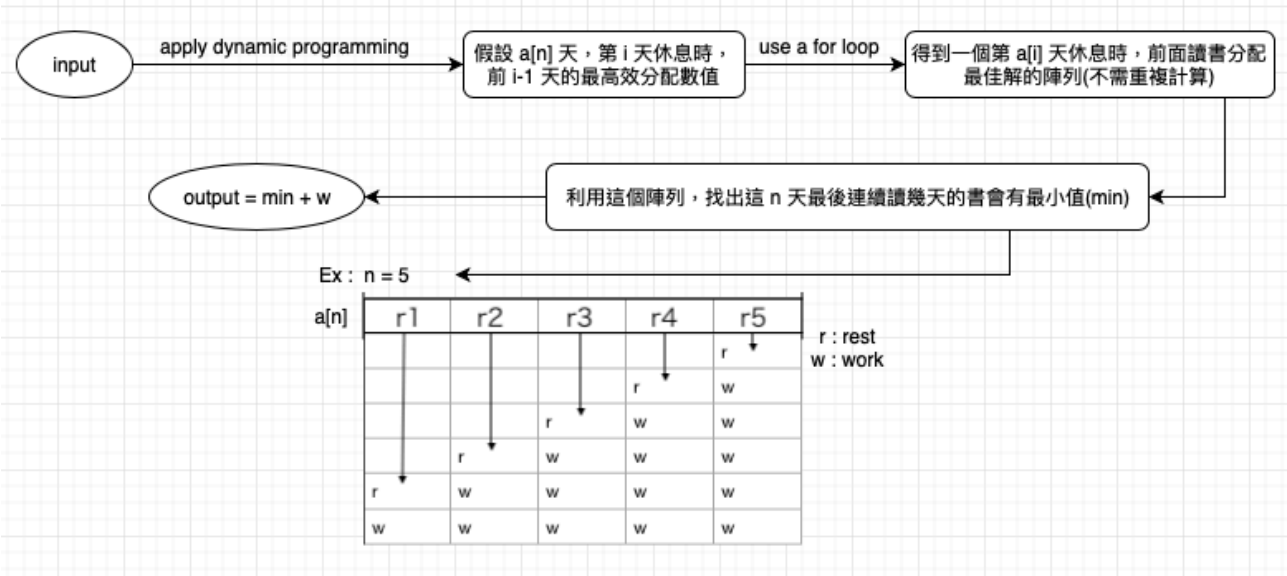
106081005 許雅晴

FlowChart



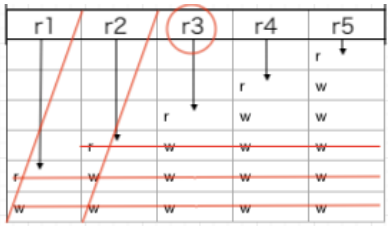
* 每天只能選擇睡覺或讀書，用 bottom-up 計算「i-1 天最小值 + D[i] = 第 i 天休息 」的最佳解
* 以休息當作紀錄點是為了去除與前面幾天的關聯性(b*n)，才方便使用DP
* 如圖設 n = 5，最後一輪有 6 種解法，找出這 6 種的 min+w 即答案，可再優化(小於 6 種的計算)
* 把休息視為一個斷點時，在此之前的計算也會 overlapping，優化主要的方向也是減少需要連續念書的計算的地方。

Optimize

* 若 b = 0，代表不休息也不會有代價， $output = w - a * n$
* 設 a[i] 為 i-1 前的最小值(為方便運算右移一位)。當 n = 5，假設我們在第 3 天的時候休息能得到最小值，那不管第 4、5 天的作息如何分配，前 3 天得到 min 值的方式都不會變(overlapp)。所以紀錄第三天為一個 breakpoint，連續讀書的計算不需要再往這之後了。如下圖能減少 3 次計算。

```
long long find_min(int i, long long *zero){
    if( i<= 0) return 0;
    long long min = zero[i], total, brk;

    for(int k=1; k<=i; k++){
        total = zero[i-k] - a*k + b*(1+k)*k/2 ;
        if( total < min) min = total, brk = i-k;
        if( i-k < temp) break;
    }
    temp = brk;
    return min;
}
```



Time complexity

* 有 n 天， 每 i 天還要找 i-1 種的解法 -> 兩層 for loop -> $T(n) = T(n-1) + O(n) = O(n^2)$
* 在普通情況下，優化後大部分第二層迴圈的次數都會減少到常數次 -> $O(n)$ —— 當 $0 < D[i] < b * m - a$ ，休息才會取得平衡，因此大概 m 天要休息一次，m 為常數，不過仍有 worst case 的情況如： A = 10^5 ，B = 1，若 D[i] 都很大還是有可能會跑 $O(n^2)$