279 完全平方数

Label：动态规划

给定正整数 n，找到若干个完全平方数（比如 1，4，9，16，...）使得它们的和等于 n。你需要让组成和的完全平方数的个数最少。
给你一个整数 n ，返回和为 n 的完全平方数的 最少数量 。
完全平方数 是一个整数，其值等于另一个整数的平方；换句话说，其值等于一个整数自乘的积。例如，1、4、9 和 16 都是完全平方数，而 3 和 11 不是。

输入：n = 12
输出：3
解释：12 = 4 + 4 + 4
输入：n = 13
输出：2
解释：13 = 4 + 9

1 <= n <= 104

- 动态规划

```
class Solution {
    public int numSquares(int n) {
        int[] dp = new int[n+1];

        for (int i = 0; i <= n; i++) {
            dp[i] = i;   // 赋值为 i 就相当于是 1+1+1+... = i，这是个数最多的，然后再来逐步更新减小

            int t = (int)Math.sqrt(i); // 求平方根，因为是正数，所以直接用 int 也能保证 t * t < i

            for (int j = 0; j <= t; j++)
                dp[i] = Math.min(dp[i], dp[i-(int)Math.pow(j, 2)] + 1);  // i-(int)Math.pow(j, 2)== 0 刚好就说明一个平方可以拿到，那么 + 1就是代表一次，如果是其他的话，就是动态规划了
        }
        return dp[n];
    }
}
```

- 广度遍历（todo）

```
class Solution {
  public int numSquares(int n) {

    ArrayList<Integer> square_nums = new ArrayList<Integer>();

    for (int i = 1; i * i <= n; ++i) {
      square_nums.add(i * i);  //  先把所有的平方数都保存
    }

    Set<Integer> queue = new HashSet<Integer>();
    queue.add(n);

    int level = 0;
    while (queue.size() > 0) {
      level += 1;
      Set<Integer> next_queue = new HashSet<Integer>();

      for (Integer remainder : queue) {
        for (Integer square : square_nums) {
          if (remainder.equals(square)) {
            return level;
          } else if (remainder < square) {
            break;
          } else {
            next_queue.add(remainder - square);
          }
        }
      }
      queue = next_queue;
    }
    return level;
  }
}
```