```
假设你有一个特殊的键盘包含下面的按键：
Key 1: (A)：在屏幕上打印一个 'A'。
Key 2: (Ctrl-A)：选中整个屏幕。
Key 3: (Ctrl-C)：复制选中区域到缓冲区。
Key 4: (Ctrl-V)：将缓冲区内容输出到上次输入的结束位置，并显示在屏幕上。
现在，你只可以按键 N 次（使用上述四种按键），请问屏幕上最多可以显示几个 'A'呢?
```

解法一： 选择有四种，分别对应四个按键； 状态有按键按下次数n，和当前屏幕上A的个数，第三个状态为剪切板中A的数量。 此时：base case全部为0，当n为0时即为所求。 状态转移如下：

```
dp(n-1, a_num+1, copy) // A, 按下A键，屏幕上a的个数+1，消耗一次操作数
dp(n-1, a_num+copy, copy) // C_V, 粘贴键，屏幕上a的个数增加剪贴板中A的个数，消耗操作数1
dp(n-2, a_num, a_num) // C_A C_V 组合使用，此时操作数消耗2次，剪贴板中a的个数变为a_num
```

算法：

```
// 递归形式
int dp(int n, int aNum, int copy) {
  if (0 >= n) {
    return aNum;
  }

  return std::max(
      std::max(dp(n - 1, aNum + 1, copy), dp(n - 1, aNum + copy, copy)),
      dp(n - 2, aNum, aNum));
}

int maxA(int N) {
  return dp(N, 0, 0);
}
```

在优化时，发现aNum和copy都为变量，难以优化。

解法二

如果选择不变，但状态即为余下步数n 最优按键序列一定只有两种情况： 要么一直按A：A,A,...A（当 N 比较小时）。 要么是这么一个形式：A,A,...C-A,C-C,C-V,C-V,...C-V（当 N 比较大时）。 因为字符数量少（N 比较小）时，C-A C-C C-V这一套操作的代价相对比较高，可能不如一个个按A；而当 N 比较大时，后期C-V的收获肯定很大。这种情况下整个操作序列大致是：开头连按几个A，然后C-A C-C组合再接若干C-V，然后再C-A C-C接着若干C-V，循环下去。

```
std::vector<int> dp = std::vector<n+1, 0>;
for(int i = 0;i<n;i++){
    dp[i] = max(
        此次按下A
        此次按下C_V
    )
}
```

算法：

```
int maxA(int n) {
  std::vector<int> dp = std::vector<int>(n + 1, 0);
  for (int i = 1; i <= n; i++) {
    // 按下A键
    dp[i] = dp[i - 1] + 1;
    for (int j = 2; j < i; j++) {
      dp[i] = std::max(dp[i], dp[j - 2] * (i - j + 1));
    }
  }

  return dp[n];
}
```