>::value &&
 5       first_N_elems_not_empty<CONT, N-1>::value;
 6 };
 7
 8 template <typename CONT>
 9 struct first_N_elems_not_empty<CONT, 0> {
10     static const bool value = true;
11 };
```

1. Representation

```
 1 template <typename CONT, int N>
 2 struct first_N_elems_not_empty {
 3     static const bool value =
 4       !is_empty_type<typename typeat<N-1,CONT>::type>::value &&
 5       first_N_elems_not_empty<CONT, N-1>::value;
 6 };
 7
 8 template <typename CONT>
 9 struct first_N_elems_not_empty<CONT, 0> {
10     static const bool value = true;
11 };
```

2. Manipulators

```
 1 template <typename SOLUTION>
 2 struct pop_front;
 3
 4 template <typename T0, typename T1,
 5           typename T2, typename T3,
 6           typename T4, typename T5,
 7           typename T6, typename T7>
 8 struct pop_front< solution<T0,T1,T2,T3,T4,T5,T6,T7> > {
 9     typedef solution<T1,T2,T3,T4,T5,T6,T7, empty> type;
10 };
11
```

2. Manipulators

```
 1 template <typename SOLUTION>
 2 struct pop_front;
 3
 4 template <typename T0, typename T1,
 5           typename T2, typename T3,
 6           typename T4, typename T5,
 7           typename T6, typename T7>
 8 struct pop_front< solution<T0,T1,T2,T3,T4,T5,T6,T7> > {
 9     typedef solution<T1,T2,T3,T4,T5,T6,T7, empty> type;
10 };
11
```

2. Manipulators

```
 1 template <typename SOLUTION, typename T>
 2 struct push_front;
 3
 4 template <typename T0, typename T1,
 5           typename T2, typename T3,
 6           typename T4, typename T5,
 7           typename T6, typename T7,
 8           typename T>
 9 struct push_front<solution<T0,T1,T2,T3,T4,T5,T6,T7>, T> {
10     typedef solution<T, T0,T1,T2,T3,T4,T5,T6> type;
11 };
```

2. Manipulators

```
 1 template <typename SOLUTION, typename T>
 2 struct push_front;
 3
 4 template <typename T0, typename T1,
 5           typename T2, typename T3,
 6           typename T4, typename T5,
 7           typename T6, typename T7,
 8           typename T>
 9 struct push_front<solution<T0,T1,T2,T3,T4,T5,T6,T7>, T> {
10     typedef solution<T, T0,T1,T2,T3,T4,T5,T6> type;
11 };
```

2. Manipulators

```
1 template <typename SOLUTION, typename T, bool COND>
2 struct push_front_if {
3     typedef SOLUTION type;
4 };
5
6 template <typename SOLUTION, typename T>
7 struct push_front_if<SOLUTION, T, true> {
8     typedef typename push_front<SOLUTION,T>::type type;
9 };
```

2. Manipulators

```
1 template <typename SOLUTION, typename T, bool COND>
2 struct push_front_if {
3     typedef SOLUTION type;
4 };
5
6 template <typename SOLUTION, typename T>
7 struct push_front_if<SOLUTION, T, true> {
8     typedef typename push_front<SOLUTION,T>::type type;
9 };
```

3. Solution lists

```
1 template <typename H=empty, typename T=empty>
2 struct solution_list {
3     typedef H head;
4     typedef T tail;
5 };
6
7 typedef solution_list<empty, empty> empty_solution_list;
```

3. Solution lists

```
 1 template < typename CONT, typename I>
 2 struct push_front_solution_to_list;
 3
 4 template < typename H, typename T, typename I>
 5 struct push_front_solution_to_list<solution_list<H,T>, I> {
 6     typedef solution_list<I, solution_list<H,T> > type;
 7 };
 8
 9 template< typename CONT, typename I, bool COND>
10 struct push_front_solution_to_list_if {
11     typedef CONT type;
12 };
13
14 template< typename CONT, typename I>
15 struct push_front_solution_to_list_if<CONT, I, true> {
16     typedef typename push_front_solution_to_list<CONT,I>::type type;
17 };
```

3. Solution lists

```
 1 template < typename CONT, typename I>
 2 struct push_front_solution_to_list;
 3
 4 template < typename H, typename T, typename I>
 5 struct push_front_solution_to_list<solution_list<H,T>, I> {
 6     typedef solution_list<I, solution_list<H,T> > type;
 7 };
 8
 9 template< typename CONT, typename I, bool COND>
10 struct push_front_solution_to_list_if {
11     typedef CONT type;
12 };
13
14 template< typename CONT, typename I>
15 struct push_front_solution_to_list_if<CONT, I, true> {
16     typedef typename push_front_solution_to_list<CONT,I>::type type;
17 };
```

3. Solution lists

```
 1 template < typename CONT, typename I>
 2 struct push_front_solution_to_list;
 3
 4 template < typename H, typename T, typename I>
 5 struct push_front_solution_to_list<solution_list<H,T>, I> {
 6     typedef solution_list<I, solution_list<H,T> > type;
 7 };
 8
 9 template< typename CONT, typename I, bool COND>
10 struct push_front_solution_to_list_if {
11     typedef CONT type;
12 };
13
14 template< typename CONT, typename I>
15 struct push_front_solution_to_list_if<CONT, I, true> {
16     typedef typename push_front_solution_to_list<CONT,I>::type type;
17 };
```

3. Solution lists

```
1 template < typename CONT>
2 struct pop_front_solution_from_list {
3     typedef empty type;
4 };
5
6 template < typename H, typename T>
7 struct pop_front_solution_from_list<solution_list<H,T> > {
8     typedef T type
9 };
```

3. Solution lists

```
1 template < typename CONT>
2 struct pop_front_solution_from_list {
3     typedef empty type;
4 };
5
6 template < typename H, typename T>
7 struct pop_front_solution_from_list<solution_list<H,T> > {
8     typedef T type
9 };
```

3. Solution lists

```
1 template <typename SINGLE>
2 struct get_solution_list_head {
3     typedef SINGLE type;
4 };
5
6 template <typename H, typename T>
7 struct get_solution_list_head<solution_list<H,T> > {
8     typedef H type;
9 };
```

3. Solution lists

```
1 template <typename SINGLE>
2 struct get_solution_list_head {
3     typedef SINGLE type;
4 };
5
6 template <typename H, typename T>
7 struct get_solution_list_head<solution_list<H,T> > {
8     typedef H type;
9 };
```

4. Merging

```
 1 template <typename SOLUTION_LIST_1, typename SOLUTION_LIST_2>
 2 struct merge_solution_lists;
 3
 4 template <typename SOLUTION_LIST_1>
 5 struct merge_solution_lists<SOLUTION_LIST_1, empty_solution_list> {
 6     typedef SOLUTION_LIST_1 type;
 7 };
 8
 9 template <typename SOLUTION_LIST_2>
10 struct merge_solution_lists<empty_solution_list, SOLUTION_LIST_2> {
11     typedef SOLUTION_LIST_2 type;
12 };
13
14 template <>
15 struct merge_solution_lists<empty_solution_list, empty_solution_list> {
16     typedef empty_solution_list type;
17 };
```

4. Merging

```
 1 template <typename SOLUTION_LIST_1, typename SOLUTION_LIST_2>
 2 struct merge_solution_lists;
 3
 4 template <typename SOLUTION_LIST_1>
 5 struct merge_solution_lists<SOLUTION_LIST_1, empty_solution_list> {
 6     typedef SOLUTION_LIST_1 type;
 7 };
 8
 9 template <typename SOLUTION_LIST_2>
10 struct merge_solution_lists<empty_solution_list, SOLUTION_LIST_2> {
11     typedef SOLUTION_LIST_2 type;
12 };
13
14 template <>
15 struct merge_solution_lists<empty_solution_list, empty_solution_list> {
16     typedef empty_solution_list type;
17 };
```

4. Merging

```
 1 template <typename SOLUTION_LIST_1, typename SOLUTION_LIST_2>
 2 struct merge_solution_lists {
 3   typedef typename merge_solution_lists <
 4      typename push_front_solution_to_list<
 5                  SOLUTION_LIST_1,
 6                  typename get_solution_list_head<SOLUTION_LIST_2>::type
 7               >::type,
 8      typename pop_front_solution_from_list<SOLUTION_LIST_2>::type
 9   >::type type;
10 };
```

4. Merging

```
 1 template <typename SOLUTION_LIST_1, typename SOLUTION_LIST_2>
 2 struct merge_solution_lists {
 3   typedef typename merge_solution_lists <
 4      typename push_front_solution_to_list<
 5                  SOLUTION_LIST_1,
 6                  typename get_solution_list_head<SOLUTION_LIST_2>::type
 7                >::type,
 8      typename pop_front_solution_from_list<SOLUTION_LIST_2>::type
 9   >::type type;
10 };
```

4. Merging

```
 1 template <typename SOLUTION_LIST_1, typename SOLUTION_LIST_2>
 2 struct merge_solution_lists {
 3   typedef typename merge_solution_lists <
 4      typename push_front_solution_to_list<
 5                  SOLUTION_LIST_1,
 6                  typename get_solution_list_head<SOLUTION_LIST_2>::type
 7               >::type,
 8      typename pop_front_solution_from_list<SOLUTION_LIST_2>::type
 9   >::type type;
10 };
```

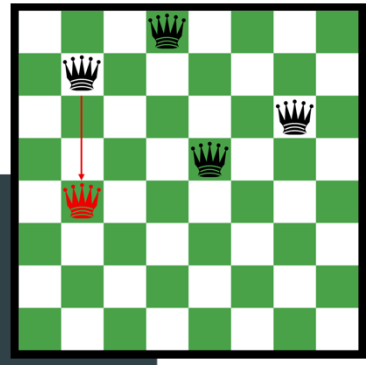
4. Merging

```
1 template <typename SOLUTION_LIST_1, typename SOLUTION_LIST_2>
2 struct merge_solution_lists {
3   typedef typename merge_solution_lists <
4      typename push_front_solution_to_list<
5                  SOLUTION_LIST_1,
6                  typename get_solution_list_head<SOLUTION_LIST_2>::type
7               >::type,
8      typename pop_front_solution_from_list<SOLUTION_LIST_2>::type
9   >::type type;
10 };
```

4. Merging

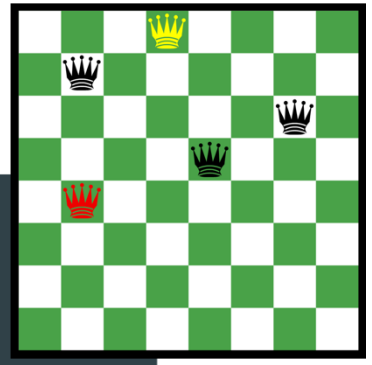
```
 1 template <typename SOLUTION_LIST_1, typename SOLUTION_LIST_2>
 2 struct merge_solution_lists {
 3   typedef typename merge_solution_lists <
 4      typename push_front_solution_to_list<
 5                  SOLUTION_LIST_1,
 6                  typename get_solution_list_head<SOLUTION_LIST_2>::type
 7               >::type,
 8      typename pop_front_solution_from_list<SOLUTION_LIST_2>::type
 9   >::type type;
10 };
```

5. Attack logic



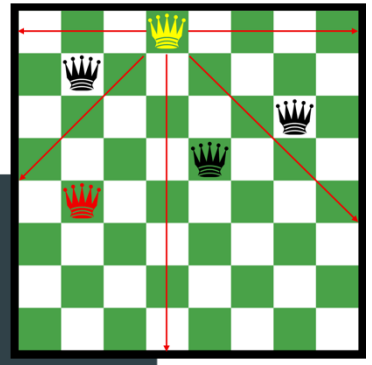
```
 1 template <typename PARTIAL_SOLUTION, typename POS>
 2 struct is_under_attack {
 3     typedef typename typeat<0, PARTIAL_SOLUTION>::type FirstQueen;
 4
 5     static const bool value =
 6          attacks<FirstQueen, POS>::value ||
 7          is_under_attack<typename pop_front<PARTIAL_SOLUTION>::type,
 8                          POS>::value;
 9 };
10
11 template<typename POS>
12 struct is_under_attack< solution<empty, empty, empty, empty,
13                                  empty, empty, empty, empty>,
14                         POS > {
15     static const bool value = false;
16 };
```

5. Attack logic



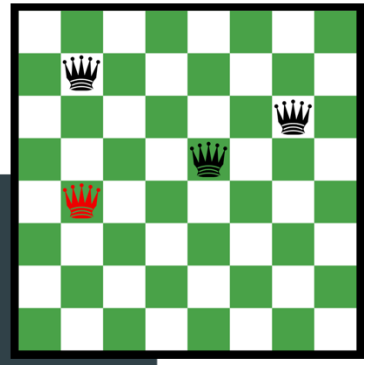
```
 1 template <typename PARTIAL_SOLUTION, typename POS>
 2 struct is_under_attack {
 3     typedef typename typeat<0, PARTIAL_SOLUTION>::type FirstQueen;
 4
 5     static const bool value =
 6          attacks<FirstQueen, POS>::value ||
 7          is_under_attack<typename pop_front<PARTIAL_SOLUTION>::type,
 8                          POS>::value;
 9 };
10
11 template<typename POS>
12 struct is_under_attack< solution<empty, empty, empty, empty,
13                                  empty, empty, empty, empty>,
14                         POS > {
15     static const bool value = false;
16 };
```

5. Attack logic



```
 1 template <typename PARTIAL_SOLUTION, typename POS>
 2 struct is_under_attack {
 3     typedef typename typeat<0, PARTIAL_SOLUTION>::type FirstQueen;
 4
 5     static const bool value =
 6          attacks<FirstQueen, POS>::value ||
 7          is_under_attack<typename pop_front<PARTIAL_SOLUTION>::type,
 8                          POS>::value;
 9 };
10
11 template<typename POS>
12 struct is_under_attack< solution<empty, empty, empty, empty,
13                                  empty, empty, empty, empty>,
14                         POS > {
15     static const bool value = false;
16 };
```

5. Attack logic



```
 1 template <typename PARTIAL_SOLUTION, typename POS>
 2 struct is_under_attack {
 3     typedef typename typeat<0, PARTIAL_SOLUTION>::type FirstQueen;
 4
 5     static const bool value =
 6          attacks<FirstQueen, POS>::value ||
 7          is_under_attack<typename pop_front<PARTIAL_SOLUTION>::type,
 8                          POS>::value;
 9 };
10
11 template<typename POS>
12 struct is_under_attack< solution<empty, empty, empty, empty,
13                                  empty, empty, empty, empty>,
14                         POS > {
15     static const bool value = false;
16 };
```

5. Attack logic

```
 1 template <typename Q, typename POS>
 2 struct attacks {
 3     static const int posV = POS::value;
 4
 5     static const bool value =
 6         Q::col == POS::col ||
 7         Q::row == POS::row ||
 8         diag_attack<Q::col, Q::row ,posV>::value;
 9 };
10
11 template<typename POS>
12 struct attacks<empty, POS> {
13     static const bool value = false;
14 };
```

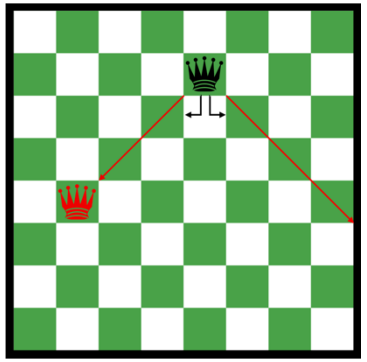
5. Attack logic

```
 1 template <typename Q, typename POS>
 2 struct attacks {
 3     static const int posV = POS::value;
 4
 5     static const bool value =
 6         Q::col == POS::col ||
 7         Q::row == POS::row ||
 8         diag_attack<Q::col, Q::row ,posV>::value;
 9 };
10
11 template<typename POS>
12 struct attacks<empty, POS> {
13     static const bool value = false;
14 };
```

5. Attack logic

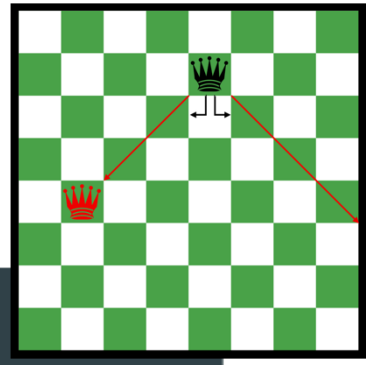
```
 1 template <typename Q, typename POS>
 2 struct attacks {
 3     static const int posV = POS::value;
 4
 5     static const bool value =
 6         Q::col == POS::col ||
 7         Q::row == POS::row ||
 8         diag_attack<Q::col, Q::row ,posV>::value;
 9 };
10
11 template<typename POS>
12 struct attacks<empty, POS> {
13     static const bool value = false;
14 };
```

5. Attack logic



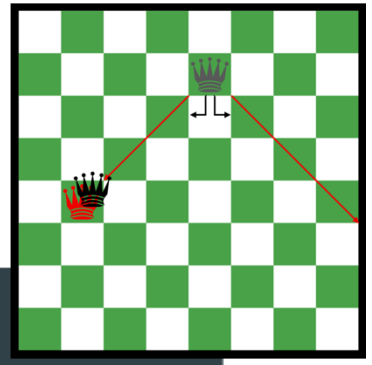
```
1 template <int QCOL, int QROW, int POS>
2 struct diag_attack {
3     static const bool value =
4       diag_attack_down_right<QCOL+1, QROW+10, POS>::value ||
5       diag_attack_down_left <QCOL-1, QROW+10, POS>::value;
6 };
```

5. Attack logic



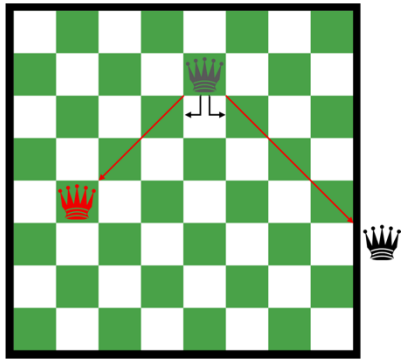
```
 1 template <int QCOL, int QROW, int POS>
 2 struct diag_attack_down_left {
 3     static const bool value = (QROW + QCOL == POS) ||
 4                     diag_attack_down_left<QCOL-1, QROW+10, POS>::value;
 5 };
 6
 7 template <int QCOL, int QROW, int POS>
 8 struct diag_attack_down_right {
 9     static const bool value = (QROW + QCOL == POS) ||
10                     diag_attack_down_right<QCOL+1, QROW+10, POS>::value;
11 };
```

5. Attack logic



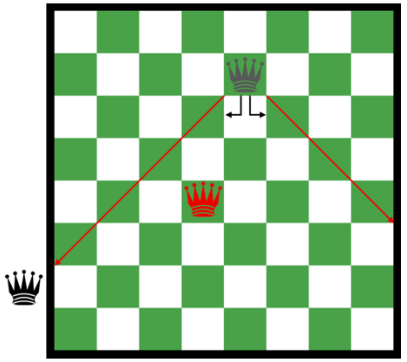
```
 1 template <int QCOL, int QROW, int POS>
 2 struct diag_attack_down_left {
 3     static const bool value = (QROW + QCOL == POS) ||
 4                     diag_attack_down_left<QCOL-1, QROW+10, POS>::value;
 5 };
 6
 7 template <int QCOL, int QROW, int POS>
 8 struct diag_attack_down_right {
 9     static const bool value = (QROW + QCOL == POS) ||
10                     diag_attack_down_right<QCOL+1, QROW+10, POS>::value;
11 };
```

5. Attack logic



```
 1 template <int QROW, int POS>
 2 struct diag_attack_down_right<9, QROW, POS> {
 3     static const bool value = false;
 4 };
 5
 6 template <int QCOL, int POS>
 7 struct diag_attack_down_right<QCOL, 90, POS> {
 8     static const bool value = false;
 9 };
10
11 template <int POS>
12 struct diag_attack_down_right<9, 90, POS> {
13     static const bool value = false;
14 };
```

5. Attack logic

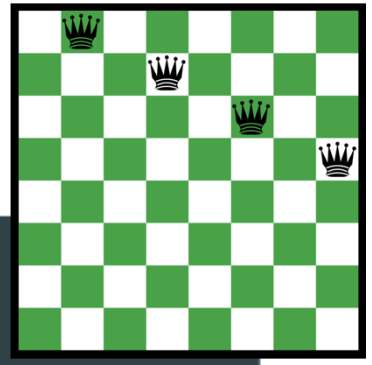


```
 1 template <int QROW, int POS>
 2 struct diag_attack_down_left<0, QROW, POS> {
 3     static const bool value = false;
 4 };
 5
 6 template <int QCOL, int POS>
 7 struct diag_attack_down_left<QCOL, 90, POS> {
 8     static const bool value = false;
 9 };
10
11 template <int POS>
12 struct diag_attack_down_left<0, 90, POS> {
13     static const bool value = false;
14 };
```

6. Backtracking

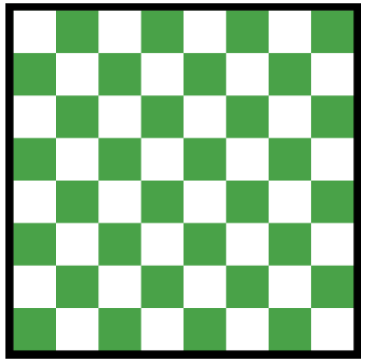
```
1 template<typename CURR_SOLUTION,
2           int CURR_ROW,
3           int CURR_COL,
4           bool CONTINUE>
5 struct ProcessSquare;
```

6. Backtracking



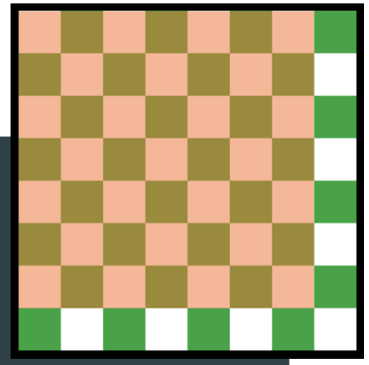
```
solution<
        SQUARE<12>,     // Queen in row 1, col 2
        SQUARE<24>,     // Queen in row 2, col 4
        SQUARE<36>,     // Queen in row 3, col 6
        SQUARE<48>,     // Queen in row 4, col 8
        empty,          // Row 5 (current) - not filled yet
        empty,          // Row 6 - not filled yet
        empty,          // Row 7 - not filled yet
        empty           // Row 8 - not filled yet
      >
```

6. Backtracking



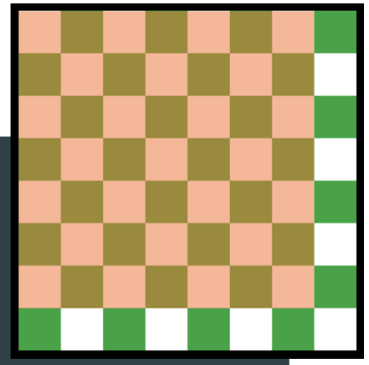
```
template<typename CURR_SOLUTION,
         int CURR_ROW,
         int CURR_COL>
struct ProcessSquare<
                     CURR_SOLUTION,
                     CURR_ROW,
                     CURR_COL,
                     false
                    > {
    typedef solution_list<> SolutionList;
};
```

6. Backtracking



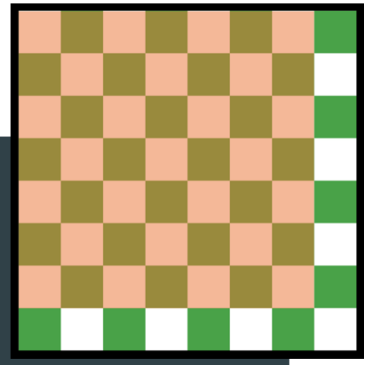
```
 1 template<typename CURR_SOLUTION, int CURR_ROW, int CURR_COL>
 2 struct ProcessSquare<CURR_SOLUTION,CURR_ROW, CURR_COL, true> {
 3     static const int pos = CURR_ROW + CURR_COL;
 4     static const int num_elems_in_curr_solution = (CURR_ROW-10)/10;
 5
 6     static const bool is_curr_solution_valid =
 7            first_N_elems_not_empty<CURR_SOLUTION, num_elems_in_curr_solution>::value;
 8
 9     typedef typename push_front_if<CURR_SOLUTION,
10                                    SQUARE(pos),
11                                    !is_under_attack<CURR_SOLUTION, SQUARE(pos)>::value
12                                    >::type next_curr_solution;
13
14     typedef typename ProcessSquare<next_curr_solution,
15                                    CURR_ROW+10, 1,
16                                    is_curr_solution_valid>::SolutionList
17             SolutionsIncludingCurrentPos;
```

6. Backtracking



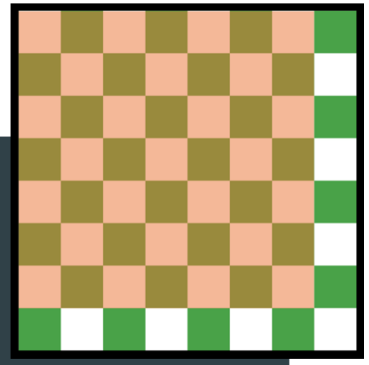
```
 1 template<typename CURR_SOLUTION, int CURR_ROW, int CURR_COL>
 2 struct ProcessSquare<CURR_SOLUTION,CURR_ROW, CURR_COL, true> {
 3     static const int pos = CURR_ROW + CURR_COL;
 4     static const int num_elems_in_curr_solution = (CURR_ROW-10)/10;
 5
 6     static const bool is_curr_solution_valid =
 7            first_N_elems_not_empty<CURR_SOLUTION, num_elems_in_curr_solution>::value;
 8
 9     typedef typename push_front_if<CURR_SOLUTION,
10                                    SQUARE(pos),
11                                    !is_under_attack<CURR_SOLUTION, SQUARE(pos)>::value
12                                    >::type next_curr_solution;
13
14     typedef typename ProcessSquare<next_curr_solution,
15                                    CURR_ROW+10, 1,
16                                    is_curr_solution_valid>::SolutionList
17             SolutionsIncludingCurrentPos;
```

6. Backtracking



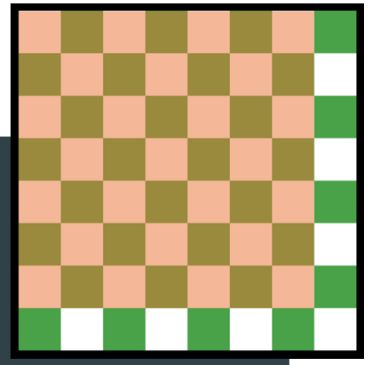
```
 1 template<typename CURR_SOLUTION, int CURR_ROW, int CURR_COL>
 2 struct ProcessSquare<CURR_SOLUTION,CURR_ROW, CURR_COL, true> {
 3     static const int pos = CURR_ROW + CURR_COL;
 4     static const int num_elems_in_curr_solution = (CURR_ROW-10)/10;
 5
 6     static const bool is_curr_solution_valid =
 7            first_N_elems_not_empty<CURR_SOLUTION, num_elems_in_curr_solution>::value;
 8
 9     typedef typename push_front_if<CURR_SOLUTION,
10                                    SQUARE(pos),
11                                    !is_under_attack<CURR_SOLUTION, SQUARE(pos)>::value
12                                    >::type next_curr_solution;
13
14     typedef typename ProcessSquare<next_curr_solution,
15                                    CURR_ROW+10, 1,
16                                    is_curr_solution_valid>::SolutionList
17             SolutionsIncludingCurrentPos;
```

6. Backtracking



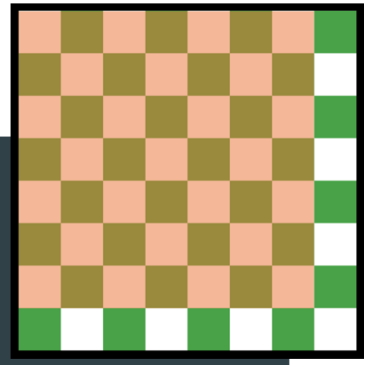
```
template<typename CURR_SOLUTION, int CURR_ROW, int CURR_COL>
struct ProcessSquare<CURR_SOLUTION,CURR_ROW, CURR_COL, true> {
    static const int pos = CURR_ROW + CURR_COL;
    static const int num_elems_in_curr_solution = (CURR_ROW-10)/10;

    static const bool is_curr_solution_valid =
           first_N_elems_not_empty<CURR_SOLUTION, num_elems_in_curr_solution>::value;

    typedef typename push_front_if<CURR_SOLUTION,
                                   SQUARE(pos),
                                   !is_under_attack<CURR_SOLUTION, SQUARE(pos)>::value
                                   >::type next_curr_solution;

    typedef typename ProcessSquare<next_curr_solution,
                                   CURR_ROW+10, 1,
                                   is_curr_solution_valid>::SolutionList
            SolutionsIncludingCurrentPos;
```

6. Backtracking



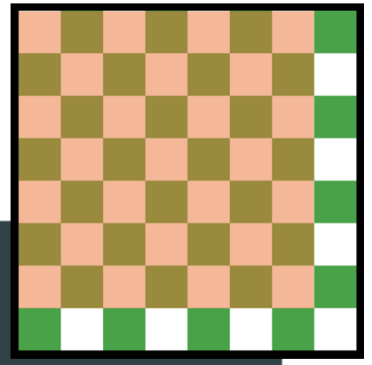
```
template<typename CURR_SOLUTION, int CURR_ROW, int CURR_COL>
struct ProcessSquare<CURR_SOLUTION,CURR_ROW, CURR_COL, true> {
    static const int pos = CURR_ROW + CURR_COL;
    static const int num_elems_in_curr_solution = (CURR_ROW-10)/10;

    static const bool is_curr_solution_valid =
           first_N_elems_not_empty<CURR_SOLUTION, num_elems_in_curr_solution>::value;

    typedef typename push_front_if<CURR_SOLUTION,
                                   SQUARE(pos),
                                   !is_under_attack<CURR_SOLUTION, SQUARE(pos)>::value
                                   >::type next_curr_solution;

    typedef typename ProcessSquare<next_curr_solution,
                                   CURR_ROW+10, 1,
                                   is_curr_solution_valid>::SolutionList
            SolutionsIncludingCurrentPos;
```

6. Backtracking



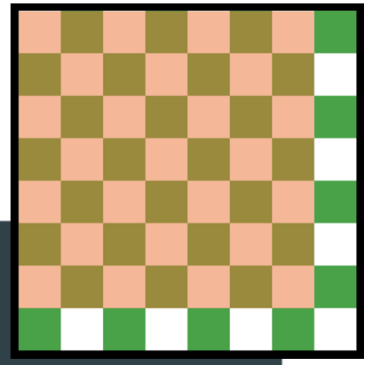
```
 1 template<typename CURR_SOLUTION, int CURR_ROW, int CURR_COL>
 2 struct ProcessSquare<CURR_SOLUTION,CURR_ROW, CURR_COL, true> {
 3     static const int pos = CURR_ROW + CURR_COL;
 4     static const int num_elems_in_curr_solution = (CURR_ROW-10)/10;
 5
 6     static const bool is_curr_solution_valid =
 7           first_N_elems_not_empty<CURR_SOLUTION, num_elems_in_curr_solution>::value;
 8
 9     typedef typename push_front_if<CURR_SOLUTION,
10                                    SQUARE(pos),
11                                    !is_under_attack<CURR_SOLUTION, SQUARE(pos)>::value
12                                    >::type next_curr_solution;
13
14     typedef typename ProcessSquare<next_curr_solution,
15                                    CURR_ROW+10, 1,
16                                    is_curr_solution_valid>::SolutionList
17           SolutionsIncludingCurrentPos;
```

6. Backtracking



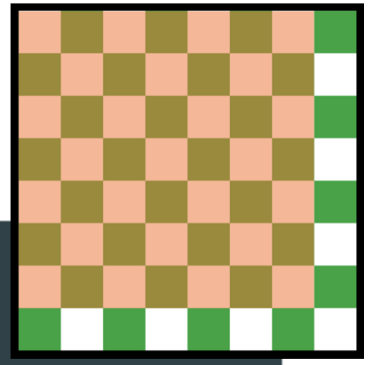
```
1     typedef ProcessSquare<CURR_SOLUTION,
2                           CURR_ROW,
3                           CURR_COL+1,
4                           is_curr_solution_valid> ProcessNextCol;
5
6     typedef typename
7         merge_solution_lists<
8                               SolutionsIncludingCurrentPos,
9                               typename ProcessNextCol::SolutionList
10                            >::type SolutionList;
11 };
```

6. Backtracking



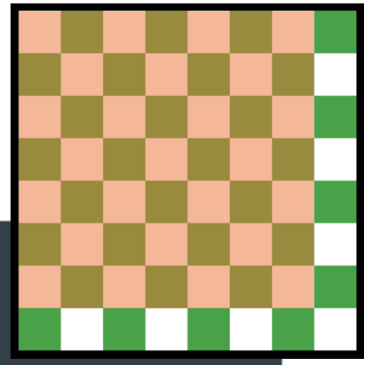
```
1     typedef ProcessSquare<CURR_SOLUTION,
2                           CURR_ROW,
3                           CURR_COL+1,
4                           is_curr_solution_valid> ProcessNextCol;
5
6     typedef typename
7         merge_solution_lists<
8                               SolutionsIncludingCurrentPos,
9                               typename ProcessNextCol::SolutionList
10                            >::type SolutionList;
11 };
```

6. Backtracking



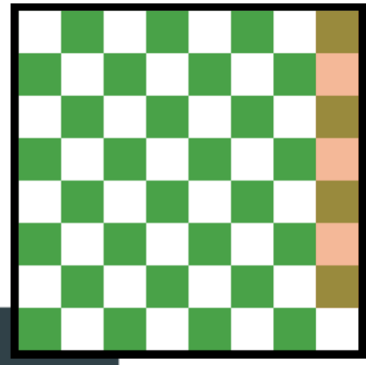
```
1     typedef ProcessSquare<CURR_SOLUTION,
2                           CURR_ROW,
3                           CURR_COL+1,
4                           is_curr_solution_valid> ProcessNextCol;
5
6     typedef typename
7         merge_solution_lists<
8                               SolutionsIncludingCurrentPos,
9                               typename ProcessNextCol::SolutionList
10                            >::type SolutionList;
11 };
```

6. Backtracking



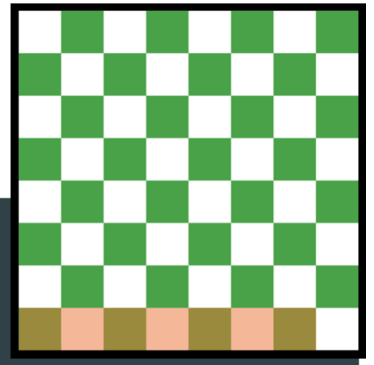
```
1     typedef ProcessSquare<CURR_SOLUTION,
2                           CURR_ROW,
3                           CURR_COL+1,
4                           is_curr_solution_valid> ProcessNextCol;
5
6     typedef typename
7         merge_solution_lists<
8                             SolutionsIncludingCurrentPos,
9                             typename ProcessNextCol::SolutionList
10                          >::type SolutionList;
11 };
```

6. Backtracking



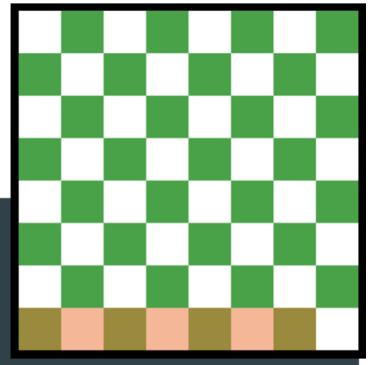
```
1 template<typename CURR_SOLUTION, int CURR_ROW>
2 struct ProcessSquare<
3                       CURR_SOLUTION,
4                       CURR_ROW, 8,
5                       true
6                     >
7
```

6. Backtracking



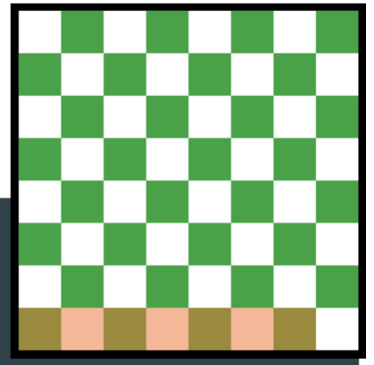
```
template<typename CURR_SOLUTION, int CURR_COL>
struct ProcessSquare<CURR_SOLUTION, 80, CURR_COL, true> {
    static const int pos = 80 + CURR_COL;

    typedef typename push_front_if< CURR_SOLUTION, SQUARE(pos),
                                    !is_under_attack<CURR_SOLUTION, SQUARE(pos)>::value
                                  >::type
            last_curr_solution;

    typedef typename
            push_front_solution_to_list_if <
                                              solution_list<>,
                                              last_curr_solution,
                                              first_N_elems_not_empty<last_curr_solution, 8>::value
                                            >::type
            SolutionListWithCurrentPosOrEmpty;
```

6. Backtracking



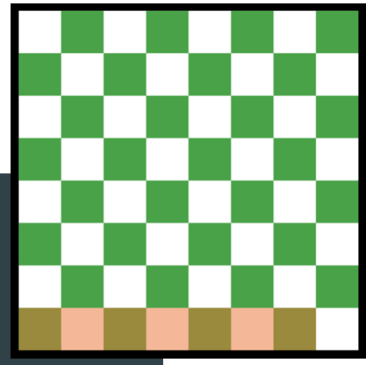
```
template<typename CURR_SOLUTION, int CURR_COL>
struct ProcessSquare<CURR_SOLUTION, 80, CURR_COL, true> {
    static const int pos = 80 + CURR_COL;

    typedef typename push_front_if< CURR_SOLUTION, SQUARE(pos),
                                    !is_under_attack<CURR_SOLUTION, SQUARE(pos)>::value
                                  >::type
            last_curr_solution;

    typedef typename
            push_front_solution_to_list_if <
                                              solution_list<>,
                                              last_curr_solution,
                                              first_N_elems_not_empty<last_curr_solution, 8>::value
                                            >::type
            SolutionListWithCurrentPosOrEmpty;
```

6. Backtracking



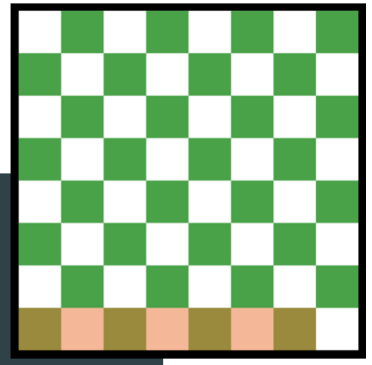
```
 1 template<typename CURR_SOLUTION, int CURR_COL>
 2 struct ProcessSquare<CURR_SOLUTION, 80, CURR_COL, true> {
 3     static const int pos = 80 + CURR_COL;
 4
 5     typedef typename push_front_if< CURR_SOLUTION, SQUARE(pos),
 6                                     !is_under_attack<CURR_SOLUTION, SQUARE(pos)>::value
 7                                   >::type
 8             last_curr_solution;
 9
10     typedef typename
11             push_front_solution_to_list_if <
12                                               solution_list<>,
13                                               last_curr_solution,
14                                               first_N_elems_not_empty<last_curr_solution, 8>::value
15                                             >::type
16             SolutionListWithCurrentPosOrEmpty;
```

6. Backtracking



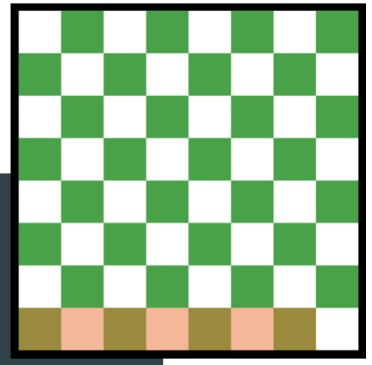
```
1     static const bool is_curr_solution_valid =
2               first_N_elems_not_empty<CURR_SOLUTION, 7>::value;
3
4     typedef ProcessSquare<
5                           CURR_SOLUTION,
6                           80, CURR_COL+1,
7                           is_curr_solution_valid
8                          >
9             ProcessNextCol;
10
11    typedef typename
12            merge_solution_lists<
13                                  SolutionListWithCurrentPosOrEmpty,
14                                  typename ProcessNextCol::SolutionList
15                                 >::type
16            SolutionList;
17 };
```

6. Backtracking



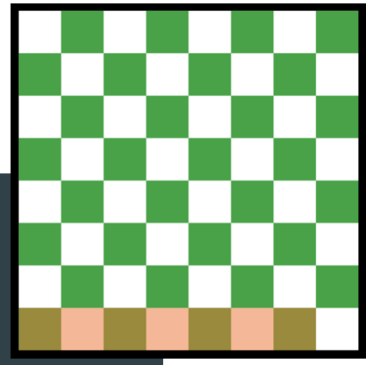
```
1     static const bool is_curr_solution_valid =
2               first_N_elems_not_empty<CURR_SOLUTION, 7>::value;
3
4     typedef ProcessSquare<
5                           CURR_SOLUTION,
6                           80, CURR_COL+1,
7                           is_curr_solution_valid
8                         >
9             ProcessNextCol;
10
11    typedef typename
12            merge_solution_lists<
13                                  SolutionListWithCurrentPosOrEmpty,
14                                  typename ProcessNextCol::SolutionList
15                                >::type
16            SolutionList;
17 };
```

6. Backtracking



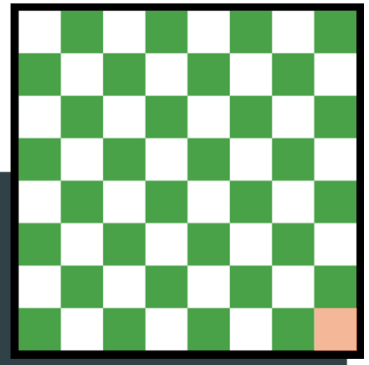
```
1     static const bool is_curr_solution_valid =
2               first_N_elems_not_empty<CURR_SOLUTION, 7>::value;
3
4     typedef ProcessSquare<
5                           CURR_SOLUTION,
6                           80, CURR_COL+1,
7                           is_curr_solution_valid
8                          >
9             ProcessNextCol;
10
11    typedef typename
12            merge_solution_lists<
13                                  SolutionListWithCurrentPosOrEmpty,
14                                  typename ProcessNextCol::SolutionList
15                                 >::type
16            SolutionList;
17 };
```

6. Backtracking



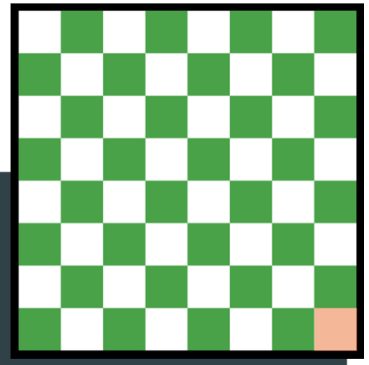
```
1     static const bool is_curr_solution_valid =
2               first_N_elems_not_empty<CURR_SOLUTION, 7>::value;
3
4     typedef ProcessSquare<
5                           CURR_SOLUTION,
6                           80, CURR_COL+1,
7                           is_curr_solution_valid
8                          >
9             ProcessNextCol;
10
11    typedef typename
12            merge_solution_lists<
13                                 SolutionListWithCurrentPosOrEmpty,
14                                 typename ProcessNextCol::SolutionList
15                                >::type
16            SolutionList;
17 };
```

6. Backtracking



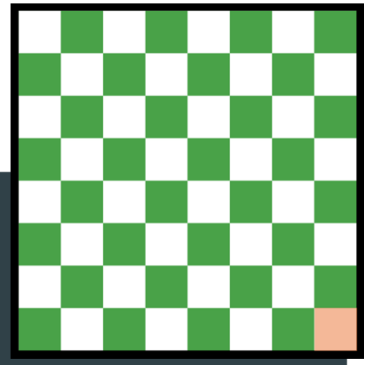
```
template<typename CURR_SOLUTION>
struct ProcessSquare<CURR_SOLUTION, 80, 8, true> {
    static const int pos = 88;
    typedef typename push_front_if<
                                    CURR_SOLUTION, SQUARE(pos),
                                    !is_under_attack<CURR_SOLUTION, SQUARE(pos)>::value
                                  >::type
            last_curr_solution;

    typedef typename
            push_front_solution_to_list_if<
                                             solution_list<>,
                                             last_curr_solution,
                                             first_N_elems_not_empty<last_curr_solution, 8>::value
                                           >::type
            SolutionList;
};
```

6. Backtracking



```
1 template<typename CURR_SOLUTION>
2 struct ProcessSquare<CURR_SOLUTION, 80, 8, true> {
3     static const int pos = 88;
4     typedef typename push_front_if<
5                                     CURR_SOLUTION, SQUARE(pos),
6                                     !is_under_attack<CURR_SOLUTION, SQUARE(pos)>::value
7                                   >::type
8             last_curr_solution;
9
10    typedef typename
11            push_front_solution_to_list_if<
12                                             solution_list<>,
13                                             last_curr_solution,
14                                             first_N_elems_not_empty<last_curr_solution, 8>::value
15                                           >::type
16            SolutionList;
17 };
```

6. Backtracking



```
template<typename CURR_SOLUTION>
struct ProcessSquare<CURR_SOLUTION, 80, 8, true> {
    static const int pos = 88;
    typedef typename push_front_if<
                                    CURR_SOLUTION, SQUARE(pos),
                                    !is_under_attack<CURR_SOLUTION, SQUARE(pos)>::value
                                  >::type
            last_curr_solution;

    typedef typename
            push_front_solution_to_list_if<
                                             solution_list<>,
                                             last_curr_solution,
                                             first_N_elems_not_empty<last_curr_solution, 8>::value
                                           >::type
            SolutionList;
};
```

7. Instantiation

```
1 struct Solve {
2     typedef ProcessSquare<
3                             solution<>,
4                             10, 1,
5                             true
6                         >::SolutionList
7             TheSolutions;
8 };
```

7. Instantiation

```
1 struct Solve {
2     typedef ProcessSquare<
3                           solution<>,
4                           10, 1,
5                           true
6                         >::SolutionList
7             TheSolutions;
8 };
```

7. Instantiation

```
1 struct Solve {
2     typedef ProcessSquare<
3                             solution<>,
4                             10, 1,
5                             true
6                         >::SolutionList
7             TheSolutions;
8 };
```

7. Instantiation

```
1 struct Solve {
2     typedef ProcessSquare<
3                             solution<>,
4                             10, 1,
5                             true
6                         >::SolutionList
7             TheSolutions;
8 };
```

7. Instantiation

```
1 struct Solve {
2     typedef ProcessSquare<
3                           solution<>,
4                           10, 1,
5                           true
6                         >::SolutionList
7             TheSolutions;
8 };
```

8. Run-time logic

```
1 int main(int argc, char* argv[])
2 {
3       std::cout<< "Solutions:\n";
4       Solve::TheSolutions s;
5       printAllSolutions(s);
6       return 0;
7 }
```

8. Run-time logic

```
 1 typedef solution<empty,empty,empty,empty,
 2                  empty,empty,empty,empty> EmptySolution;
 3
 4 void seqPrint(const EmptySolution&)
 5 {}
 6
 7 template<typename S>
 8 void seqPrint(const S&)
 9 {
10     typename typeat<0,S>::type front;
11     std::cout<< "[" << front << "]";
12
13     typename pop_front<S>::type tail;
14     seqPrint(tail);
15 }
```

8. Run-time logic

```
1 template<typename S>
2 void printSolution(const S& container) {
3     std::cout<< "Solution:";
4     seqPrint(container);
5     std::cout<<"\n";
6 }
7
8 void printSolution(const empty&){
9 }
```

8. Run-time logic

```
 1 void printAllSolutions(const empty&) {}
 2
 3 template<typename ALL_SOLUTIONS>
 4 void printAllSolutions(const ALL_SOLUTIONS&){
 5     typename get_solution_list_head<ALL_SOLUTIONS>::type aSolution;
 6     printSolution(aSolution);
 7
 8     typename pop_front_solution_from_list<ALL_SOLUTIONS>::type remainingSolutions;
 9     printAllSolutions(remainingSolutions);
10 }
```

Template Metaprogramming solution C++03

1. Representation
2. Manipulators
3. Solution lists
4. Merging
5. Attack logic
6. Backtracking
7. Instantiation
8. Run-time logic

Bloomberg

Engineering

Building TMP solution

```
8 Queens - C++03
gmake[3]: Entering directory '/home9/mmarcell/src_personal/eightqueens/build'
[ 50%] Building CXX object src/c++_03/CMakeFiles/c++_03_EightQueens.dir/EightQueens.cpp.o
cd /home/mmarcell/src_personal/eightqueens/build/src/c++_03 && /opt/bb/bin/c++   -O0  -std=gnu++98 -Wall -MD -MT src/c++_03/CMakeFile
s/c++_03_EightQueens.dir/EightQueens.cpp.o -MF CMakeFiles/c++_03_EightQueens.dir/EightQueens.cpp.o.d -o CMakeFiles/c++_03_EightQueens
.dir/EightQueens.cpp.o -c /home/mmarcell/src_personal/eightqueens/src/c++_03/EightQueens.cpp
[100%] Linking CXX executable c++_03_EightQueens
cd /home/mmarcell/src_personal/eightqueens/build/src/c++_03 && /opt/bb/bin/cmake -E cmake_link_script CMakeFiles/c++_03_EightQueens.d
ir/link.txt --verbose=1
/opt/bb/bin/c++ -O0  CMakeFiles/c++_03_EightQueens.dir/EightQueens.cpp.o -o c++_03_EightQueens
gmake[3]: Leaving directory '/home9/mmarcell/src_personal/eightqueens/build'
[100%] Built target c++_03_EightQueens
gmake[2]: Leaving directory '/home9/mmarcell/src_personal/eightqueens/build'
/opt/bb/bin/cmake -E cmake_progress_start /home/mmarcell/src_personal/eightqueens/build/CMakeFiles 0
gmake[1]: Leaving directory '/home9/mmarcell/src_personal/eightqueens/build'


real    0m5.319s
user    0m4.898s
sys     0m0.350s
```

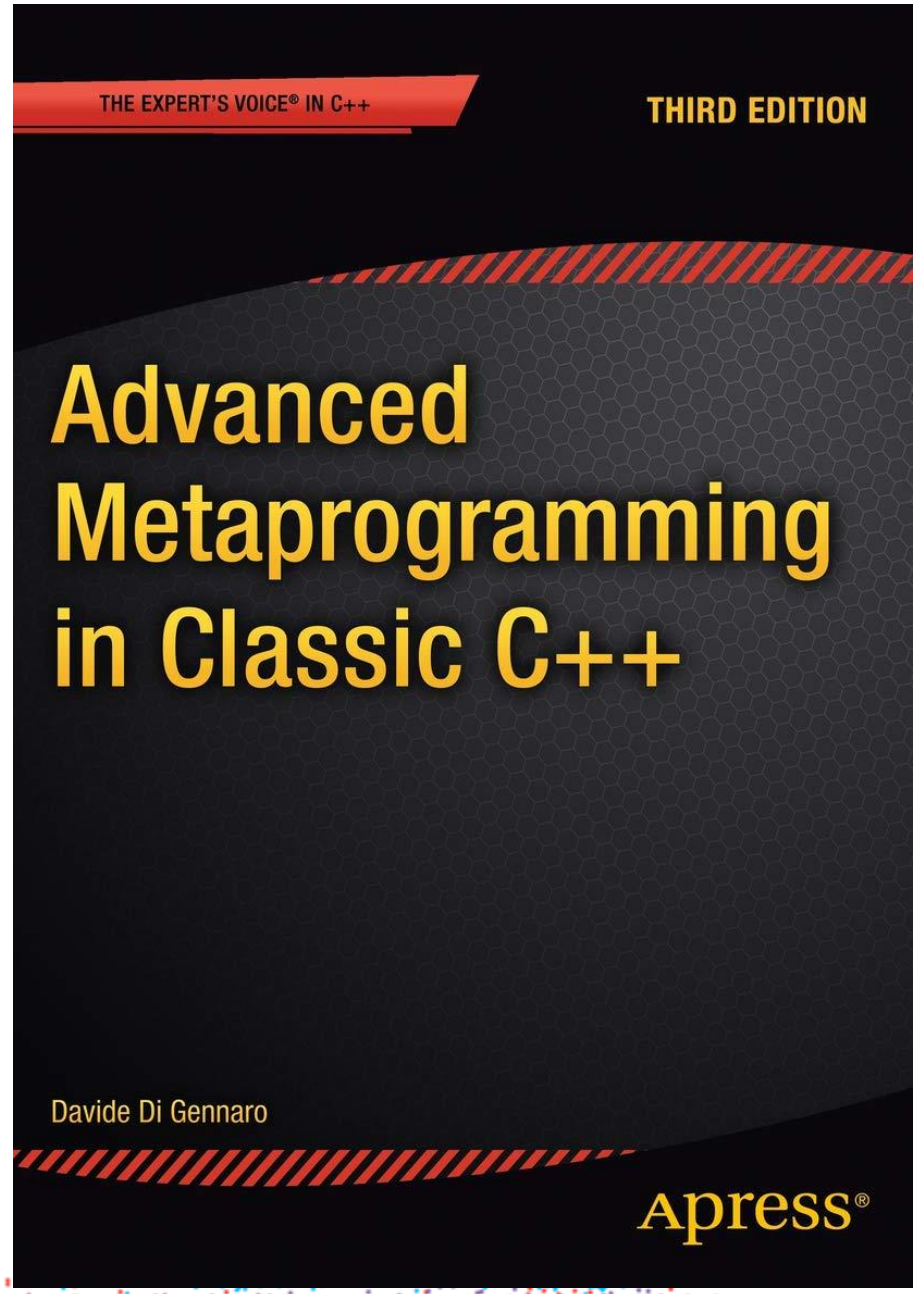
Running TMP solution

8 Queens - C++03

```
mmarcell@rdev-ob-628 eightqueens (main)$ build/src/c++_03/c++_03_EightQueens
Solutions:
Solution:[85][77][61][53][48][36][24][12]
Solution:[85][73][68][54][47][31][26][12]
Solution:[85][77][64][51][43][38][26][12]
Solution:[84][77][65][53][41][36][28][12]
Solution:[83][76][64][51][48][35][27][12]
Solution:[84][71][65][58][46][33][27][12]
Solution:[83][76][68][51][44][37][25][12]
Solution:[84][76][68][53][41][37][25][12]
Solution:[86][73][67][52][48][35][21][14]
Solution:[82][77][63][56][48][35][21][14]
Solution:[87][73][68][52][45][31][26][14]
Solution:[82][75][67][51][43][38][26][14]
Solution:[85][73][61][57][42][38][26][14]
Solution:[85][77][62][56][43][31][28][14]
Solution:[83][76][62][57][45][31][28][14]
Solution:[86][72][67][51][43][35][28][14]
Solution:[86][71][65][52][48][33][27][14]
Solution:[82][78][66][51][43][35][27][14]
Solution:[88][73][61][56][42][35][27][14]
Solution:[83][76][62][55][48][31][27][14]
Solution:[85][71][68][56][43][37][22][14]
Solution:[81][75][68][56][43][37][22][14]
```

8 Queens - C++03

```
Solution:[83][76][68][51][45][37][22][14]
Solution:[87][75][63][51][46][38][22][14]
Solution:[86][73][61][57][45][38][22][14]
Solution:[87][73][61][56][48][35][22][14]
Solution:[84][77][63][58][42][35][21][16]
Solution:[84][72][68][55][47][31][23][16]
Solution:[87][74][62][55][48][31][23][16]
Solution:[85][77][62][54][48][31][23][16]
Solution:[85][71][68][54][42][37][23][16]
Solution:[84][71][65][58][42][37][23][16]
Solution:[85][72][68][51][44][37][23][16]
Solution:[87][72][64][51][48][35][23][16]
Solution:[88][72][64][51][47][35][23][16]
Solution:[83][75][67][51][44][32][28][16]
Solution:[83][71][67][55][48][32][24][16]
Solution:[83][75][62][58][41][37][24][16]
Solution:[88][72][65][53][41][37][24][16]
Solution:[83][77][62][58][45][31][24][16]
Solution:[83][75][68][54][41][37][22][16]
Solution:[84][78][65][53][41][37][22][16]
Solution:[86][73][65][57][41][34][22][18]
Solution:[86][74][67][51][43][35][22][18]
Solution:[85][77][62][56][43][31][24][18]
Solution:[84][77][65][52][46][31][23][18]
```



TechAtBloomberg.com

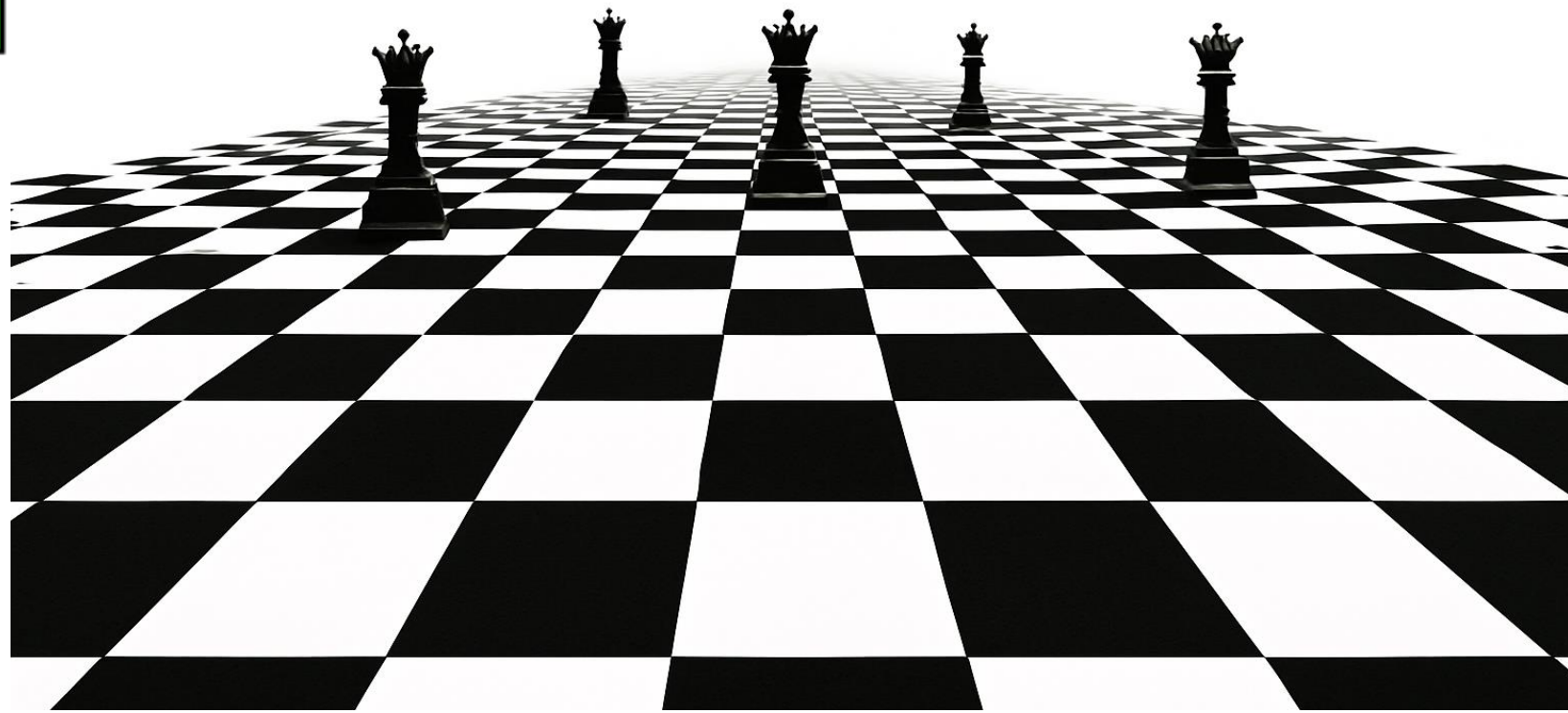
Bloomberg
Engineering

Source of generated image on this slide: ChatGPT

Source of generated image on this slide: ChatGPT

The N Queens problem

- Variadic template metaprogramming?
- constexpr approach ✅



TechAtBloomberg.com

Bloomberg

Engineering

What is a constant expression?

- An expression whose value can be computed at compile time
  - Only constexpr function calls ✅
  - Reinterpret cast, undefined behaviour, goto, assembly, … ❌
  - Thrown exceptions ❌ (since C++26: unless caught within evaluation ✅)
- Since C++20:
  - constexpr virtual functions ✅
  - new/delete ✅ (if all allocations are deallocated within evaluation)
- Only objects of literal type can be created within constant expressions
  - constexpr constructor & trivial destructor (or constexpr destructor since C++20)
  - Other restrictions: no virtual base classes ❌

Positions

```
struct Position {
    int d_row;
    int d_col;

    constexpr bool is_on_last_row(size_t size) const {
        return d_row == size - 1;
    }

    constexpr bool is_on_last_column(size_t size) const {
        return d_col == size - 1;
    }

    constexpr Position get_start_of_next_row() const {
        return Position{.d_row = d_row+1, .d_col = 0};
    }

    constexpr Position get_next_col() const {
        return Position{.d_row = d_row, .d_col = d_col+1};
    }

    friend std::ostream& operator<<(std::ostream& os, Position const& pos) {
        os << '(' << pos.d_row << ", " << pos.d_col << ')';
        return os;
    }
};
```

Queens

```
1  struct Queen {
2      Position d_position;
3
4      constexpr bool can_take(Position const& other_position) const {
5          return d_position.d_row == other_position.d_row
6              || d_position.d_col == other_position.d_col
7              || d_position.d_col - other_position.d_col == d_position.d_row - other_position.d_row
8              || d_position.d_col - other_position.d_col == -1*(d_position.d_row - other_position.d_row);
9      }
10
11     friend std::ostream& operator<<(std::ostream& os, Queen const& queen) {
12         os << queen.d_position;
13         return os;
14     }
15 };
```

std::abs

Compile time assertions

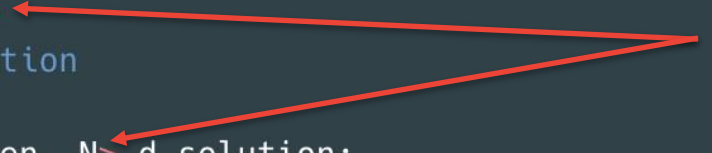
```
 1 static_assert(Queen{{1,1}}.can_take({1, 2}));
 2 static_assert(Queen{{1,1}}.can_take({2, 2}));
 3 static_assert(Queen{{1,1}}.can_take({2, 1}));
 4 static_assert(Queen{{1,1}}.can_take({2, 0}));
 5 static_assert(Queen{{1,1}}.can_take({1, 0}));
 6 static_assert(Queen{{1,1}}.can_take({0, 0}));
 7 static_assert(Queen{{1,1}}.can_take({0, 1}));
 8 static_assert(Queen{{1,1}}.can_take({0, 2}));
 9 static_assert(Queen{{1,1}}.can_take({2, 2}));
10
11 static_assert(!Queen{{1,1}}.can_take({3, 0}));
12 static_assert(!Queen{{1,1}}.can_take({3, 2}));
13 static_assert(!Queen{{1,1}}.can_take({2, 3}));
14 static_assert(!Queen{{1,1}}.can_take({0, 3}));
15
```

Partial Solution

```cpp
template<size_t N>
struct PartialSolution
{
    std::array<Queen, N> d_solution;
    size_t d_number_pieces;

    constexpr PartialSolution create_updated(Position const& new_pos) const {
        PartialSolution updated_partial_solution = *this;
        ++updated_partial_solution.d_number_pieces;
        updated_partial_solution.d_solution[d_number_pieces] = Queen{new_pos};
        return updated_partial_solution;
    }

    constexpr auto begin() const {
        return d_solution.begin();
    }

    constexpr auto end() const {
        return d_solution.begin() + d_number_pieces;
    }
};
```

Partial Solution

```cpp
template<size_t N>
struct PartialSolution
{
    std::array<Queen, N> d_solution;
    size_t d_number_pieces;

    constexpr PartialSolution create_updated(Position const& new_pos) const {
        PartialSolution updated_partial_solution = *this;
        ++updated_partial_solution.d_number_pieces;
        updated_partial_solution.d_solution[d_number_pieces] = Queen{new_pos};
        return updated_partial_solution;
    }

    constexpr auto begin() const {
        return d_solution.begin();
    }

    constexpr auto end() const {
        return d_solution.begin() + d_number_pieces;
    }
};
```



Solution Concept

```cpp
template<typename T>
concept Solution = std::ranges::random_access_range<T>
                && std::same_as<std::ranges::range_value_t<T>, Queen>;

template<size_t N>
using ArraySolution = std::array<Queen, N>;

using VectorSolution = std::vector<Queen>;

static_assert(Solution<ArraySolution<3>>);
static_assert(Solution<VectorSolution>);
```

Partial Solution

concept Solution

std::vector

std::array

```
 1 template<Solution S>
 2 struct PartialSolution
 3 {
 4     S d_solution;
 5     size_t d_number_pieces;
 6
 7     PartialSolution(size_t board_size)
 8         requires std::is_same_v<S, VectorSolution>
 9         : d_solution{}
10         , d_number_pieces{0} {
11         d_solution.resize(board_size);
12     }
13
14     constexpr PartialSolution()
15         requires (!std::is_same_v<S, VectorSolution>) = default;
16     // rest...
```

Build Solutions

```
1 template<Solution S>
2 constexpr void build_solution(
3     std::vector<S>& solution_set,
4     size_t board_size,
5     PartialSolution<S> current_partial_solution,
6     Position position_under_evaluation)
7 {
8     //...
```

Build Solutions

```
1 template<Solution S>
2 constexpr void build_solution(
3     std::vector<S>& solution_set, size_t board_size, PartialSolution<S> current_partial_solution, Position
  position_under_evaluation) {
4     if (std::ranges::none_of(current_partial_solution, [&](Queen const& existing_queen) {
5         return existing_queen.can_take(position_under_evaluation);
6     })) {
7         auto next_partial_solution = current_partial_solution.create_updated(position_under_evaluation);
8 
9         if (position_under_evaluation.is_on_last_row(board_size)) {
10            solution_set.push_back(next_partial_solution.d_solution);
11        }
12        else {
13            build_solution(
14                solution_set, board_size, next_partial_solution, position_under_evaluation.get_start_of_next_row());
15        }
16    }
17    if (!position_under_evaluation.is_on_last_column(board_size)) {
18        build_solution(
19            solution_set, board_size, current_partial_solution, position_under_evaluation.get_next_col());
20    }
21 }
```

Build Solutions

```cpp
template<size_t N, typename S = ArraySolution<N>>
constexpr std::vector<S> find_n_queens_solutions() {
    std::vector<S> solution_set{};
    build_solution<S>(solution_set, N, PartialSolution<S>{}, Position{.d_row=0, .d_col=0});
    return solution_set;
}

std::vector<VectorSolution> find_n_queens_solutions(size_t board_size) {
    std::vector<VectorSolution> solution_set{};
    build_solution<VectorSolution>(
        solution_set,
        board_size,
        PartialSolution<VectorSolution>{board_size},
        Position{.d_row=0, .d_col=0});
    return solution_set;
}
```

Done?

```
1 static_assert(find_n_queens_solutions<8>().size() == 92);
2 static_assert(find_n_queens_solutions<9>().size() == 352);
```

Done?

```
1 int main() {
2     static constexpr auto eight_queens_solutions = find_n_queens_solutions<8>();
```

gcc

```
1 [build] /include/c++/13/bits/allocator.h: In function ‘int main()’:
2 [build] /include/c++/13/bits/allocator.h:195:52: error: ‘find_n_queens_solutions() [with long unsigned int size
  = 8; Sol = std::array<Queen, 8>]()’ is not a constant expression because it refers to a result of ‘operator new’
3 [build]   195 |             return static_cast<_Tp*>(::operator new(__n));
```

clang

```
1 ConstexprEightQueens.cpp:312:27: error: constexpr variable 'eight_queens_solutions' must be initialized by a constant expression
2   312 |     static constexpr auto eight_queens_solutions = find_n_queens_solutions<8>();
3       |                           ^                        ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
4 ConstexprEightQueens.cpp:312:27: note: pointer to subobject of heap-allocated object is not a constant expression
5 /include/c++/v1/__memory/allocator.h:103:32: note: heap allocation performed here
6   103 |       return static_cast<_Tp*>(::operator new(__n * sizeof(_Tp)))
```

msvc

```
1 <source>(312): error C2131: expression did not evaluate to a constant
2 <source>(312): note: (sub-)object points to memory which was heap allocated during constant evaluation
```

Vector to array

```cpp
template<size_t N, typename T>
constexpr std::array<T, N> vec_to_array(std::vector<T> const& vec) {
    std::array<T, N> out;
    std::copy(vec.begin(), vec.end(), out.begin());
    return out;
}

template<size_t N>
constexpr auto find_n_queens_solutions_array() {
    return vec_to_array<find_n_queens_solutions<N>().size()>(find_n_queens_solutions<N>());
}
```

Vector to array

```
1 template<size_t N>
2 constexpr auto find_n_queens_solutions_array() {
3     auto res = find_n_queens_solutions<N>();
4     return vec_to_array<res.size()>(res);
5 }
```

```
1 [build] /workarea/eightqueens/src/constexpr/ConstexprEightQueens.cpp:255:36:
  error: the value of 'res' is not usable in a constant expression
2 [build]   255 |     return vec_to_array<res.size()>(res);
3 [build]       |                        ~~~~~~~~~~~~~~~~~~~~~~~^~~~
4 [build] /workarea/eightqueens/src/constexpr/ConstexprEightQueens.cpp:254:10:
  note: 'res' was not declared 'constexpr'
5 [build]   254 |     auto res = find_n_queens_solutions<N>();
6 [build]       |
```

C++26: std::define_static_array

```
1   template <ranges::input_range R>
2   consteval auto define_static_array(R&& r) -> span<ranges::range_value_t<R> const>;
```

```
1 template<size_t N>
2 consteval auto find_n_queens_solutions_array() {
3     // return vec_to_array<find_n_queens_solutions<N>().size()>(find_n_queens_solutions<N>());
4     return std::define_static_array(find_n_queens_solutions<N>());
5 }
```

wg21.link/P3491R3

Source of generated image on this slide: ChatGPT

Done ✅

```
1     static constexpr auto eight_queens_solutions = find_n_queens_solutions_array<8>();
2     print_solution(eight_queens_solutions);
```

Building a lookup table for board sizes

```
 1 template<size_t N>
 2 constexpr auto build_n_queens_lookup_table()
 3 {
 4     return []<size_t... Indices>(std::index_sequence<Indices...>)
 5         {
 6             return std::tuple{find_n_queens_solutions_array<Indices+1>()... };
 7         }(std::make_index_sequence<N>{});
 8     // return std::tuple{
 9     //     find_n_queens_solutions_array<1>(),
10     //     find_n_queens_solutions_array<2>(),
11     //     ... ,
12     //     find_n_queens_solutions_array<N>()};
13 }
```

std::index_sequence<0, 1, 2, …, N-1>

```cpp
constexpr int lookup_size = 10;
static constexpr auto lookup_table = build_n_queens_lookup_table<lookup_size>();
std::cout << "Number of solutions for each board size:\n";

[]<size_t... I>(std::index_sequence<I...>) {
    ((std::cout << (I + 1)
                << " Queens Problem solutions count: "
                << std::get<I>(lookup_table).size() << '\n'), ...);
}(std::make_index_sequence<lookup_size>{});
```

std::index_sequence<0, 1, 2, …, 9>

```
Number of solutions for each board size:
1 Queens Problem solutions count: 1
2 Queens Problem solutions count: 0
3 Queens Problem solutions count: 0
4 Queens Problem solutions count: 2
5 Queens Problem solutions count: 10
6 Queens Problem solutions count: 4
7 Queens Problem solutions count: 40
8 Queens Problem solutions count: 92
9 Queens Problem solutions count: 352
10 Queens Problem solutions count: 724
```

Accessing the lookup table

```
 1 [board_size]<size_t... I>(std::index_sequence<I...>) {
 2     (void)((board_size == I+1
 3             ? (print_solution(std::get<I>(lookup_table)), true)
 4             : false) || ...);
 5 }(std::make_index_sequence<lookup_size>{});
 6
 7 //    (board_size == 1 ? (print_solution(std::get<0>(lookup_table)), true) : false)
 8 // || (board_size == 2 ? (print_solution(std::get<1>(lookup_table)), true) : false)
 9 // || ...
10 // || (board_size == N ? (print_solution(std::get<N-1>(lookup_table)), true) : false)
```

Compiling Quirks

gcc

```
1 [build] /workarea/eightqueens/src/constexpr/ConstexprEightQueens.cpp:201:23:   in ‘constexpr’ expansion of
  ‘build_solution<std::array<Queen, 9> >((* & solution_set), board_size, current_partial_solution,
  position_under_evaluation.Position::get_next_col())’
2 [build] /workarea/eightqueens/src/constexpr/ConstexprEightQueens.cpp:253:61: error: ‘constexpr’ evaluation operation
  count exceeds limit of 33554432 (use ‘-fconstexpr-ops-limit=’ to increase the limit)
3 [build]   253 |     return vec_to_array<find_n_queens_solutions<N>().size()>(find_n_queens_solutions<N>());
4 [build]       |            ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~~~~~~~~~
```

clang

```
1 /Users/jmarriott13/code/eightqueens/src/constexpr/ConstexprEightQueens.cpp:240:15: error: static assertion expression
  is not an integral constant expression
2   240 | static_assert(find_n_queens_solutions<9>().size() == 352);
3       |               ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
4 /Library/Developer/CommandLineTools/SDKs/MacOSX26.0.sdk/usr/include/c++/v1/array:220:94: note: constexpr evaluation
  hit maximum step limit; possible infinite loop?
5   220 |   _LIBCPP_HIDE_FROM_ABI _LIBCPP_CONSTEXPR_SINCE_CXX17 const_iterator begin() const _NOEXCEPT {
```

msvc

```
1 <source>(240): error C2131: expression did not evaluate to a constant
2 <source>(144):  note: failure was caused by evaluation exceeding step limit of 1048576 (/constexpr:steps<NUMBER>)
```

Compiling Quirks

```
1 add_executable(constexprEightQueens ConstexprEightQueens.cpp)
2
3 if (CMAKE_CXX_COMPILER_ID MATCHES "Clang")
4     target_compile_options(constexprEightQueens PRIVATE -fconstexpr-steps=100000000)
5 elseif (CMAKE_CXX_COMPILER_ID STREQUAL "GNU")
6     target_compile_options(constexprEightQueens PRIVATE -fconstexpr-ops-limit=1000000000)
7 endif()
```

Compiling Quirks

~~std::ranges::none_of~~
for loop ✅

```cpp
constexpr bool piece_can_be_placed_at(Position const& position_under_evaluation,
                                      Solution auto const& current_partial_solution) {
    for (auto const& existing_queen: current_partial_solution) {
        if (existing_queen.can_take(position_under_evaluation))
            return false;
    }
    return true;
}

template<Solution Sol>
constexpr void build_solution(
    PartialSolution<Sol> current_partial_solution, Position position_under_evaluation, /*other args*/)
    {
        // if (std::ranges::none_of(current_partial_solution, [&](Queen const& existing_queen) {
        //     return existing_queen.can_take(position_under_evaluation); }))
        if (piece_can_be_placed_at(position_under_evaluation, current_partial_solution))
        { // rest...
```

```
> time cmake --build /build --config RelWithDebInfo --clean-first
  --target constexprEightQueens
[1/1] Cleaning all built files...
Cleaning... 0 files.
[2/2] Linking CXX executable src/constexpr/constexprEightQueens

real    2m28.997s
user    2m11.935s
sys     0m14.090s
```

std::ranges::none_of

```
> time cmake --build /build --config RelWithDebInfo --clean-first
  --target constexprEightQueens
[1/1] Cleaning all built files...
Cleaning... 0 files.
[2/2] Linking CXX executable src/constexpr/constexprEightQueens

real    0m57.195s
user    0m53.185s
sys     0m3.415s
```

piece_can_be_placed_at

~60% decrease in compile time

Compile Time (clang)

Board sizes: 1 to 8
~5 seconds

```
❯ time cmake --build build --target constexprEightQueens
[ 50%] Building CXX object src/constexpr/CMakeFiles/constexprEightQueens.dir/ConstexprEightQueens.cpp.o
[100%] Linking CXX executable constexprEightQueens
[100%] Built target constexprEightQueens
cmake --build build --target constexprEightQueens  4.72s user 0.18s system 96% cpu 5.084 total
```

Board sizes: 1 to 9
~7 seconds

```
❯ time cmake --build build --target constexprEightQueens
[ 50%] Building CXX object src/constexpr/CMakeFiles/constexprEightQueens.dir/ConstexprEightQueens.cpp.o
[100%] Linking CXX executable constexprEightQueens
[100%] Built target constexprEightQueens
cmake --build build --target constexprEightQueens  6.69s user 0.17s system 97% cpu 7.010 total
```

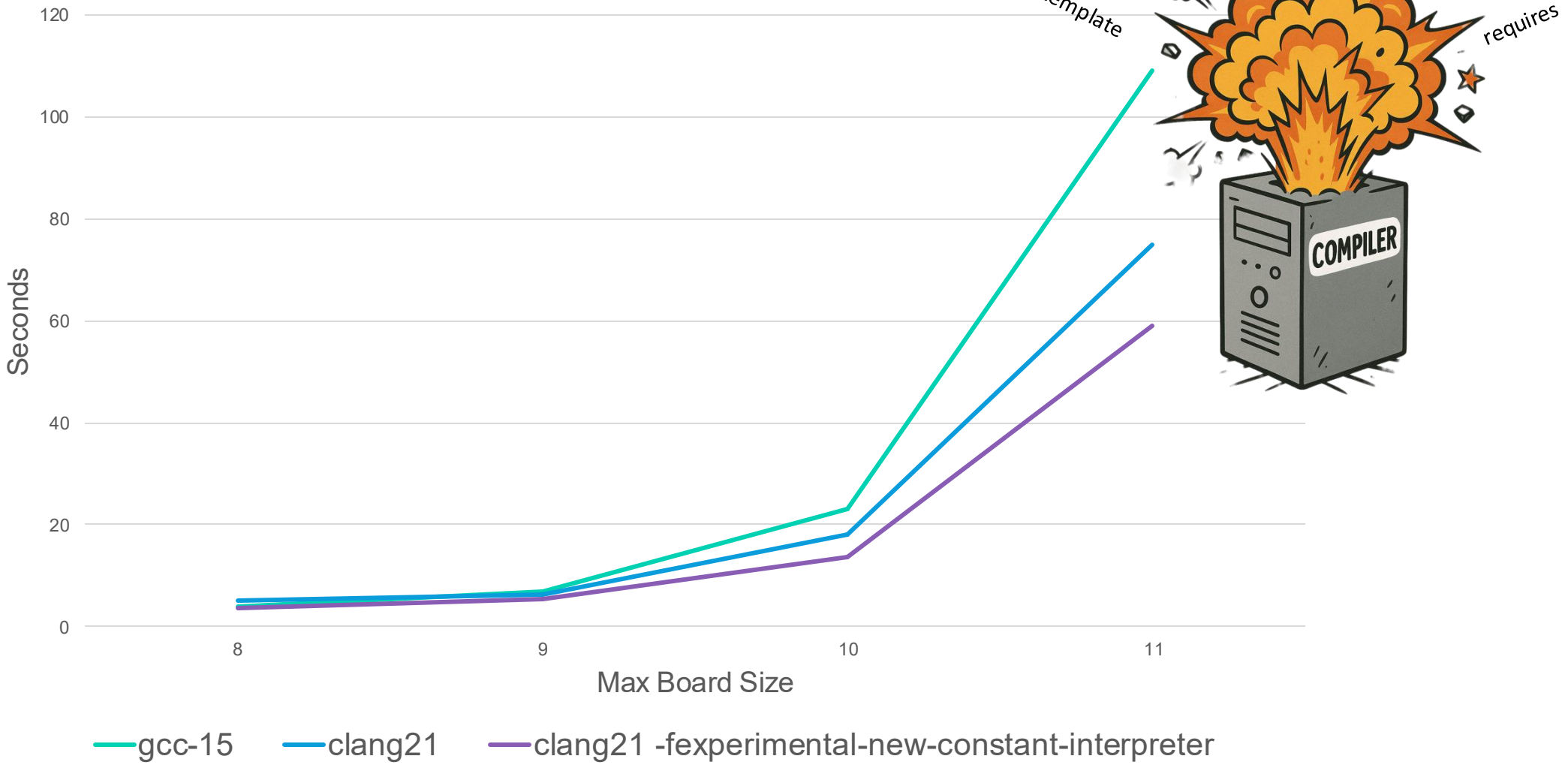
Board sizes: 1 to 10
~18 seconds

```
❯ time cmake --build build --target constexprEightQueens
[ 50%] Building CXX object src/constexpr/CMakeFiles/constexprEightQueens.dir/ConstexprEightQueens.cpp.o
[100%] Linking CXX executable constexprEightQueens
[100%] Built target constexprEightQueens
cmake --build build --target constexprEightQueens  17.71s user 0.18s system 99% cpu 18.028 total
```

Board sizes: 1 to 11
~79 seconds

```
❯ time cmake --build build --target constexprEightQueens
[ 50%] Building CXX object src/constexpr/CMakeFiles/constexprEightQueens.dir/ConstexprEightQueens.cpp.o
[100%] Linking CXX executable constexprEightQueens
[100%] Built target constexprEightQueens
cmake --build build --target constexprEightQueens  79.39s user 0.30s system 99% cpu 1:19.94 total
```

Compile Time



Source of generated image on this slide: ChatGPT

Demo

```
❯ ./build/src/constexpr/constexprEightQueens
Number of solutions for each board size:
1 Queens Problem solutions count: 1
2 Queens Problem solutions count: 0
3 Queens Problem solutions count: 0
4 Queens Problem solutions count: 2
5 Queens Problem solutions count: 10
6 Queens Problem solutions count: 4
7 Queens Problem solutions count: 40
8 Queens Problem solutions count: 92
9 Queens Problem solutions count: 352
10 Queens Problem solutions count: 724

Interactive mode: View solutions from the lookup table or runtime

Enter a positive board size (1-10) to see its solutions from the compile-time generated lookup table
Enter a negative board size (-1 and below) to see its solutions computed at runtime
Enter 0 to exit
Enter a value:
```

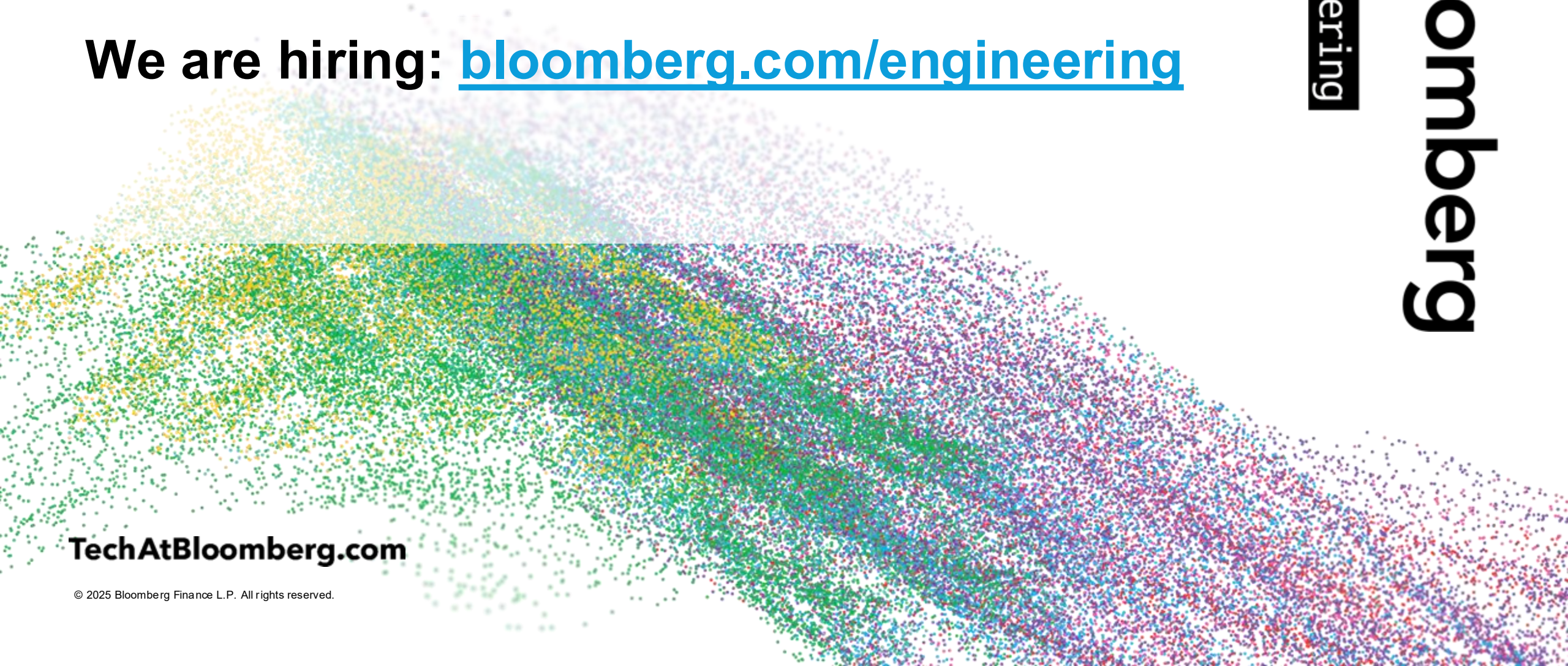
Source of generated image on this slide: ChatGPT

Thank you!

We are hiring: bloomberg.com/engineering

Bloomberg

Engineering



TechAtBloomberg.com