Chú ý: khi xem các GIF hướng dẫn, tua chậm hoặc dừng khi cần để hiểu rõ bài giải.

I. Cộng dồn trên mảng một chiều

1. KHỞI TẠO:

- Cho mảng ***A*** có ***n*** phần tử, ta dựng mảng ***S(A)*** theo quy tác:

- ***S[0]*** = ***c*** với ***c*** là một hằng số thực.
- **Công thức: *S[i]* = *S[i-1] + A[i]*** với $1 \le i \le n$.

- Ta gọi ***S(A)*** là mảng cộng dồn theo ***c*** của ***A*** *(prefix sum của A)*. Từ đó, ta có ***S[i]*** là tổng của ***i*** phần tử đầu tiên của mảng ***A***.

- Từ mảng ***A*** có thể sinh ra vô số mảng ***S(A)*** tùy theo hằng số ***c***. Nhưng ta thường đặt ***c*** = **0** để thuận tiện cho việc tính toán.

- Xem GIF minh họa: imgur.com/lzBYJ89

- Cài đặt: *($n \le 10^6$)*

```
int n; cin >> n;
int A[1000001], S[1000001];
S[0] = 0; // Hằng số c = 0
for (int i=1; i<=n; i++) {
    cin >> A[i];
    S[i] = S[i-1] + A[i];
}
```

2. TRUY CẬP & SỬ DỤNG:

- Ví dụ: Nhập vào một dãy ***a*** nguyên có ***n*** phần tử *($n \le 10^6$)* và hai số nguyên dương ***L, R*** *($1 \le L < R \le n$)*. Hãy tính tổng các phần tử thuộc đoạn *[L; R]*.

INPUT	OUTPUT
7 2 5 5 6 -4 1 0 3 7	8

- **Hướng dẫn giải:** Từ mảng cộng dồn đã được dựng, ta có thể tìm được tổng của đoạn [L; R] như sau:

Công thức: ***Sum(L, R)*** **=** ***S[R] - S[L-1]*** với $1 \le L < R \le n$.

Trong đó:
*+ **L** là vị trí bắt đầu của đoạn cần tính tổng.*
*+ **R** là vị trí kết thúc của đoạn cần tính tổng.*
*+ **sum(L, R)** là tổng các phần tử thuộc đoạn [L; R] trong mảng.*

- Với bài toán trên, ta có:

				L				R
i	0	1	2	3	4	5	6	7
A[i]		2	5	5	6	-4	1	0
S[i]	0	2	7	12	18	14	15	15

- **L** = 3, **R** = 7.
- **S**[**L**-1] = 7, **S**[**R**] = 15.
- **S**[**R**] - **S**[**L**-1] = 15 - 7 = 8.

- **Cài đặt:** Độ phức tạp **O(n)**

```
int n; cin >> n;
int S[1000001], A[1000001];
S[0] = 0;
for (int i=1; i<=n; i++) {
    cin >> A[i];
    S[i] = S[i-1] + A[i];
}
int L, R; cin >> L >> R;
int res = S[R] - S[L-1];
cout << res;
```

3. BÀI TẬP VẬN DỤNG:

Bài 1: **EASY** (*cses.fi/problemset/task/1661*)

Cho dãy số nguyên ***a*** có ***n*** (phần tử và một số nguyên ***x***. Đếm có bao nhiêu dãy con thuộc dãy ***a*** có tổng bằng ***x***. ($1 \le n \le 2.10^5$, $-10^9 \le x, a[i] \le 10^9$).

Dữ liệu:

- Dòng đầu tiên là 2 số nguyên dương ***n*** và ***x***.
- Dòng tiếp theo là dãy số nguyên ***a***.

Kết quả: in ra kết quả cần tìm trên một dòng duy nhất.

INPUT	OUTPUT
5 7 2 -1 3 5 -2	2

Bài 2: **MEDIUM** *(oj.vnoi.info/problem/nkseq)*

Cho dãy số nguyên ***a*** có ***n*** phần tử (*$1 \le n \le 10^6$, $-10^5 \le a[i] \le 10^5$)* được viết trên một vòng tròn. Khi cắt vòng tròn tại vị trí ***j***, ta thu được: *a[j], a[j+1], a[j+2], ..., a[n], a[1], a[2], ..., a[j-1]*. Hãy đếm có bao nhiêu vị trí cắt ***j*** tốt? . Biết rằng vị trí cắt j tốt cần thỏa mãn các điều kiện:

- $a_j > 0$
- $a_j + a_{j+1} > 0$
- ....
- $a_j + a_{j+1} + ... + a_n > 0$
- $a_j + a_{j+1} + ... + a_n + a_1 > 0$
- ...
- $a_j + a_{j+1} + ... + a_n + a_1 + a_2 + ... + a_{j-2} > 0$
- $a_j + a_{j+1} + ... + a_n + a_1 + a_2 + ... + a_{j-2} + a_{j-1} > 0$

Dữ liệu:

- Dòng đầu tiên là số nguyên dương ***n***.
- Dòng tiếp theo là dãy số nguyên ***a***.

Kết quả: in ra kết quả cần tìm trên một dòng duy nhất.

INPUT	OUTPUT
5 0 1 -2 10 3	2

II. Cộng dồn trên mảng hai chiều

1. KHỞI TẠO:

- Đối với mảng hai chiều A có n hàng và m cột, mảng S(A) được dựng như sau:

Công thức: (*∀ i, j : $1 \le i \le n$, $1 \le j \le m$)*
S[i][j] = S[i-1][j] + S[i][j-1] - S[i-1][j-1] + A[i][j]

- Tức là ô S[i][j] sẽ lưu tổng các phần tử thuộc hình chữ nhật *[1, i] x [1, j]*.

- Sự khác biệt so với mảng một chiều là ta lược bỏ hằng số thực c. Ta ngầm quy ước: ***S****[i][0]* = ***S****[0][j]* = ***0***.

- Ví dụ: xem hình ảnh bên dưới và GIF: *imgur.com/3vhyF37*

	1	2	3	4
1	3	-6	7	8
2	-7	9	4	-4
3	1	-1	7	1
4	4	4	-2	-3

A

	0	1	2	3	4
0	0	0	0	0	0
1	0	3	-3	4	12
2	0	-4	-1	10	14
3	0	-3	-1	17	22
4	0	1	7	23	25

S

- Cài đặt: *($n, m \le 100$)*

```
int n, m; cin >> n >> m;
int A[101][101], S[101][101];
S[0][i] = S[0][j] = 0;
for (int i=1; i<=n; i++) {
    cin >> A[i][j];
    S[i][j] = S[i-1][j] + S[i][j-1] - S[i-1][j-1] + A[i];
}
```

2. TRUY CẬP & SỬ DỤNG:

- Ví dụ: Nhập vào một dãy ***a*** nguyên 2 chiều ***n*** dòng và ***m*** cột *($n, m \le 100$)* và hai tọa độ (x_1, y_1) và (x_2, y_2) *($1 \le x \le n$, $1 \le y \le m$)*. Hãy tính tổng các phần tử thuộc hình chữ nhật có góc trái trên là (x_1, y_1) và góc phải dưới là (x_2, y_2).

Dữ liệu:

- Dòng đầu tiên là 2 số nguyên dương ***n*** và ***m***.
- Trên ***n*** dòng tiếp theo, mỗi dòng có ***m*** số nguyên: số nguyên thứ ***j*** ở dòng thứ ***i*** là phần tử ***a[i][j]***.
- Dòng tiếp theo là tọa độ **(x_1, y_1)**.
- Dòng cuối cùng là tọa độ **(x_2, y_2)**.

Kết quả: in ra kết quả cần tìm trên một dòng duy nhất.

INPUT	**OUTPUT**
4 4 3 -6 7 8 -7 9 4 -4 1 -1 7 1 4 4 -2 -3 2 2 3 4	16

- Hướng dẫn giải: Để tính tổng các phần tử của một hình chữ nhật cho trước hai tọa độ trái trên và phải dưới, ta lấy tổng của hình chữ nhật *[1, x_2] x [1, y_2]* trừ đi phần ***không*** thuộc hình chữ nhật cần tính.

Công thức: *($\forall$ x, y : 1 ≤ x ≤ n, 1 ≤ y ≤ m)*

$Sum([x_1, x_2 - x_1] \times [y_1, y_2 - y_1]) = S[x_2][y_2] - S[x_1 - 1][y_2] - S[x_2][y_1 - 1] + S[x_1 - 1][y_1 - 1]$

- Xem GIF minh họa: imgur.com/aqfFaz3

- **Cài đặt:**

```
int n, m; cin >> n >> m;
int A[101][101], S[101][101];
S[0][i] = S[0][j] = 0;
for (int i=1; i<=n; i++) {
    cin >> A[i][j];
    S[i][j] = S[i-1][j] + S[i][j-1] - S[i-1][j-1] + A[i];
}
int x1, y1; cin >> x1 >> y1;
int x2, y2; cin >> x2 >> y2;
int res = S[x2][x2] - S[x1-1][y2] - S[x2][y1-1] + S[x1-1][y1-1];
cout << res;
```

3. BÀI TẬP VẬN DỤNG:

Cho mảng hai chiều nguyên ***a*** có kích thước ***n*m*** và một số nguyên dương ***k***. Tìm hình vuông có cạnh ***k*** sao cho tổng các phần tử thuộc hình vuông đó là lớn nhất. (*1≤ n, m ≤ 100, -1000 ≤ a[i][j] ≤ 1000).* In ra tổng của hình vuông tìm được.

Dữ liệu:

- Dòng đầu tiên là 3 số nguyên dương ***n***, ***m*** và ***k***.
- Trên ***n*** dòng tiếp theo, mỗi dòng có ***m*** số nguyên: số nguyên thứ ***j*** ở dòng thứ ***i*** là phần tử ***a[i][j]***.

Kết quả: in ra kết quả cần tìm trên một dòng duy nhất.

INPUT	**OUTPUT**
4 4 2 3 -6 7 8 -7 9 4 -4 1 -1 7 1 4 4 -2 -3	19

III. Mảng hiệu *(mở rộng)*

Lưu ý: Trái ngược với mảng cộng dồn được sử dụng rộng rãi, mảng hiệu lại rất ít được dùng trong lập trình thi đấu cơ bản. Đặc biệt, trong các dạng đề thi HSGTP9 và TS10-HCM cũng ít có bài cần dùng mảng hiệu (hoặc có thể thay thế bằng mảng cộng dồn) nên đây chỉ là phần mở rộng để bạn đọc biết thêm về chủ đề tháng này.

- Với một mảng A có n phần tử, ta có thể dựng được mảng hiệu D(A) như sau:

Công thức: ***D[i]*** **=** ***A[i+1] - A[i]*** với $1 \le i \le n$.

- Cài đặt: *($n \le 10^6$)*

```
int n; cin >> n;
int A[1000001], D[1000001];
for (int i=1; i<=n; i++) cin >> A[i];
for (int i=n-1; i>=1; i--)
    D[i] = A[i+1] - A[i];
```

- So sánh tính chất:

Nội dung	Mảng cộng dồn	Mảng hiệu
Độ dài	Đối với **mảng cộng dồn**, do ta cần thêm một hằng số thực ***c*** ở đầu mảng nên độ dài là ***n*+*1***. Nhiều hơn 1 phần tử so với mảng ***A*** gốc.	**Mảng hiệu** được dựng dựa trên hiệu của hai phần tử liền kề nhau. Tuy nhiên, trong mảng A chỉ có ***n−1*** cặp như vậy, vì thế độ dài của ***D(A)*** là ***n−1***, ít hơn 1 phần tử so với mảng ***A*** gốc.
Tính riêng biệt	Từ một mảng ***A***, ta có thể sinh được vô số **mảng cộng dồn** ***S(A)*** tùy vào hằng số c.	Với mỗi mảng ***A***, ta chỉ có thể dựng được một và chỉ một **mảng hiệu** ***D(A)***.

IV. Tổng kết và dặn dò

✏️ Mảng cộng dồn có thể được sử dụng để giải quyết rất nhiều bài toán ở nhiều mức độ khác nhau. Ở bài học này chỉ giới thiệu về mảng cộng dồn cơ bản để các bạn làm quen với các công thức và cách tư duy của giải thuật.

✏️ Về sau, mảng cộng dồn sẽ có "đa dạng" cách cài đặt và sử dụng hơn. Đặc biệt, giải thuật này có thể được sử dụng xuyên suốt các contest của GG9.

✏️ Đón chờ contest luyện tập sắp tới tại **GDOJ**.

✏️ Để theo dõi dự án, vui lòng tham gia nhóm **Guidelines for Grade 9**.

✏️ Tham gia đoạn chat cộng đồng: m.me/ch/Aba9cpzNk_Td6_fl/ để giao lưu cùng các bạn đồng môn và các anh chị học sinh chuyên Tin trên Tp.HCM.

✏️ Thông tin liên hệ:
- Page: **Căng TIN 2326**
- Group: **Guidelines for Grade 9**
- Gmail: cangtin2326@gmail.com

>> Nguồn tham khảo <<
- VNOI: vnoi.info/wiki/algo/data-structures/prefix-sum-and-difference-array.md
- Usaco: usaco.guide/silver/prefix-sums