4.2 節末問題 4.2 的解答

問題 4.2.1

本問題中，如果用簡單的方法計算每個經過地點之間的距離，計算複雜度為 $O(N)$，整體計算複雜度為 $O(NM)$，無法滿足本問題的執行時間限制，但若使用累積和（→**4.2.1 項**）的概念，可以將演算法進行改進。。

首先，令 $X < Y$ 時，以下性質成立。

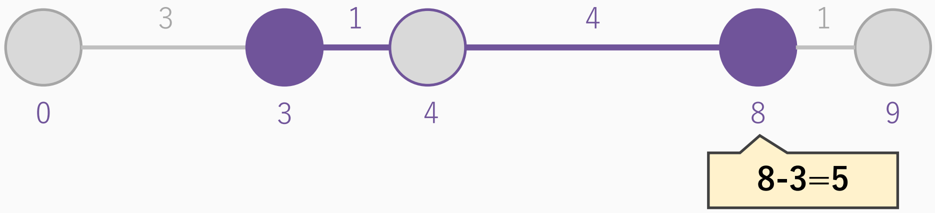
(從站 X 到站 Y 的距離)

$=$ (從站 1 到站 Y 的距離 S_Y) $-$ (從站 1 到站 X 的距離 S_X)

下圖是當 $N = 5, (A_1, A_2, A_3, A_4) = (3, 1, 4, 1)$ 時的具體例。從站 2 到站 4 的距離可以計算為$1 + 4 = 5$，但也可以使用以下數值，求出 $8 - 3 = 5$ 來取代。

- 從站 1 到站 4 的距離： 8
- 從站 1 到站 2 的距離： 3

當$X > Y$時，將 X 和 Y 顛倒過來即可。



因此，從站 1 到站 i 的距離為 $S_i = A_1 + \cdots + A_{i-1}$，對數列 $[A_1, A_2, \ldots, A_N]$ 取累積和，可以得到數列 $[S_1, S_2, \ldots, S_N]$。因此，如下實作時，可以得到正確答案。計算複雜度為 $O(N + M)$。

```
#include <iostream>
using namespace std;

int N;
int A[200009], B[200009]; // 站間距離、累積和
```

```
int main() {
    // 輸入
    cin >> N;
    for (int i = 1; i <= N - 1; i++) cin >> A[i];
    cin >> M;
    for (int i = 1; i <= M; i++) cin >> B[i];

    // 取累積和
    S[1] = 0;
    for (int i = 2; i <= N; i++) S[i] = S[i - 1] + A[i - 1];

    // 求出答案
    long long Answer = 0;
    for (int i = 1; i <= M - 1; i++) {
        if (B[i] < B[i + 1]) {
            Answer += (S[B[i + 1]] - S[B[i]]);
        }
        else {
            Answer += (S[B[i]] - S[B[i + 1]]);
        }
    }

    // 輸出
    cout << Answer << endl;
    return 0;
}
```

※ Python等原始碼請參閱 chap4-2.md。

問題 4.2.2

這個問題可以照以下步驟解決：

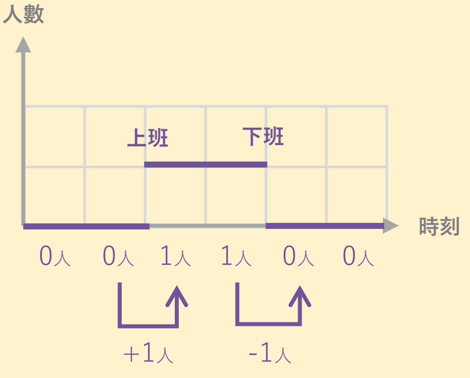
- 計算時刻 $t - 0.5$ 和時刻 $t + 0.5$ 之間的員工數差 B_t 。
- 對列 $[B_1, B_2, \ldots, B_T]$ 取累積和，得到列 $[A_1, A_2, \ldots, A_T]$ 。

關於差 B_i 可以如下計算。

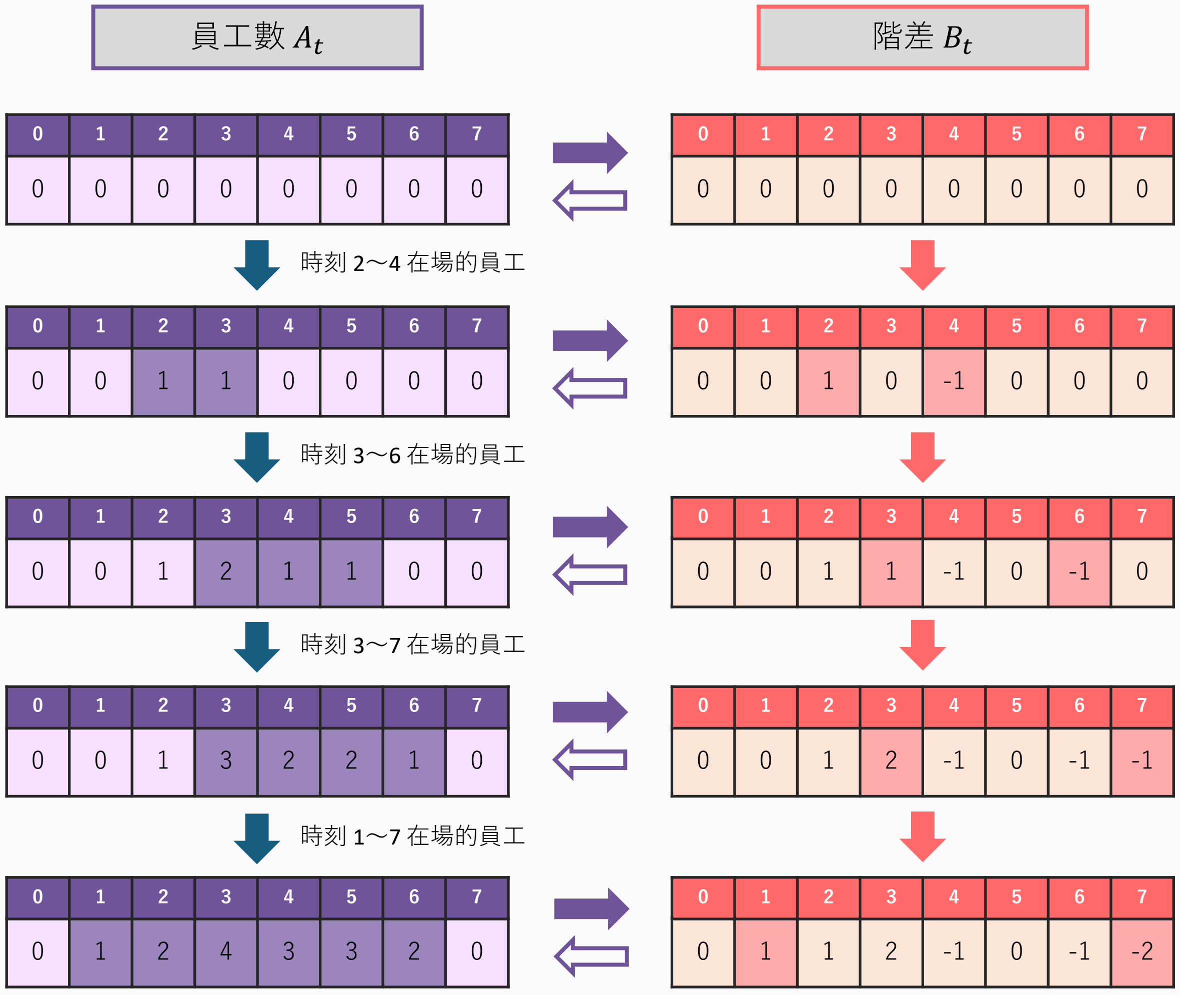
對於在時刻 L 上班、時刻 R 下班的員工，進行以下操作：

- 對 B_L 加 1 。
- 對 B_{R+1} 減 1 。

在進行操作前，初始化為 $B_i = 0$ 。



例如，當 $T=8, (L_i, R_i) = (2,4), (3,6), (3,7), (1,7)$ 時，差 B_t 和員工數 A_t 的變化如下所示。



因此，撰寫程式對各個 i $(1 \le i \le N)$，於$B[L_i]$ $+1$ 且於 $B[R_i]$ -1 之後，輸出陣列 B 的累積和，即可得到正解。計算複雜度為 $O(N+T)$。

```
#include <iostream>
using namespace std;

int N, T;
int L[500009], R[500009];
int A[500009], B[500009];

int main() {
	// 輸入
	cin >> T >> N;
	for (int i = 1; i <= N; i++) cin >> L[i] >> R[i];

	// 計算階差 B[i]
	for (int i = 0; i <= T; i++) B[i] = 0;
```

```
	for (int i = 1; i <= N; i++) {
		B[L[i]] += 1;
		B[R[i]] -= 1;
	}

	// 計算累積和 A[i]
	A[0] = B[0];
	for (int i = 1; i <= T; i++) {
		A[i] = A[i - 1] + B[i];
	}

	// 輸出答案
	for (int i = 0; i < T; i++) cout << A[i] << endl;
	return 0;
}
```

※ Python等原始碼請參閱 chap4-2.md。

問題 4.2.3

首先，如果可以證明以下兩點，就能知道只有以 $f(x) = ax^2 + bx + c$ 的形式表示時，才會回傳 `true` 。

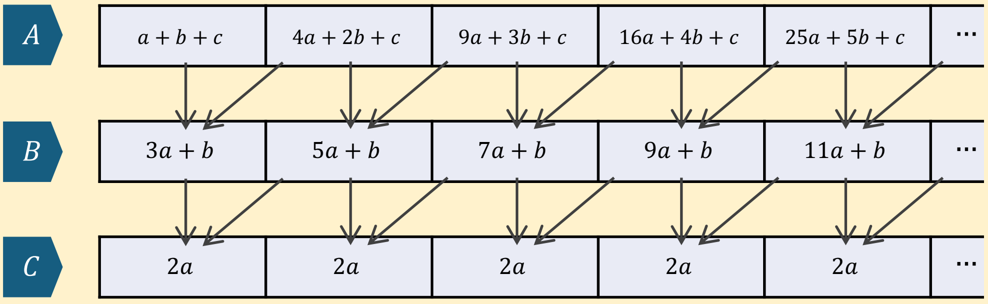
1. 如果為 $f(x) = ax^2 + bx + c$ 的形式，則回傳 `true` 。
2. 當回傳 `true` 時，是以 $f(x) = ax^2 + bx + c$ 的形式表示。

對以上兩點分別進行證明吧。

1. 的證明

當 $f(x) = ax^2 + bx + c$ 時，$A[1] = a + b + c$、$A[2] = 4a + 2b + c$、$A[3] = 9a + 3b + c$、$\cdots$ 以此類推。。

因此，由於 $B[1] = A[2] - A[1]$，故$B[1] = 3a + b$ 。其他情況也進行計算，如下表所示。。

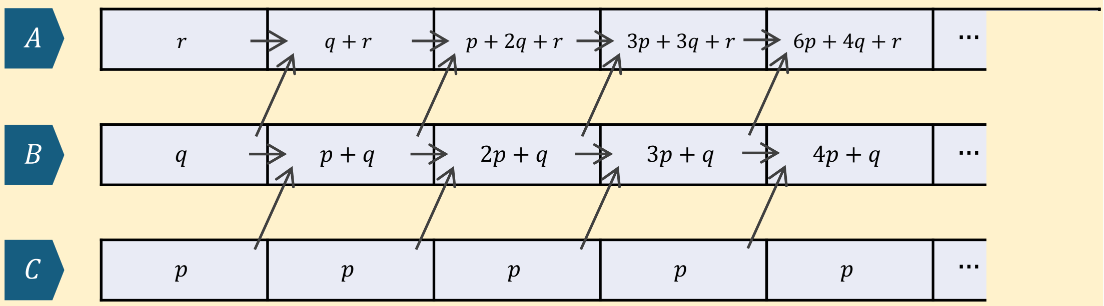


因為 $C = [2a, 2a, \ldots, 2a]$，回傳 `true` 。

2. 的證明

思考函數 `func` 回傳 `true` 的狀況，即 $C[1]=C[2]=\cdots=C[N]=p$ 。

階差是累積和的相反操作，所以 B 是 C 的累積和，A 是 B 的累積和。因此，若 $A[1]=r, B[1]=q$ ，則例如 $B[2]=p+q$ 。其他情況也進行計算，如下表所示。



因此，當$\left[A[1], A[2], \ldots, A[N]\right]=[r, q+r, p+2q+r, 3p+3q+r, \ldots]$ 時，可以表示成：

$$f(x)=A[x]=\left(\frac{1}{2}x^2-\frac{3}{2}x+1\right)p+(x-1)q+r$$

此為二次函數（$f(x)=ax^2+bx+c$ 的形式），因此可證明 2. 。

證明如上。此外，由於已知當 $f(x)$ 為 K 次函數時，取一次階差變成 $K-1$ 次函數，因此取K次階差後所有元素變為相同。（本問題是 $K=2$ 的情況）