a. $\because t(n) \in O(g(n))$

$\therefore \exists n_0, c$ 使得 $t(n) \le cg(n) \quad (n > n_0)$

$\because c \ne 0$

$\therefore \exists n_0, c'$ 使得 $g(n) \ge c' t(n) \quad (n > n_0)$, $c' = \frac{1}{c}$

$\therefore g(n) \in \Omega(t(n))$

b. 设 $t(n) = n$, $g(n) = n^2$.

则 $t(n) \notin \theta(\alpha g(n))$ 但 $t(n) \in O(g(n))$

$\therefore \theta(\alpha g(n)) \ne O(g(n))$

c. 若 $t(n) \in \theta(g(n))$

$\exists n_0, c_1, c_2$ 使得 $c_1 g(n) \le t(n) \le c_2 g(n) \quad (n > n_0)$

即有 $t(n) \in O(g(n))$, $t(n) \in \Omega(g(n))$

即 $t(n) \in O(g(n)) \cap \Omega(g(n))$

若 $t(n) \in O(g(n)) \cap \Omega(g(n))$

即 $t(n) \in O(g(n)) \Rightarrow \exists n_0, c_1$ 使得 $t(n) \le c_1 g(n) \quad (n > n_0)$

同理 $\exists n_0', c_2$ 使得 $t(n) \ge c_2 g(n) \quad (n > n_0')$

有 $n > \max(n_0, n_0')$ 时 $c_2 g(n) \le t(n) \le c_1 g(n)$

即 $t(n) \in \theta(g(n))$

故 $\theta(g(n)) = O(g(n)) \cap \Omega(g(n))$

d. 设 $t(n)=\begin{cases} n, & n\text{为偶数} \\ 1, & n\text{为奇数} \end{cases}$，$g(n)=\begin{cases} n, & n\text{为3的倍数} \\ 1, & n\text{不是3的倍数} \end{cases}$

此时 $t(n)\notin O(g(n))$，$g(n)\notin O(t(n))$

a. input : n

basic operation : $i=i+1$ 的加法

check : 对所有n相同的 加法次数相同

time efficiency : $C(n)=\sum_{i=0}^{\lfloor\sqrt{n-1}\rfloor}1=\lfloor\sqrt{n}-1\rfloor+1\in\Theta(\sqrt{n})$

b. input : n

basic operation : x++

check : 对所有相同的n，x++ 的次数相同

time efficiency : $C(n)=\sum_{i=1}^{n}\sum_{k=1}^{i}\sum_{j=1}^{k}1=\sum_{i=1}^{n}\sum_{k=1}^{i}k=\sum_{i=1}^{n}\frac{(i+1)i}{2}=\frac{1}{6}n^3+\frac{1}{2}n^2+\frac{1}{3}n\in\Theta(n^3)$

No.

Date

4.

a. $T(n) = T(n-1) + n$, $T(0) = 1$

$\therefore T(n) = T(n-2) + n + n-1$

$= T(n-3) + n + (n-1) + (n-2)$

$\cdots$

$= T(0) + n + (n-1) + (n-2) + \cdots + 1$

$= 1 + \frac{n(n+1)}{2} \in \Theta(n^2)$

b. $T(n) = 4T(n-1)$, $T(1) = 5$

$\therefore T(n) = 4 \times 4T(n-2)$

$= 4^{n-1} \times 5$

$= 5 \times 4^{n-1} \in \Theta(4^n)$

c. $T(n) = T(n/3) + n$, $n>2$, $T(1) = 1$

设 $n = 3^k$ 则 $T(3^k) = T(3^{k-1}) + 3^k$

$= T(3^{k-2}) + 3^k + 3^{k-1}$

$\cdots$

$= T(3^0) + 3^k + 3^{k-1} + \cdots + 3$

$\therefore T(3^k) = \sum_{n=0}^{k} 3^n = \frac{1}{2}(3^{k+1} - 1)$

$T(n) = \frac{1}{2}(3n-1) \in \Theta(n)$

KOKUYO

No.

Date

3. a. input : n.

basic operation : $n\%2==1$ 的判断

check : 对于相同的 n，不同情况不会影响 basic operation 次数

time efficiency : $T(n) = T(n/2) + 1$, $T(0) = 0$; $n \geq 0$

令 $n = 2^k \Rightarrow T(2^k) = T(2^{k-1}) + 1$

$= T(2^{k-2}) + 1 + 1$

$= T(0) + \sum_{i=1}^{k} 1$

$\therefore T(2^k) = k$ 则 $T(n) = \log_2 n \in \Theta(\log n)$

b. input : high 和 low

basic operation : 数组元素的比较.

check : 对于相同的 high, low，不同情况会影响基本操作的次数

best : 每次找出的 pivot 都位于 low 与 high 中间，平分数组

令 $n = high - low + 1$

$T(n) = 2T(n/2) + n$, $T(1) = 0$

令 $n = 2^k$ 则 $T(2^k) = 2T(2^{k-1}) + 2^k$

$= 2(2T(2^{k-2}) + 2^{k-1}) + 2^k$

$= 2^2 T(2^{k-2}) + 2^k \times 2$

......

$= 2^k T(1) + k \cdot 2^k$

$\therefore T(n) = n\log_2 n \in \Theta(n\log n)$

worst : 数组有序，第 i 次要比较 n-i 次元素.

$T(n) = \sum_{i=1}^{n-1} n-i = \frac{n(n-1)}{2} \in \Theta(n^2)$

Q1：

• 方法 1：

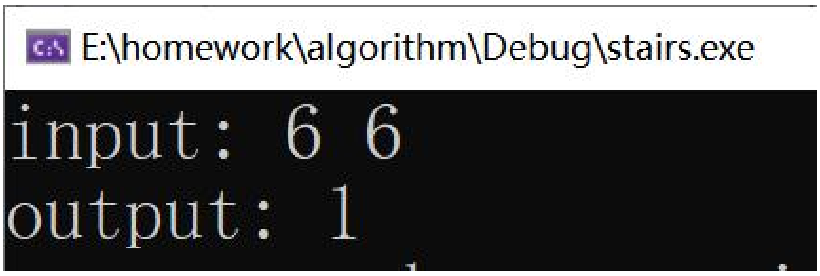
I. 首先判断台阶数目 m 是否为 1 或者 2，单独判断 m == 1 与 m == 2 的情况，即若 m == 1 时，判断卡路里 n >= 1，则返回方法数 1（利用 1 点卡路里走一步），否则返回方法数 0；若 m == 2 时，判断卡路里 n >= 3，则返回方法数 2（可以利用 3 点卡路里走两步，也可以利用 2 点卡路里走两次一步），否则判断若 n >= 2，返回方法数 1（尽可以利用 2 点卡路里走两次一步），否则返回方法数 0。

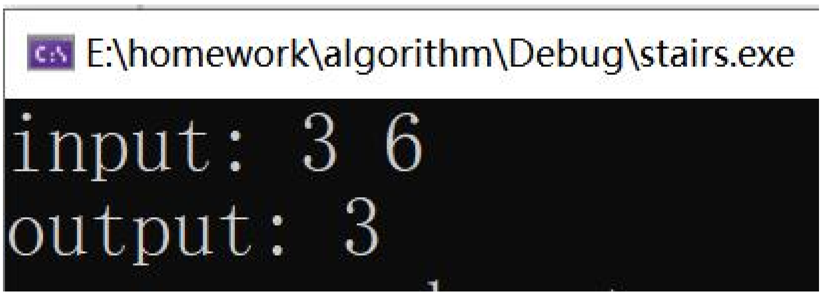
II. 若 m >= 3，设置达到每层总共方法数 num_of_ways 以及可以有的走一步的次数、走两步的次数以及所消耗的卡路里的组合，每个组合再映射到该组合的组合数。求解第 i 层的参数时，第 i 层的方法数 num_of_ways 可以从第 i - 1 层走一步上来的方法数加上第 i - 2 层走两步上来的方法数；然后通过 i - 1 层以及 i - 2 层的走一步的次数、走两步的次数以及所消耗的卡路里的组合对第 i 层的进行更新，若出现相同的则进行合并；同时更新这些组合到组合数的映射。

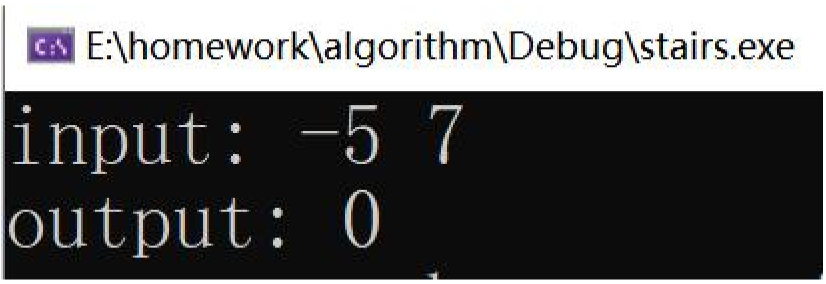
III. 重复 II 中步骤，直到计算到第 m 层台阶为止。

IV. 第 m 层所对应的 num_of_ways 即为所求。

测试：







• 方法 2：

I. 设爬上 m 个台阶需要 x 次一步，y 次两步，则有关系

$$\begin{cases} x + 2y = m \\ x + 3y <= n \end{cases}$$

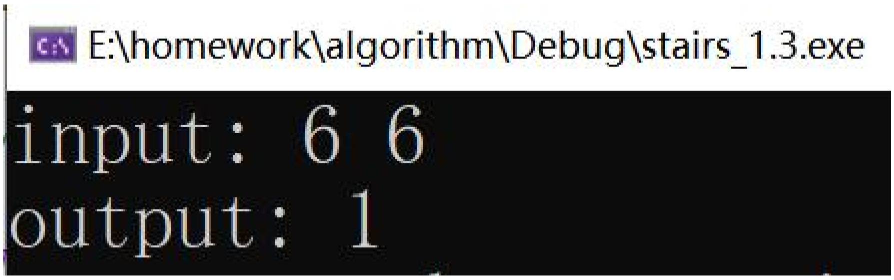
并且可得出关系 0 <= y <= n - m

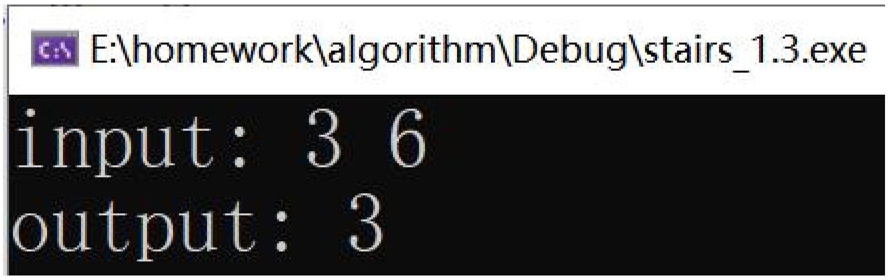
II. 若 n - m < 0，则说明卡路里不足以爬上 m 个台阶，放回方法数 0

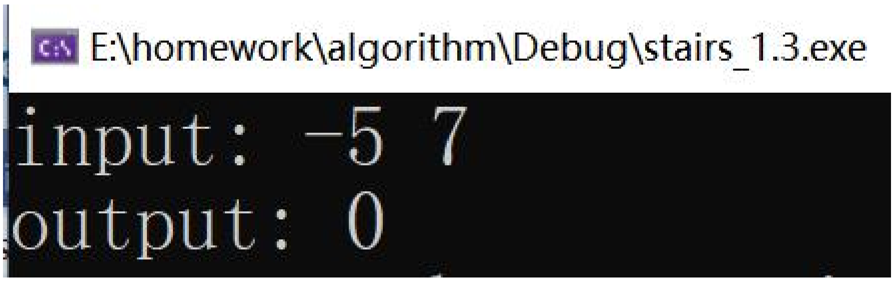
III. 否则，遍历 y 从 0 到 n - m，通过 y 求解出 x，保证 x >= 0 的前提下，由此得出 x 次一步与 y 次两步的组合，然后通过 x 和 y 求出该组合对应的组合数 $C_{x+y}^{\min(x,y)}$。

IV. 重复 II 中操作，直到将所有组合数相加，其和即为所求。

测试







Q2：

• 方法 1：

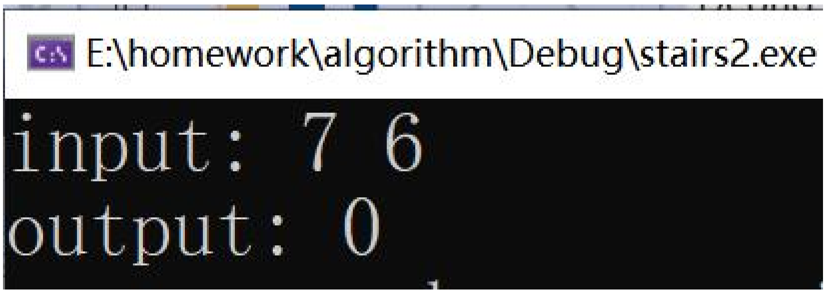
i. 首先判断台阶数目 m 是否为 1 或者 2，单独判断 m == 1 与 m == 2 的情况，即若 m == 1 时，判断卡路里 n >= 1，则返回方法数 1（利用 1 点卡路里走一步），否则返回方法数 0；若 m == 2 时，判断卡路里 n >= 3，则返回方法数 2（可以利用 3 点卡路里走两步，也可以利用 2 点卡路里走两次一步），否则判断若 n >= 2，返回方法数 1（尽可以利用 2 点卡路里走两次一步），否则返回方法数 0。

ii. 若 m >= 3，设置达到每层总共方法数 num_of_ways 以及可以有的走一步的次数、走两步的次数以及所消耗的卡路里的组合，每个组合再映射到该组合的组合数。求解第 i 层的参数时，第 i 层的方法数 num_of_ways 可以从第 i - 1 层走一步上来的方法数加上第 i - 2 层走两步上来的方法数；然后通过 i - 1 层以及 i - 2 层的走一步的次数、走两步的次数以及所消耗的卡路里的组合对第 i 层的进行更新，若出现相同的则进行合并；同时更新这些组合到组合数的映射。

iii. 重复 II 中步骤，直到计算到第 m 层台阶为止

iv. 在最高层 m 层对应的组合到组合数的映射中，寻找走两步的次数最多的组合，该组合映射的组合数即为所求。

测试



- 方法 2：

I.设爬上 m 个台阶需要 x 次一步，y 次两步，则有关系

$$\begin{cases} x + 2y = m \\ x + 3y <= n \end{cases}$$

并且可得出关系 0 <= y <= n - m

II. 遍历 y 从 n - m 到 0，求解出 x 的值，在 x >= 0 的前提下，由此得出 x 次一步与 y 次两步的组合，然后通过 x 和 y 求出该组合对应的组合数 $C_{x+y}^{\min(x,y)}$ 。

III. 跳出循环，所求组合数即为所求。

测试