2 Keys Keyboard

Initially on a notepad only one character 'A' is present. You can perform two operations on this notepad for each step:

1. `Copy All`: You can copy all the characters present on the notepad (partial copy is not allowed).
2. `Paste`: You can paste the characters which are copied**last time**.

Given a number `n`. You have to get **exactly** `n` 'A' on the notepad by performing the minimum number of steps permitted. Output the minimum number of steps to get `n` 'A'.

Example 1:

```
Input: 3
Output: 3
Explanation:
Intitally, we have one character 'A'.
In step 1, we use Copy All operation.
In step 2, we use Paste operation to get 'AA'.
In step 3, we use Paste operation to get 'AAA'.
```

Note:

1. The `n` will be in the range [1, 1000].

Solution 1

```
    public int minSteps(int n) {
        int[] dp = new int[n+1];

        for (int i = 2; i <= n; i++) {
            dp[i] = i;
            for (int j = i-1; j > 1; j--) {
                if (i % j == 0) {
                    dp[i] = dp[j] + (i/j);
                    break;
                }

            }
        }
        return dp[n];
    }
```

written by LinfinityLab original link here

Solution 2

We look for a divisor `d` so that we can make `d` copies of `(n / d)` to get `n`

The process of making `d` copies takes `d` steps (`1` step of **Copy All** and `d - 1` steps of **Paste**)

We keep reducing the problem to a smaller one in a loop.

The **best** cases occur when `n` is decreasing fast, and method is almost `O(log(n))`

For example, when `n = 1024` then `n` will be divided by `2` for only `10` iterations, which is much faster than `O(n)` DP method.

The **worst** cases occur when `n` is some multiple of large prime, e.g. `n = 997` but such cases are rare.

```
    public int minSteps(int n) {
        int s = 0;
        for (int d = 2; d <= n; d++) {
            while (n % d == 0) {
                s += d;
                n /= d;
            }
        }
        return s;
    }
```

written by yuxiangmusic original link here

Solution 3

```
/**
 * It take 2 op to double, 3 ops to triple, ...
 * if n % 2 == 0, then f(n) = f(n/2) + 2
 * if n % 3 == 0, then f(n) = f(n/3) + 3
 * 2 * 2 = 2 + 2, 2 * 3 > 2 + 3, 4 * 4 > 4 + 4, so it is always better to divide wh
enever possible.
 * now it became a problem for finding all possible factors;
 */
class Solution {
public:
    int minSteps(int n) {
        if (n == 1) return 0;
        for (int i = 2; i < n; i++)
            if (n % i == 0) return i + minSteps(n / i);
        return n;
    }
};
```

written by alexander original link here