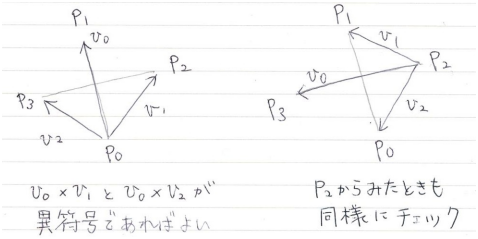
アルゴリズムと数学　演習問題集 037 - Intersection

問題ページへのリンク： 037 - Intersection[*1]

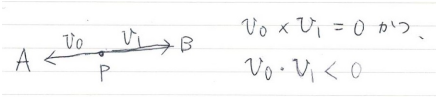
線分 p_0 - p_1 と線分 p_2 - p_3 が交わることを調べるには、点 p_0 から p_1 の向きを正面としたときに、p_2, p_3 が左右別々の方向にあることを確認する。これは、外積をもちいて確認できる。点 p_2 からも同様の確認をすれば、OK。



……と言いたいところだが、これだけではダメで、コーナーケースをいくつか考慮する必要があった。

- 線分の端点が、もう片方の線分と同一直線上にあるケース
  - 同一直線上だが、もう片方の線分上にはないケース
  - もう片方の線分のいずれかの端点と一致するケース
  - もう片方の線分の内部に含まれるケース
- 二つの線分の方向が同じケース
  - 二つの線分が平行で交わらない（同一直線上にはないケース）
  - 二つの線分が同一直線上にあり、共通部分を持たないケース
  - 二つの線分が同一直線上にあり、共通部分を持つケース

これらのうち、線分の端点がもう片方の線分の内部に乗っかるかどうかは、外積と内積の両方を使うことで検査できる（下図）。



さらに、二つの外積が同符号かどうかを確認する際に、掛けて正になるかで判定しようとすると桁あふれするおそれがあることにも注意が必要。

特に最後の件に気づいていなかったため、随分通すまでに苦労してしまった。

ちゃんと通せたコードが以下のもの：

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

using P = complex<ll>;
ll cross(P a, P b){ return a.real() * b.imag() - a.imag() * b.real(); }
```

[*1] https://atcoder.jp/contests/math-and-algorithm/tasks/math_and_algorithm_ah

```
ll dot(P a, P b){ return a.real() * b.real() + a.imag() * b.imag(); }

// 点 p が線分 a-b に含まれるかどうか
bool on_segment(P p, P a, P b)
{
  if (p == a || p == b) return true;
  P v0 = a - p;
  P v1 = b - p;
  // p が a-b と同一直線上      かつ p からみて a と b が反対側
  return (cross(v0, v1) == 0 && dot(v0, v1) < 0);
}

bool intersect(P p0, P p1, P q0, P q1)
{
  // 1 端点がもう片方の線分上にあれば true
  if (on_segment(p0, q0, q1) || on_segment(p1, q0, q1) ||
      on_segment(q0, p0, p1) || on_segment(q1, p0, p1)) return true;

  // 2. この時点で平行のときは false
  //（同一直線上で重なるケースは上の 1 で検出すみ）
  if (cross(p0 - p1, q0 - q1) == 0) return false;

  // 3. 以降は平行でないケース
  //   注意： c1 * c2 > 0 で同符号判定しようとすると桁あふれのおそれがある

  // 3.1 p からみたとき
  P v0 = p1 - p0;
  P v1 = q0 - p0;
  P v2 = q1 - p0;
  ll c1 = cross(v0, v1);
  ll c2 = cross(v0, v2);
  if ((c1 > 0 && c2 > 0) || (c1 < 0 && c2 < 0)) return false;

  // 3.2 q からみたとき
  v0 = q1 - q0;
  v1 = p0 - q0;
  v2 = p1 - q0;
  ll c3 = cross(v0, v1);
  ll c4 = cross(v0, v2);
  if ((c3 > 0 && c4 > 0) || (c3 < 0 && c4 < 0)) return false;
```

```
  return true;
}

int main()
{
  vector<P> p(4);
  for (int i = 0; i < 4; i++){
    ll x, y;
    cin >> x >> y;
    p[i] = P(x, y);
  }

  cout << (intersect(p[0], p[1], p[2], p[3]) ? "Yes" : "No") << endl;
  return 0;
}
```