650. 2 Keys Keyboard

There is only one character `'A'` on the screen of a notepad. You can perform one of two operations on this notepad for each step:

- Copy All: You can copy all the characters present on the screen (a partial copy is not allowed).
- Paste: You can paste the characters which are copied last time.

Given an integer `n`, return *the minimum number of operations to get the character* `'A'` *exactly* `n` *times on the screen*.

Example 1:

```
Input: n = 3
Output: 3
Explanation: Initially, 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'.
```

Example 2:

```
Input: n = 1
Output: 0
```

Constraints:

- `1 <= n <= 1000`

650. 2 Keys Keyboard

```
/*
    Brute force: Recursion
    Time complexity: O(2^n)
    Space complexity: O(n)
*/

class Solution {
public:
    int minSteps(int n) {

        auto solve=[&](int i,int j,auto& self)->int{
            if(i==n) return 0;
            if(i>n) return INT_MAX/2;
            return std::min(2+self(2*i,i,self) , 1+self(i+j,j,self));
        };

        if(n==1) return 0;

        return 1+solve(1,1,solve);
    }
};
```

650. 2 Keys Keyboard

```
/*
    DP: Memoization (Top-Down)
    Time complexity: O(n^2)
    Space complexity: O(n+n^2)
*/

typedef std::vector<int> vi;
typedef std::vector<vi> vvi;

class Solution {
public:
    int minSteps(int n) {

        vvi memo(n+1,vi(n/2+1,-1));
        auto solve=[&](int i,int j,auto& self)->int{
            if(i==n) return 0;
            if(i>n) return INT_MAX/2;
            if(memo[i][j]!=-1) return memo[i][j];
            return memo[i][j]=std::min(2+self(2*i,i,self) , 1+self(i+j,j,self));
        };

        if(n==1) return 0;

        return 1+solve(1,1,solve);
    }
};
```

650. 2 Keys Keyboard

```
/*
    DP: Tabulation (bottom-up)
    Time complexity: O(n sqrt(n))
    Space complexity: O(n)
*/

typedef std::vector<int> vi;
typedef std::vector<vi> vvi;

class Solution {
public:
    int minSteps(int n) {
        vi dp(n+1,INT_MAX);
        dp[1] = 0;

        for (int i=2;i<=n;++i){
            for (int j=1;j*j<=i;++j){
                if (i%j==0) {
                    dp[i] = std::min(dp[i],dp[j]+i/j);
                    if (j!=i/j) dp[i] = std::min(dp[i],dp[i/j]+j);
                }
            }
        }

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

650. 2 Keys Keyboard

Version #1

```
/*
    Prime factorization
    Time complexity: O(n)
    Space complexity: O(1)
*/
class Solution {
public:
    int minSteps(int n) {
        int i=2;
        int ans=0;
        while(n!=1){
            while(n%i==0){
                n/=i;
                ans+=i;
            }
            i++;
        }

        return ans;
    }
};
```

Version #2

```
/*
    Prime factorization
    Time complexity: O(sqrt(n))
    Space complexity: O(1)
*/
class Solution {
public:
    int minSteps(int n) {
       int ans=0;
       int i=2;
       for(int i=2;i*i<=n;++i){
            while(n%i==0){
                n/=i;
                ans+=i;
            }
       }

       if(n>1) ans+=n;

       return ans;
    }
};
```