

Hình học tính toán phần 1: Những khái niệm cơ bản

Hình học tính toán phần 1: Những khái niệm cơ bản

Tác giả:

- Lê Minh Hoàng - Phổ thông Năng khiếu, ĐHQG-HCM

Reviewer:

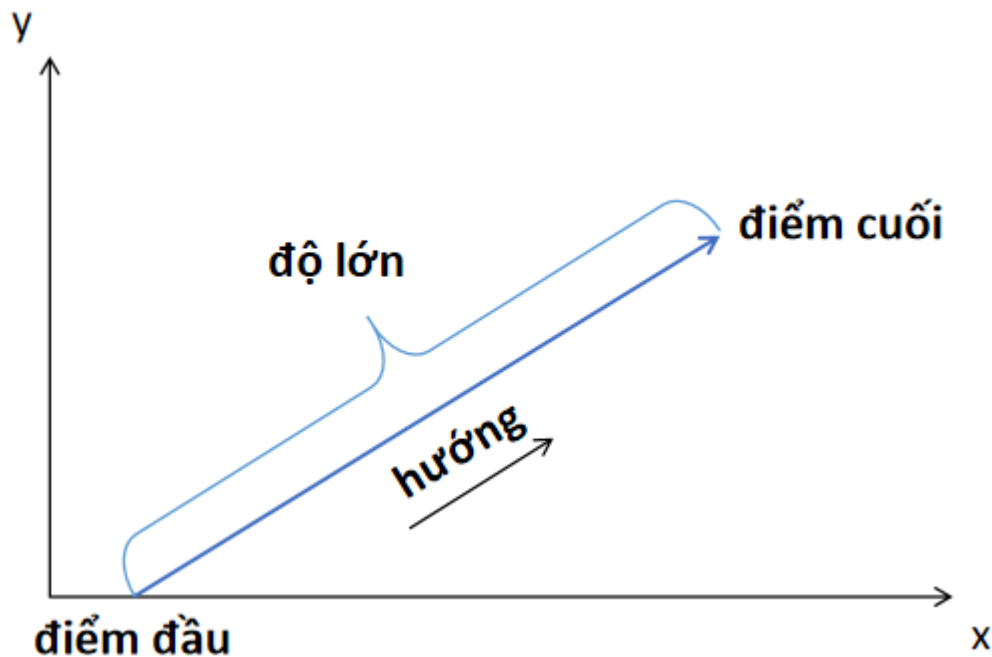
- Trần Quang Lộc - ITMO University
- Hoàng Xuân Nhật - Đại học Khoa học Tự nhiên, ĐHQG-HCM
- Hồ Ngọc Vĩnh Phát - Đại học Khoa học Tự nhiên, ĐHQG-HCM

Hình học mặc dù là một chủ đề hết sức phổ biến, song vẫn còn rất nhiều người không thích giải các bài toán hình học vì chúng khá khó chịu và lằng nhằng. Do đó, trong bài viết này, ta sẽ cùng tìm hiểu một vài khái niệm nhằm giúp cho các bài toán hình học trở nên bớt đáng sợ hơn.

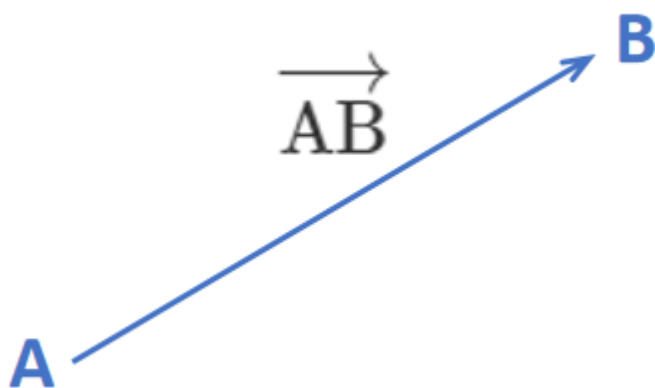
Nếu đã nắm vững các khái niệm trong bài viết này, các bạn có thể chuyển sang [phần 2](#).

Vector

Vector là một đối tượng có cả độ lớn và hướng. Hướng của vector là hướng từ điểm đầu đến điểm cuối của nó.



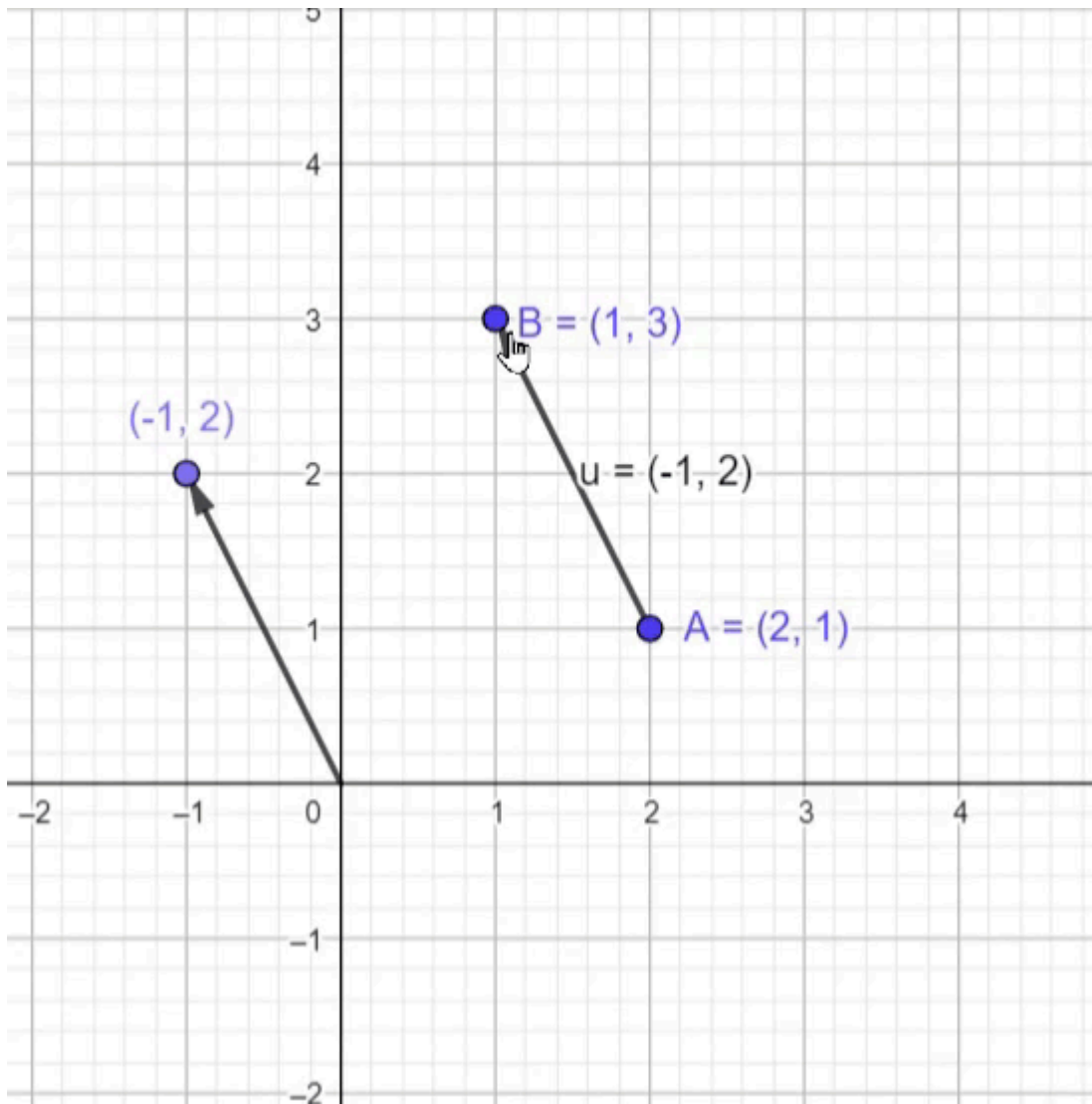
Một vector thường được biểu diễn bằng một tia (một đoạn thẳng có hướng), hoặc bằng đồ thị dưới dạng một mũi tên nối từ điểm đầu A tới điểm cuối B , và được ký hiệu là \overrightarrow{AB} .



Trong hình học phẳng, vector \overrightarrow{AB} có thể được biểu diễn bởi một cặp số (x, y) cho biết tọa độ của vector, được xác định bằng hiệu các tọa độ tương ứng của điểm cuối B với điểm đầu A :

$$\begin{cases} x = x_B - x_A \\ y = y_B - y_A \end{cases}$$

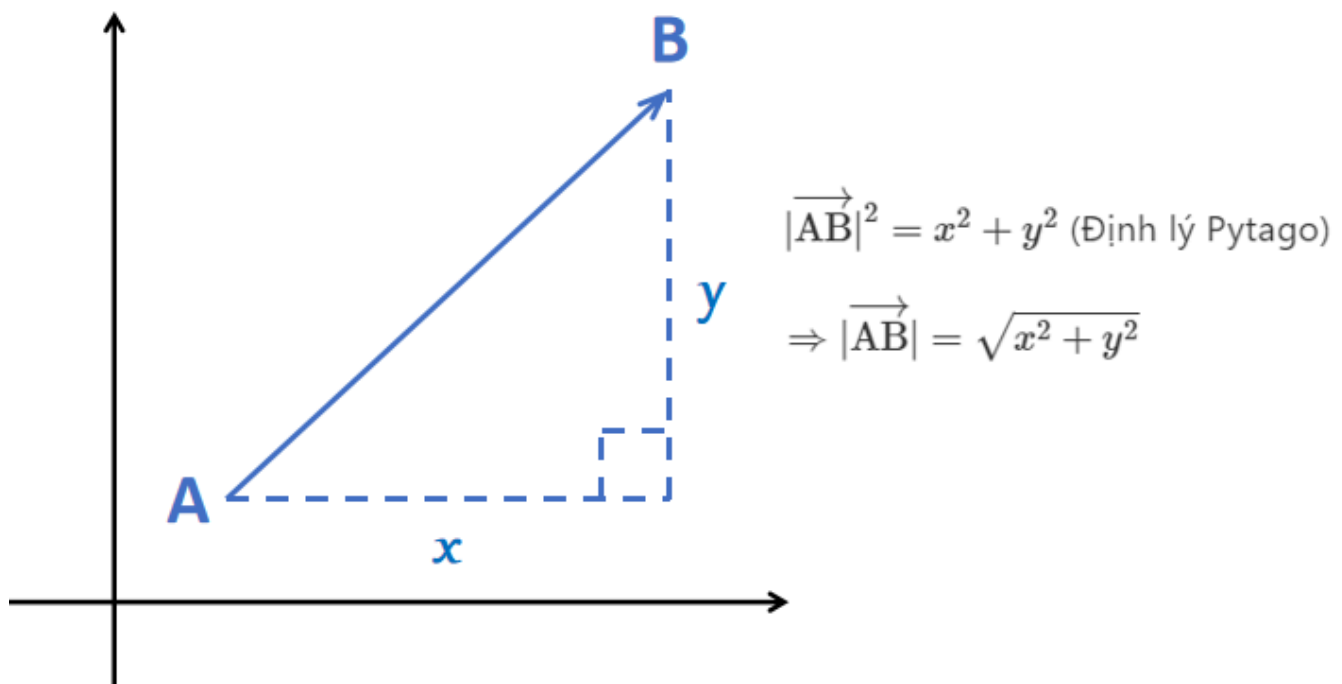
Ví dụ: một vector từ $A(3, 1)$ đến $B(2, 3)$ có thể được biểu diễn bởi $\vec{u} = (-1, 2)$.



Độ lớn của vector

Độ lớn của 1 vector được xác định bằng khoảng cách giữa điểm đầu và điểm cuối của nó.

Ví dụ: Độ lớn của $\overrightarrow{AB}(x, y)$ kí hiệu là $|\overrightarrow{AB}|$ và được xác định bằng:

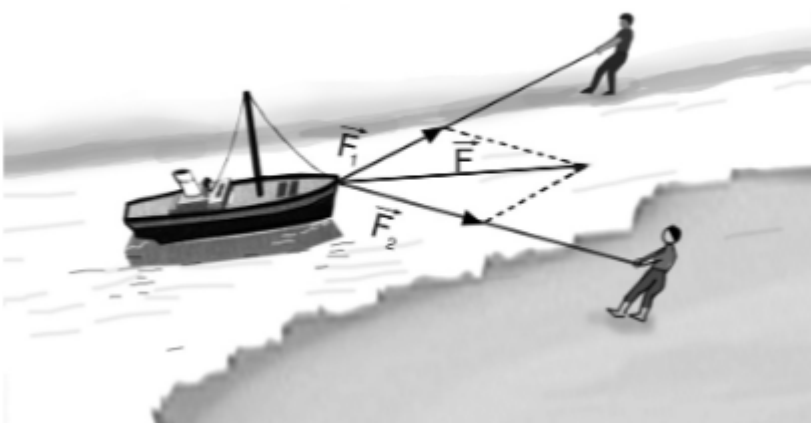


Phép cộng trừ vector

Có một số phép toán có thể thực hiện trên vector, đơn giản nhất là phép cộng trừ vector: bạn có thể cộng trừ 2 vector với nhau để được một vector mới.

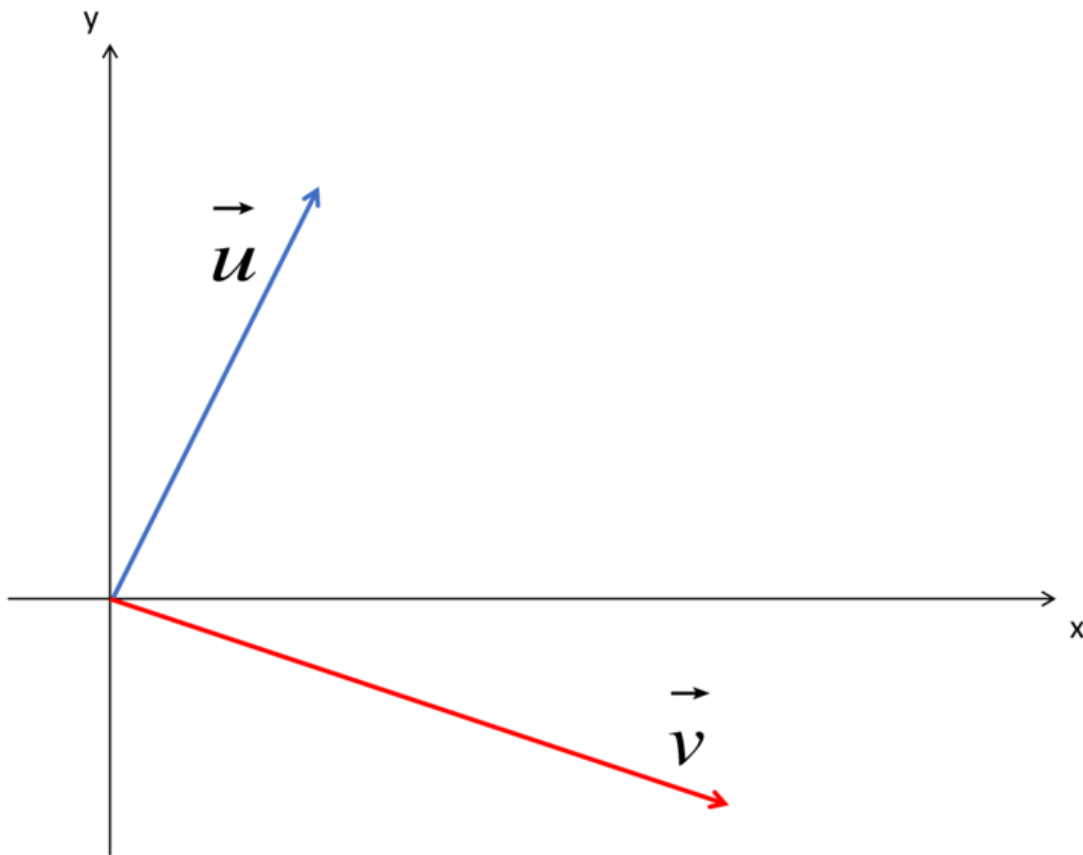
Xuyên suốt bài viết, chúng ta sẽ dùng dấu cộng (+) và trừ (-) để biểu diễn cho phép cộng và trừ vector.

Phép cộng 2 vector

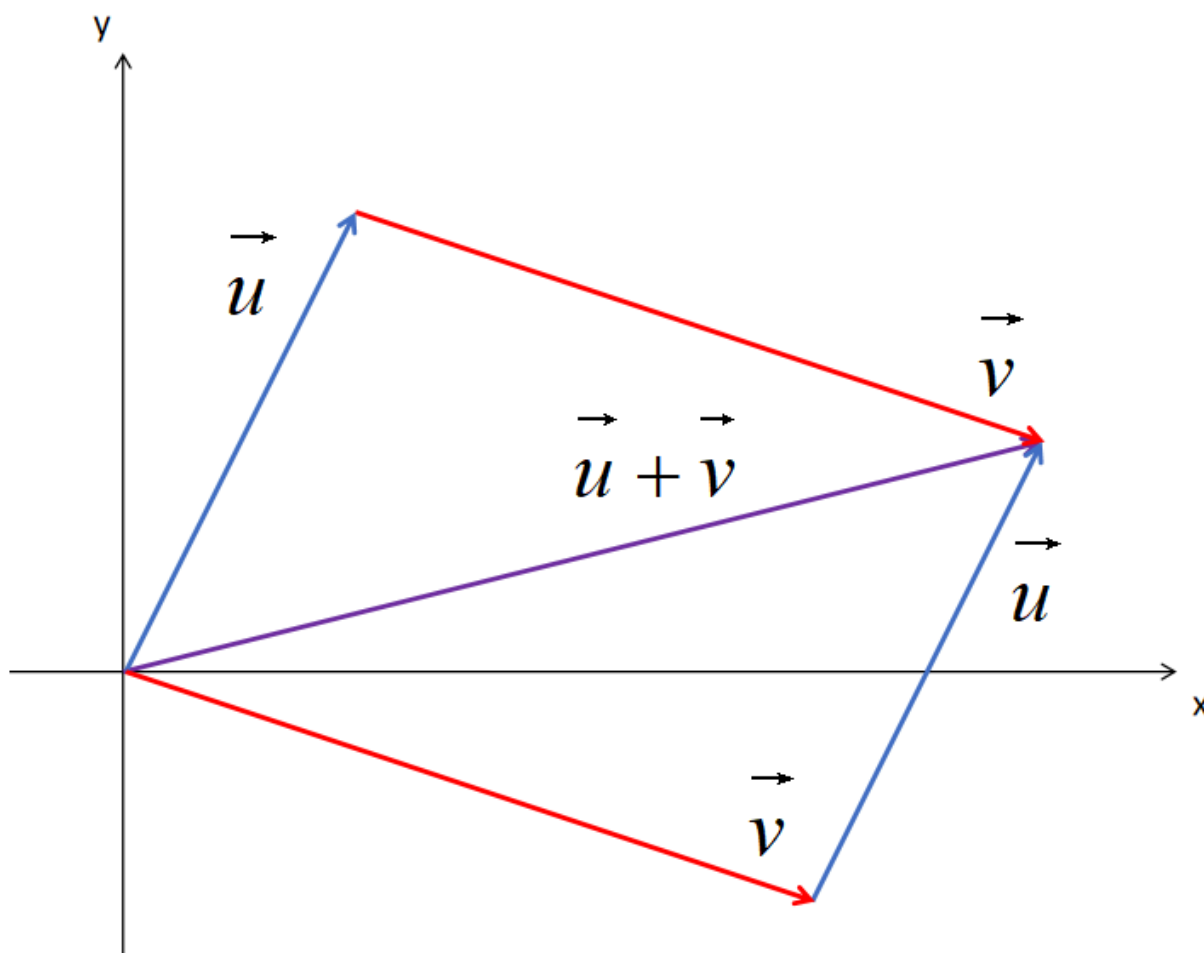


Giả sử ta có 2 vector $\vec{u}(x_1, y_1)$ và $\vec{v}(x_2, y_2)$, tổng \vec{u} và \vec{v} được tính bằng công thức:

$$\vec{u} + \vec{v} = (x_1 + x_2, y_1 + y_2).$$



Lưu ý: Thứ tự cộng các vector không quan trọng, cũng giống như phép cộng trên số (tính giao hoán).



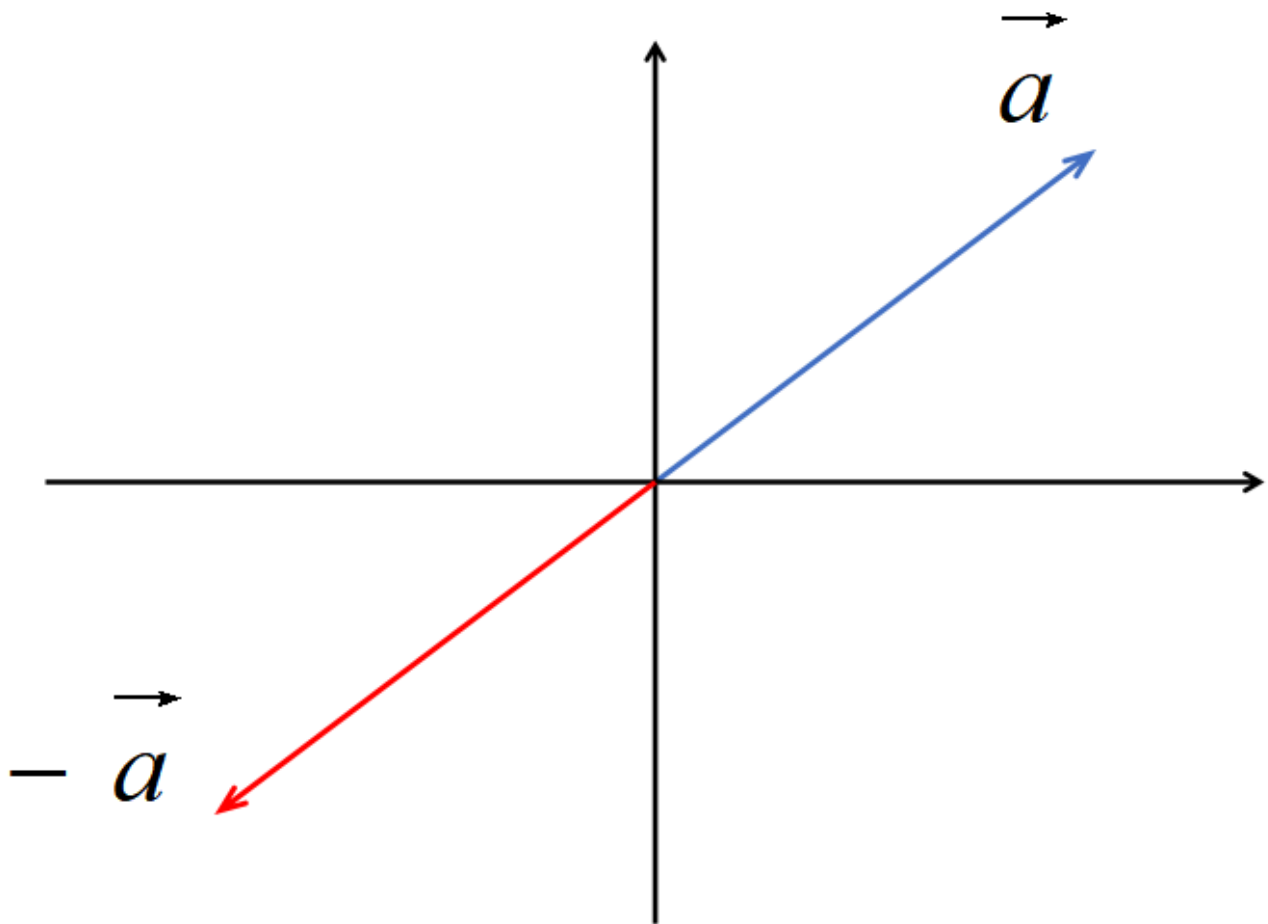
Phép trừ 2 vector



Vector đối

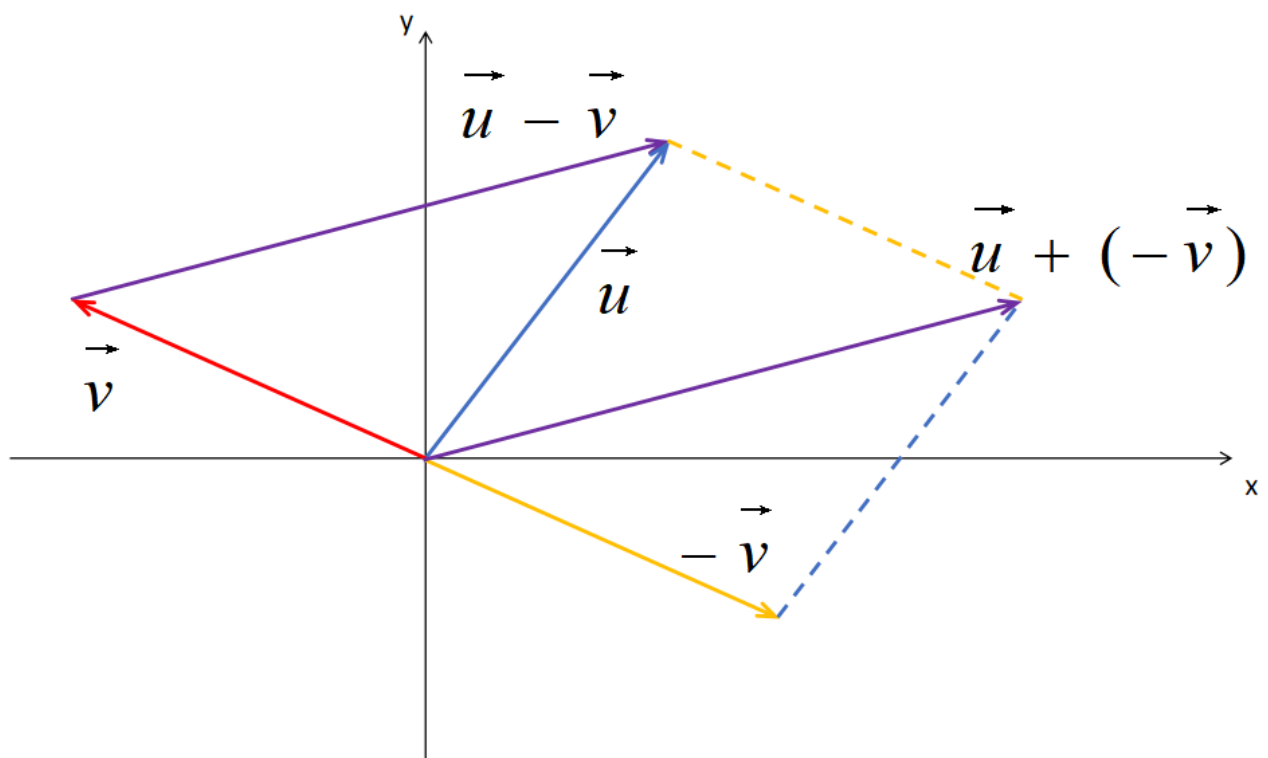
Cho vector \vec{a} , vector có cùng độ lớn và ngược hướng với vector \vec{a} được gọi là vector đối của vector \vec{a} , kí hiệu là $-\vec{a}$.

Mỗi vector đều có vector đối, chẳng hạn vector đối của \overrightarrow{AB} là \overrightarrow{BA} , nghĩa là $-\overrightarrow{AB} = \overrightarrow{BA}$.



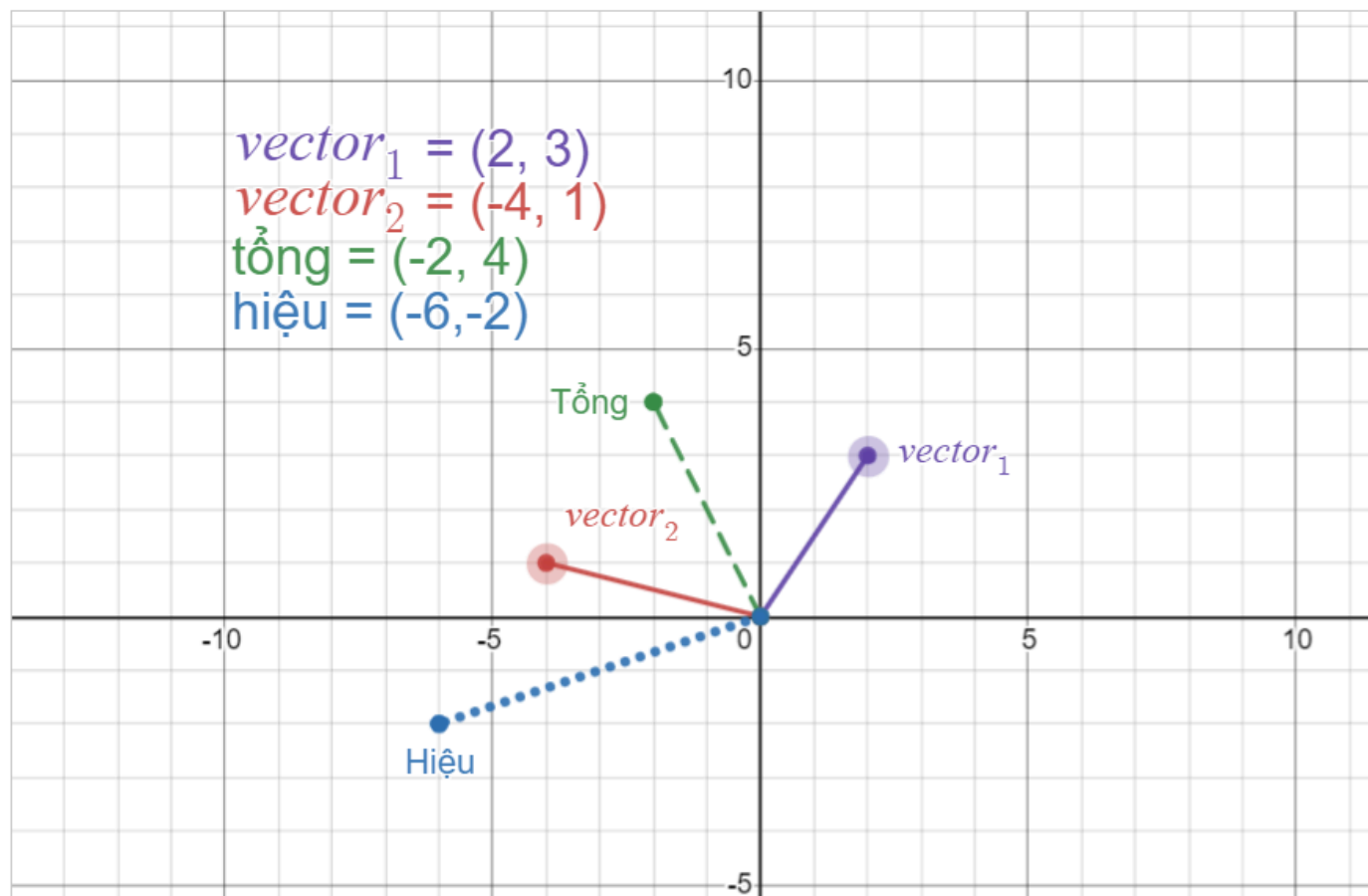
Định nghĩa hiệu của 2 vector

Hiệu của vector \vec{u} với vector \vec{v} chính là tổng của \vec{u} và $-\vec{v}$.



Nếu 2 vector có chung điểm đầu thì vector hiệu có hướng từ điểm cuối của \vec{v} đến điểm cuối của \vec{u} . **Ví dụ:**
 $\vec{OA} - \vec{OB} = \vec{BA}$.

Nếu 2 vector có chung điểm cuối thì vector hiệu có hướng từ điểm đầu của \vec{u} đến điểm đầu của \vec{v} . **Ví dụ:**
 $\vec{AO} - \vec{BO} = \vec{AB}$.



Nhấn vào [đây](#) để tương tác với hình trên Desmos.

Tích vô hướng (Dot product)

Không như phép cộng trừ vector là tương đối trực quan và dễ hiểu, vector có 2 phép toán kém trực quan hơn là tích vô hướng (dot product) và tích có hướng (cross product).

Tích vô hướng có thể được định nghĩa bằng đại số hoặc hình học. 2 định nghĩa này là tương đương khi sử dụng tọa độ Descartes.

- ▶ Theo đại số, tích vô hướng là tổng các tích tọa độ tương ứng giữa chúng. Ví dụ: tích vô hướng của $\vec{u}(x_1, y_1)$ và $\vec{v}(x_2, y_2)$ là $\vec{u} \cdot \vec{v} = x_1x_2 + y_1y_2$.
- ▶ Theo hình học, tích vô hướng là tích độ lớn của 2 vector và cos của góc giữa chúng. Ví dụ: tích vô hướng của $\vec{u}(x_1, y_1)$ và $\vec{v}(x_2, y_2)$ là

$$\begin{aligned}
 \vec{u} \cdot \vec{v} &= |\vec{u}| \cdot |\vec{v}| \cdot \cos(\theta) \\
 &= 13 \cdot 10 \cdot \cos(59,49^\circ) \\
 &= 65.9995359254 \dots \approx 66
 \end{aligned}$$

Từ 2 định nghĩa, ta có thể tính góc θ giữa $\vec{u}(x_1, y_1)$ và $\vec{v}(x_2, y_2)$ như sau:

$$\cos(\theta) = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}| \cdot |\vec{v}|} \implies \theta = \arccos\left(\frac{\vec{u} \cdot \vec{v}}{|\vec{u}| \cdot |\vec{v}|}\right) = \arccos\left(\frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} \cdot \sqrt{x_2^2 + y_2^2}}\right)$$

Lưu ý: tích vô hướng không chỉ giới hạn trong hình học phẳng, nghĩa là ta có thể sử dụng tích vô hướng cho các vector có số chiều tùy ý, và đẳng thức trên vẫn đúng.

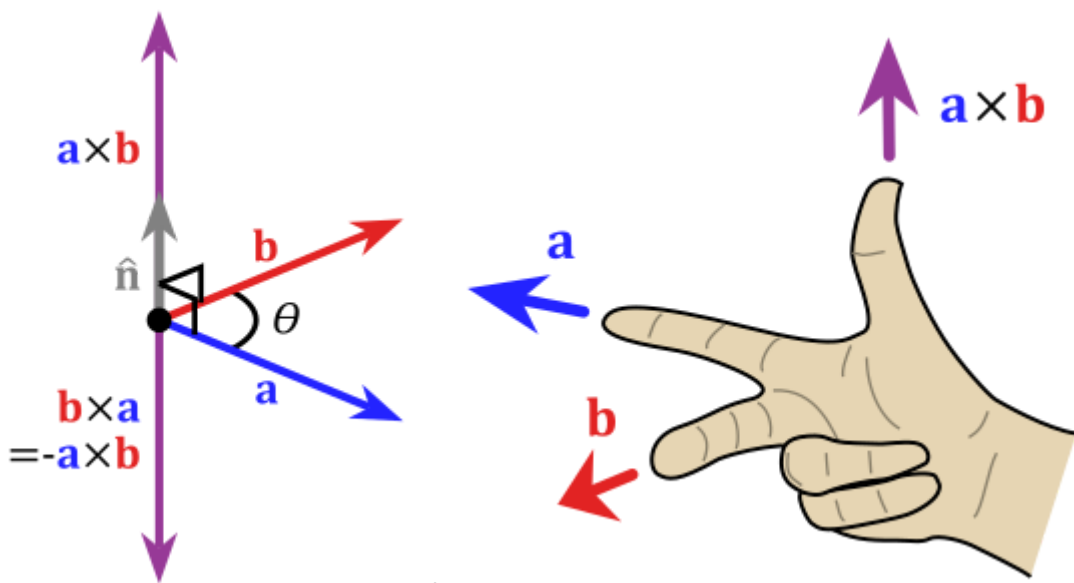
Tích có hướng (Cross product)

Trong không gian 3 chiều

Tích có hướng là một phép nhân vector trong không gian ba chiều. Nó khác tích vô hướng ở chỗ kết quả thu được là một vector thay cho một vô hướng. Vector này **vuông góc** với mặt phẳng chứa 2 vector đầu vào của phép nhân.

Tích có hướng được định nghĩa bằng công thức: $\vec{a} \times \vec{b} = \vec{n} \cdot |\vec{a}| \cdot |\vec{b}| \cdot \sin(\theta)$ với:

- θ là góc giữa \vec{a} và \vec{b} ($0^\circ \leq \theta \leq 180^\circ$)
- \vec{n} là vector đơn vị vuông góc với \vec{a} và \vec{b} . Thực tế có 2 vector thỏa điều kiện vuông góc là \vec{n} và $-\vec{n}$, do đó hướng của vector đơn vị \vec{n} phụ thuộc vào quy tắc bàn tay phải.



Trong không gian 2 chiều (mặt phẳng)

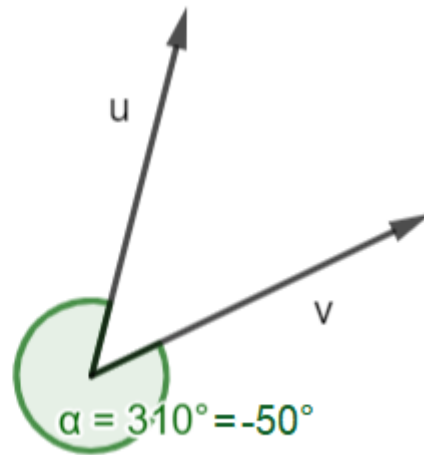
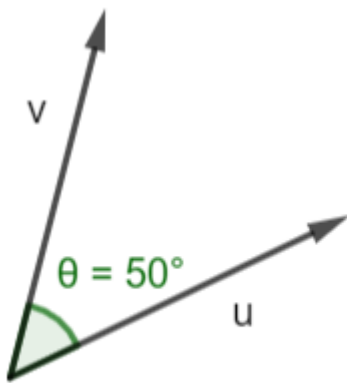
Nếu xét trong hình học phẳng thì vector kết quả lúc này vuông góc và có hướng đi vào/ra mặt phẳng đang xét, do đó ta có thể bỏ qua đặc điểm về hướng, và sử dụng tích có hướng như là một đại lượng vô hướng.

Tương tự tích vô hướng, tích có hướng trong không gian 2 chiều cũng có thể được định nghĩa bằng 2 cách:

- Theo đại số, tích có hướng giữa 2 vector $\vec{u}(x_1, y_1)$ và $\vec{v}(x_2, y_2)$ được định nghĩa bằng công thức:

$$\vec{u} \times \vec{v} = \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} = x_1 y_2 - x_2 y_1$$

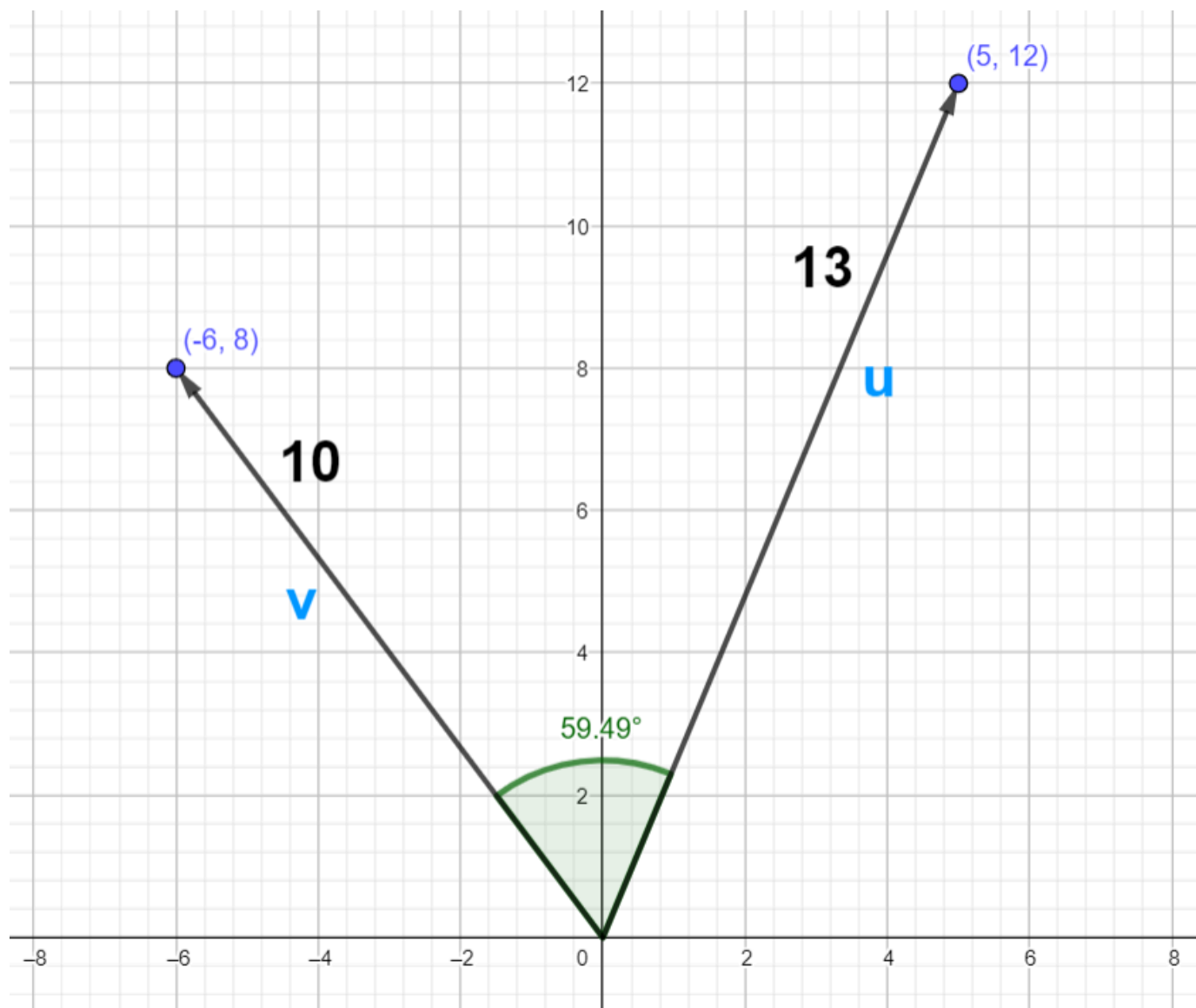
- Theo hình học, tích có hướng giữa 2 vector \vec{u} và \vec{v} được định nghĩa bằng công thức: $\vec{u} \times \vec{v} = |\vec{u}| \cdot |\vec{v}| \cdot \sin(\theta)$ với θ là góc hợp bởi 2 vector tính từ \vec{u} đến \vec{v} và **ngược chiều kim đồng hồ**. Với góc α thỏa mãn $0^\circ < \alpha < 180^\circ$ thì $\sin(\alpha) > 0$ nên nếu $\theta < 180^\circ$ thì tích có hướng **dương**, ngược lại tích có hướng **âm**.



Ta cũng có thể xác định dấu của tích có hướng bằng quy tắc bàn tay phải nhưng về bản chất thì cũng giống với việc xét góc theo chiều ngược kim đồng hồ.

Ví dụ

Tính tích có hướng của 2 vector $\vec{u}(5, 12)$ và $\vec{v}(-6, 8)$



$$\begin{aligned}\vec{u} \times \vec{v} &= \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} = x_1 y_2 - x_2 y_1 \\ &= 5 \cdot 8 - (-6) \cdot 12 \\ &= 112\end{aligned}$$

$$\begin{aligned}\vec{u} \times \vec{v} &= |\vec{u}| \cdot |\vec{v}| \cdot \sin(\theta) \\ &= 13 \cdot 10 \cdot \sin(59,49^\circ) \\ &= 112.000273471 \dots \approx 112\end{aligned}$$

Lưu ý: Một tích chất hữu dụng của tích có hướng trong hình học phẳng là

Mọi thứ trở nên phức tạp hơn một chút khi ta muốn tìm khoảng cách từ một đoạn thẳng đến một điểm. Trong trường hợp này, điểm gần nhất có thể là một trong hai đầu mút của đoạn thẳng thay vì là một điểm nào đó trên đường thẳng. Trong hình trên, điểm gần C nhất trên đường thẳng AB không nằm giữa A và B mà là tại B .

Có vài cách khác nhau để xử lý trường hợp này, một trong số đó là tích vô hướng. Đầu tiên, kiểm tra xem điểm gần nhất trên đường thẳng AB có ra khỏi B hay không bằng cách tính $\vec{BA} \cdot \vec{BC}$. Nếu tích này âm, nghĩa là

góc giữa \overrightarrow{BA} và \overrightarrow{BC} là góc tù (do với góc α thỏa mãn $90^\circ < \alpha < 270^\circ$ thì $\cos(\alpha) < 0$), do đó điểm gần C nhất trên đoạn AB sẽ là B .

Tương tự, nếu $\overrightarrow{AB} \cdot \overrightarrow{AC} < 0$, điểm gần C nhất là A . Nếu cả hai tích vô hướng đều ≥ 0 , thì điểm gần C nhất sẽ nằm giữa A và B .

```

1  #define x first
2  #define y second
3  typedef pair<int, int> pii;
4
5  // Compute the dot product AB · AC
6  int dot(pii A, pii B, pii C) {
7      pii AB, AC;
8      AB.x = B.x - A.x;
9      AB.y = B.y - A.y;
10     AC.x = C.x - A.x;
11     AC.y = C.y - A.y;
12     return AB.x * AC.x + AB.y * AC.y;
13 }
14
15 // Compute the cross product AB x AC
16 int cross(pii A, pii B, pii C) {
17     pii AB, AC;
18     AB.x = B.x - A.x;
19     AB.y = B.y - A.y;
20     AC.x = C.x - A.x;
21     AC.y = C.y - A.y;
22     return AB.x * AC.y - AB.y * AC.x;
23 }
24
25 // Compute the distance from A to B
26 double distance(pii A, pii B) {
27     int dx = A.x - B.x;
28     int dy = A.y - B.y;
29     return sqrt(dx * dx + dy * dy);
30 }
31
32 // Compute the distance from AB to C
33 // if isSegment is true, AB is a segment, not a line.
34 double linePointDist(pii A, pii B, pii C, bool isSegment) {
35     double dist = abs(cross(A, B, C)) / distance(A, B);
36     if (isSegment) {
37         int dot1 = dot(B, A, C);
38         if (dot1 < 0) return distance(B, C);
39         int dot2 = dot(A, B, C);
40         if (dot2 < 0) return distance(A, C);
41     }
42     return dist;
43 }
```

Đoạn code trên là cách mà mọi người thường dùng và có lẽ khá là dài, bên dưới là viết lại có sử dụng struct trong C++.



```

1  typedef double db;
2  struct vec {
3      db x, y;
4      vec(db _x = 0, db _y = 0) : x(_x), y(_y) {}
5      db dot(const vec &other) { // Compute the dot product
6          return x * other.x + y * other.y;
7      }
8      db cross(const vec &other) { // Compute the cross product
9          return x * other.y - y * other.x;
10     }
11     db length() const {
12         return sqrt(x * x + y * y);
13     }
14 };
15 using point = vec; // or use 'typedef vec point'
16 vec operator - (const point &B, const point &A) { // vecAB = B - A
17     return vec(B.x - A.x, B.y - A.y);
18 }
19
20 // if isSegment is true, AB is a segment, not a line.
21 db linePointDist(const point &A, const point &B, const point &C, bool isSegment) {
22     db dist = abs((B - A).cross(C - A)) / (A - B).length();
23     if (isSegment) {
24         db dot1 = (A - B).dot(C - B);
25         if (dot1 < 0) return (B - C).length();
26         db dot2 = (B - A).dot(C - A);
27         if (dot2 < 0) return (A - C).length();
28     }
29     return dist;
30 }


```

Nếu bạn sử dụng C++ thì bạn nên tìm hiểu về struct và tự viết class/struct geo_2D của riêng mình. Nó sẽ giúp các bài toán hình học trở nên đơn giản hơn nhiều.

Luyện tập

Học phải đi đôi với hành, do đó mình đề xuất cho các bạn [Codeforces Gym 100168](#) . Tuy đề bài trong gym được viết bằng tiếng Nga nhưng rất ngắn gọn và đi thẳng vào bài toán nên các bạn có thể dễ dàng [google translate](#) .

Bên dưới là một số bài tập có liên quan đến bài viết này, mình đã tóm tắt yêu cầu bài toán để các bạn có thể hiểu đề dễ dàng hơn.

- **Codeforces Gym - 100168L**: tính độ dài (độ lớn) vector
- **Codeforces Gym - 100168D**: tính diện tích tam giác
- **CSES - Polygon Area**  : tính diện tích đa giác
- **Codeforces Gym - 100168F**: tính khoảng cách từ 1 điểm đến 1 đường thẳng có dạng $Ax + By + C = 0$
- **Codeforces Gym - 100168G**: tính khoảng cách từ 1 điểm đến 1 đường thẳng đi qua 2 điểm
- **Codeforces Gym - 100168H**: tính khoảng cách từ 1 điểm đến 1 tia
- **Codeforces Gym - 100168I**: tính khoảng cách từ 1 điểm đến 1 đoạn thẳng

Được cung cấp bởi [Wiki.js](#)