Оптимизационные задачи. Часто человечество занимается тем, что берет перебор, и пытается сделать этот перебор оптимально возможным, чтобы работало ну очень-очень быстро. Для примера подобной задачи часто решают задачу коммивояжера (TSP), которая не имеет решения лучше, чем за экспоненту, но при этом современными методами решается для $n \sim 200$. Задача, между тем, актуальная, потому что транспортные компании, вот это все. Что должен сделать хороший перебор:

- Запоминать промежуточные решения, чтобы у нас была возможность экстренно остановить перебор.
- Поставить отсечения таким образом, чтобы не идти в те состояния, где **точно** не будет решений.

Задача о ферзях. Пусть мы хотим расставить n ферзей на доске $n \times n$. Как закодировать состояние? Можно, например, с помощью $2n$ координат. Заметим, что гораздо лучше зафиксировать перестановку, а потом делать ферзей (i, p_i). Тогда такие ферзи гарантированно не бьют друг друга по вертикали или горизонтали. Остается только проверить диагонали. Тогда если перебор будет строить перестановку слева направо, то мы сможем «отрезать» некоторые состояния (например, не ставить ферзя в те клетки, которые еще не бились предыдущими ферзями).

Branch & bound method. Мы будем оптимизировать две вещи:

Границы. Нам хочется понимать, когда мы сможем отрезать ветку перебора, чтобы не упустить оптимальный ответ. Например, выходить из ветки, если текущий ответ превышает нынешний оптимальный. Идеально — ввести соответствующую функцию $f(p)$, и делать так:

```
if (cur + f(p) > best) {
    return;
}
```

Такая функция должна, в случае задачи TSP, возвращать такую длину пути, который нам **точно** понадобится пройти. Например, можно взять вес остовного дерева на оставшихся вершинах. Тогда $span \le TSP \le 2 \cdot span$. Хорошая функция оценки!

Кроме того, очень хорошо иметь нормальное приближение ответа *best*. То есть, изначально как-нибудь (отжигом, жадником, итд) найти неплохой ответ, чтобы перебор не шел в заранее ущербные шаги.

Ветви. Тут мы хотим сделать какую-то магию, чтобы перебирать ветки в правильном порядке. То есть, мы хотим запускать самые хорошие ветви в самом начале, особенно на первых слоях. Например, можно ввести оценочную функцию, и запускать перебор в порядке сортировки по этой оценочной функции. В задаче TSP, из физических соображений, можно запускать перебор сначала из ближайшей вершины к текущей. Также можно попытаться прыгнуть сразу на много уровней вниз, соптимизировав это какой-то простой динамикой ($dp_{mask,i,j}$ — наименьшая длина пути через всю маску, если мы начали в i и закончили в j) на маленьких подмножествах-кластерах. Тогда мы знали самый хороший способ взять какое-то подмножество, а тогда мы вместо 2^k ходов сделаем k ходов.

В ветвях есть много пространства для спекулятивных переборов. Например, можно идти в топ-k веток по оценочной функции.

Можно инициализировать решение для неспекулятивного перебора решением из спекулятивного перебора!