1. BFS / DFS → LinkedList → Remove First / Remove Last.
   (LinkedList : BFS / DFS. ArrayList : DFS.)

2. LinkedList / ArrayList

3. What should be stored in LinkedList tries (A: 可以存比较复杂. 甚至可以自创 class).

4. Main a container (HashSet) 1. 因为 HashSet.contains(Key) O(1) 复杂度.
   2. HashSet 最好存 String 或 Integer

   * 即使 State1.level == State2.level | state1.curState[i] == state2.curState[i] 对于所有 slots.
   但是 State1 ≠ State2.

   String1 = "abc";
   String2 = "abc";
   ⇒ String1.equals(String2) → true.

5. Generate function based on current state. (nextVists(State state). escapeRoom)
   ArrayList<LinkedList> nextVisits (可以和 3. tries 连起来)

6. isFinished(State state).

Procedure:

1. 创立空的 tries 和 visited

2. 对于 initState 的 next visits 直接加到 tries
   Visited 加 initState
   Visited 加 next Vists of initState.

3 While ...

Chances of winning

1. 已知 Game 所产生的分数 Table

Key	Value
1	1
2	2
3	6
4	1

why 5 ?

int[] points = new int[5]

points[3] → 6

HashMap<Integer, Integer>

int[3] → 6 (key: 3, value: 6)

int[4] → 1 (key: 4, value: 1)

2. Two Games left (3×3).

⇒ for ... for ... } 两层for循环

Three Games left (3×3×3)

⇒

```
for state in [state1, state2, state3]:   <- Game1 3种情况.
    for stategame2 in [state1, state2, state3]   <- Game2 3种情况.
        for .....
```

三层循环

如何动态去产生多层 for 循环.

★ Recursive Function

对每一次递归，pts, gamesLeft 都会发生变化.

simulate(gamesLeft, pts, oneGame, ...)