

Tổng Minkowski

Tổng Minkowski của các bao lồi

Người viết:

- ▶ Đặng Đoàn Đức Trung - UT Austin

Reviewer:

- ▶ Nguyễn Hoàng Vũ - Trường Đại học Công nghệ - ĐHQGHN
- ▶ Trần Xuân Bách - Đại học Chicago (Mỹ)

Giới thiệu

Một số bài toán trung bình khó yêu cầu người giải phải thực hiện thao tác *ghép* hai bao lồi như sau: với hai bao lồi A và B , dựng một đa giác C chỉ bao gồm các điểm $a + b$ với $a \in A$ và $b \in B$ (ở đây một điểm thuộc bao lồi nếu nó nằm trong/trên bao lồi, và việc cộng điểm được thực hiện như việc cộng véctơ). Đây chính là việc tìm *tổng Minkowski* của hai bao lồi A và B .

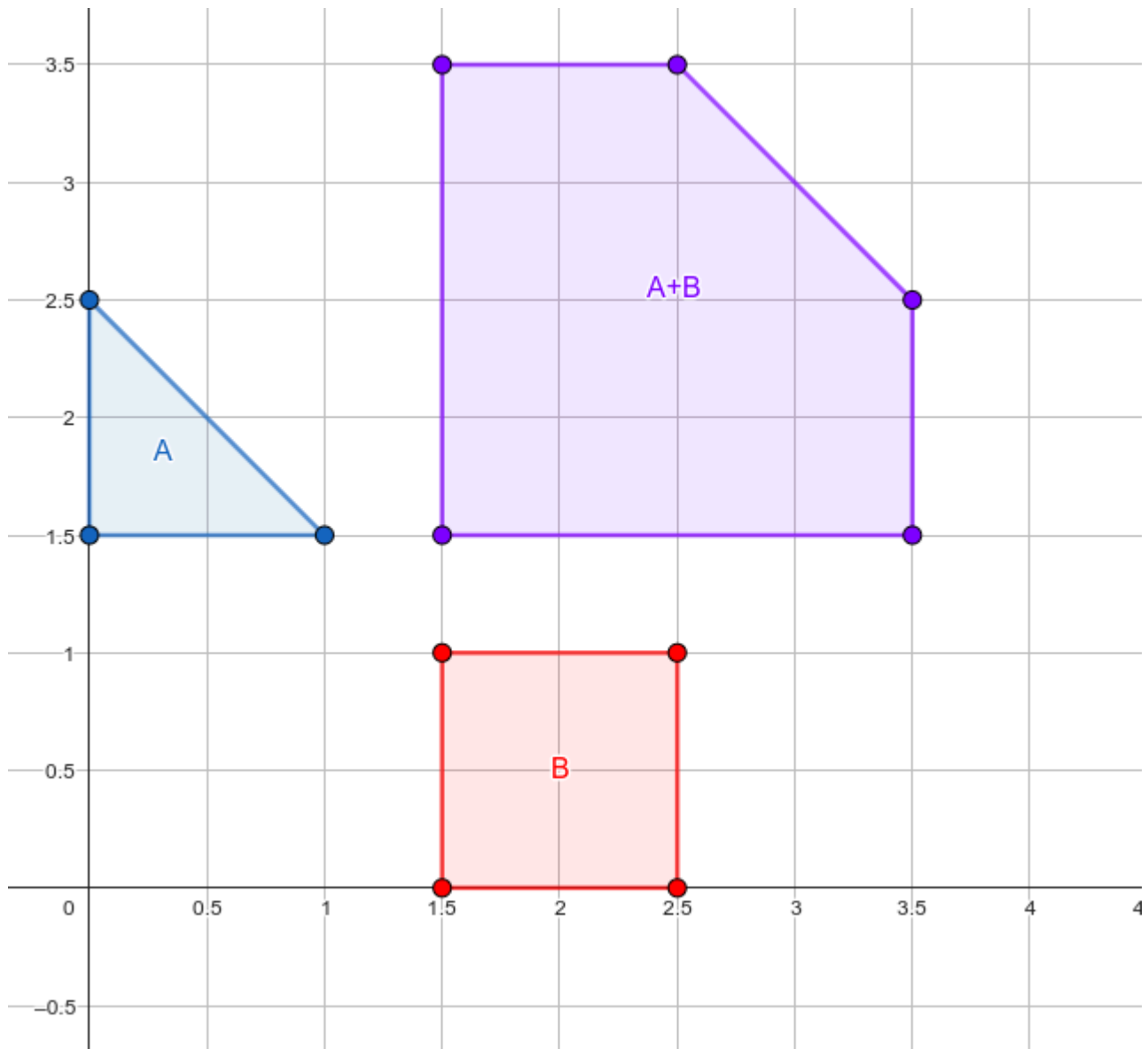
Ở bài viết sau, chúng ta hãy cùng xem qua cách tìm tổng Minkowski một cách hiệu quả, cũng như một số bài toán có thể được giải thông qua thuật toán này.

Bài viết này là phần 2 của chuỗi bài viết về các kĩ thuật xử lí bao lồi/hàm lồi; các bạn có thể đọc phần 1 [ở đây](#). Một phần của bài viết được tham khảo từ [cp-algorithm](#) [↗](#). Template code hình được lấy từ [kactl](#) [↗](#).

Định nghĩa

Để nhắc lại từ phần giới thiệu, ta định nghĩa tổng Minkowski $A + B$ của hai bao lồi A và B như sau:

- ▶ Với mọi điểm $c \in A + B$, tồn tại hai điểm $a \in A$ và $b \in B$ sao cho $a + b = c$.

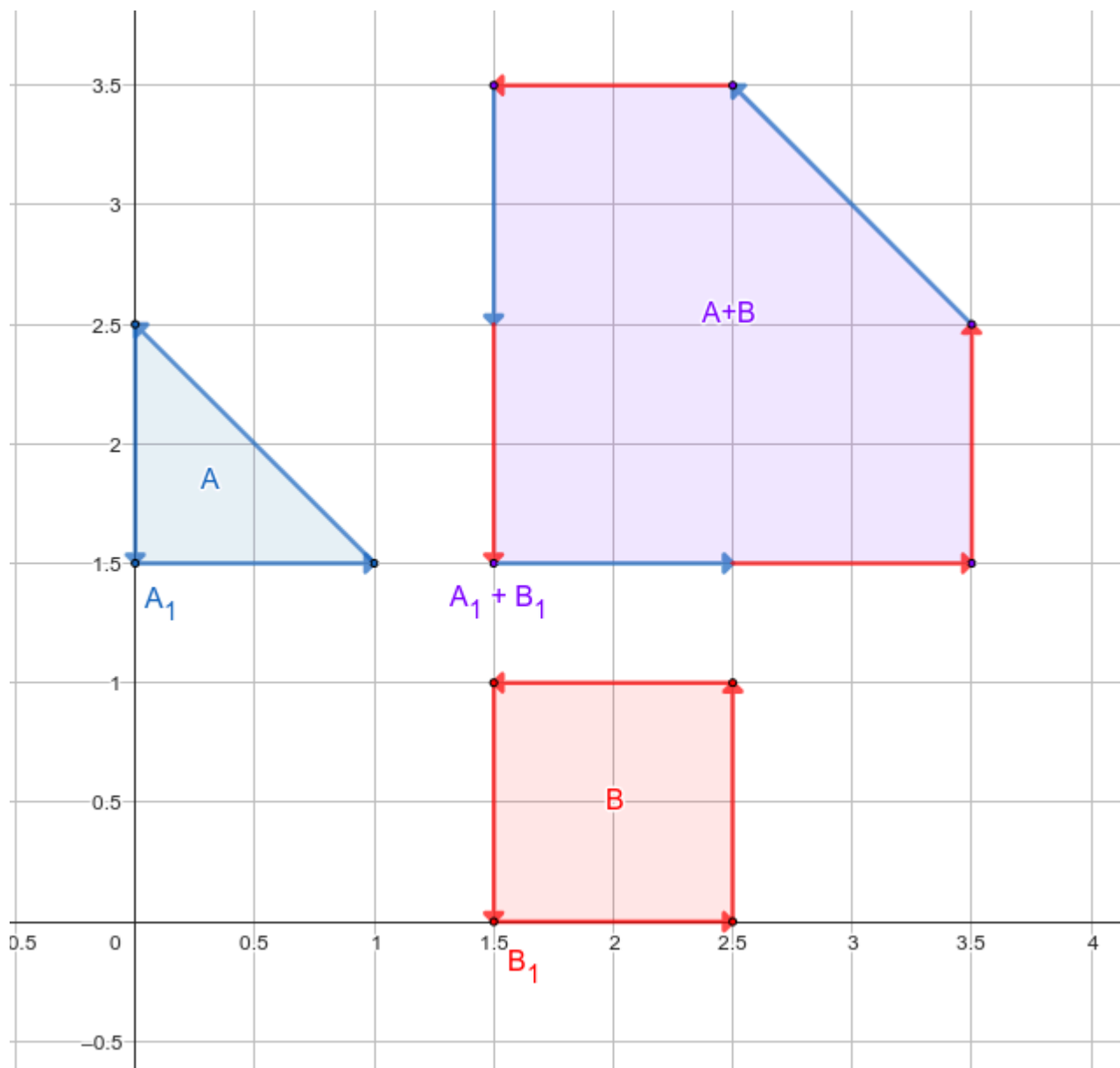


Ví dụ của tổng Minkowski của hai bao lồi A và B

Như ta thấy ở ví dụ trên, $A + B$ cũng là một bao lồi và có số lượng điểm là 5. Một cách tổng quát hơn, ta có thể chứng minh được là tổng $A + B$ cũng là đa giác lồi và có tối đa $|A| + |B|$ đỉnh.

Thuật toán tìm tổng Minkowski của hai bao lồi

Lấy ví dụ vừa nêu trên, ta có thể nhìn $A + B$ dưới một góc nhìn khác như sau.



Ta có hai nhận xét sau:

- Đỉnh trái dưới của $A + B$ bằng đỉnh trái dưới của A cộng đỉnh trái dưới của B .
- Cạnh của $A + B$ được là hợp của các cạnh từ A và các cạnh từ B .

Vì thế ta có thể dựng $A + B$ như sau (giả sử rằng A và B đi ngược chiều kim đồng hồ và được xoay lại sao cho A_1 và B_1 là đỉnh trái dưới của A và B):

- Xuất phát với $i = 1$ và $j = 1$. Thêm đỉnh $A_1 + B_1$ vào đáp án.
- Ở mỗi bước, so sánh góc của $\overrightarrow{A_i A_{i+1}}$ và $\overrightarrow{B_j B_{j+1}}$. Tăng con trỏ tương ứng với cạnh có góc nhỏ hơn theo hướng ngược chiều kim đồng hồ (tăng cả hai nếu góc của hai cạnh bằng nhau) và thêm $A_i + B_j$ vào đáp án.

Ta có thể coi thuật toán này như việc sắp xếp cạnh của hai bao lồi A và B : do cạnh của A và B đã được sắp xếp sẵn ngược chiều kim đồng hồ, ta chỉ cần chạy thuật toán ghép hai mảng đã được sắp xếp giống như bước ghép trong merge sort.

Dưới đây là một cách cài đặt mẫu:

```

1 // rút gọn từ kactl: https://github.com/kth-competitive-programming/kactl/blob
2 template<class T>
3 struct Point {
4     typedef Point P;
5     T x, y;
6     explicit Point(T x=0, T y=0) : x(x), y(y) {}
7     bool operator<(P p) const { return tie(x,y) < tie(p.x,p.y); }
8     bool operator==(P p) const { return tie(x,y)==tie(p.x,p.y); }
9     P operator+(P p) const { return P(x+p.x, y+p.y); }
10    T cross(P p) const { return x*p.y - y*p.x; }
11 };
12
13 template<class P>
14 vector<P> minkowskiSum(vector<P> a, vector<P> b) {
15     // xoay a và b sao cho điểm trái dưới là điểm đầu tiên
16     rotate(begin(a), min_element(begin(a), end(a)), end(a));
17     rotate(begin(b), min_element(begin(b), end(b)), end(b));
18     int n = a.size(), m = b.size();
19     vector<P> h(n + m + 1); h[0] = a[0] + b[0];
20     int t = 1;
21     // ở đây ta cho phép i đi tới n và j đi tới m để thể hiện việc đã duyệt qu
22     for (int i = 0, j = 0; i < n || j < m; ) {
23         if (i == n) j++;
24         else if (j == m) i++;
25         else {
26             P pa = a[(i + 1) % n] - a[i], pb = b[(j + 1) % m] - b[j];
27             auto cr = pa.cross(pb);
28             if (cr >= 0) i++;
29             if (cr <= 0) j++;
30         }
31         h[t++] = (a[i % n] + b[j % m]);
32     }
33     return {h.begin(), h.begin() + t - (t >= 2 && h[0] == h[t - 1])};
34 }

```

Độ phức tạp của thuật toán là $O(|A| + |B|)$.

Một số bài toán ví dụ

CF 87E - Mogohu-Rea Idol

Link bài: [CF 87E](#) .


Đề bài

Cho ba bao lồi A, B và C và q truy vấn. Với mỗi truy vấn, ta được nhận một điểm x , và ta cần trả lời rằng có tồn tại ba điểm $a \in A$, $b \in B$, và $c \in C$ sao cho điểm x là trọng tâm của tam giác được tạo bởi a, b, c ($1 \leq |A|, |B|, |C|, q \leq 10^5$).

Phân tích

Nhận xét rằng x là trọng tâm của tam giác tạo bởi a, b, c khi và chỉ khi $3x = a + b + c$. Nói cách khác, điểm $3x$ phải thuộc tổng Minkowski $A + B + C$. Vì thế, ta chỉ cần dựng trước $A + B + C$, và với mỗi truy vấn ta chỉ cần kiểm tra nếu điểm $3x$ thuộc bao lồi này không là được.

Cài đặt

Các bạn có thể tham khảo cách cài đặt [tại đây](#) . Ở phần cài đặt này, hàm `minkowskiSum` nhận một tập các bao lồi (không nhất thiết chỉ là 2 bao lồi) và trả về tổng Minkowski của tập bao lồi này. Độ phức tạp là $O(p + q \log p)$, với $p \leq |A| + |B| + |C|$ là số lượng điểm trong tổng Minkowski.

Tìm khoảng cách giữa hai bao lồi

Đề bài

Cho hai bao lồi A và B , tìm khoảng cách ngắn nhất giữa hai điểm bất kì thuộc hai bao lồi này. Nếu A và B có điểm chung thì in ra 0 ($1 \leq |A|, |B| \leq 2 \cdot 10^5$).

Phân tích

Đây là một bài toán kinh điển sử dụng tổng Minkowski. Nhận xét là bài toán có thể được viết như sau (ở đây $\|u\|_2 = \sqrt{x_u^2 + y_u^2}$ là khoảng cách của u tới gốc tọa độ):

$$\min_{a \in A, b \in B} \|a - b\|_2 = \min_{a \in A, b \in B} \|a + (-b)\|_2$$

Vì thế, nếu ta gọi $-B$ chứa tất cả các điểm $-b$ khi $b \in B$ thì bài toán tương đương với việc tìm $\min_{a \in A, c \in -B} \|a + c\|_2$. Gọi $S = A + (-B)$ là tổng Minkowski của bao lồi A và $-B$, thì bài toán trở thành: tìm điểm trong S gần gốc tọa độ nhất. Bài toán này ta có thể giải một cách dễ dàng: nếu S chứa gốc tọa độ thì đáp án là 0, ngược lại thì ta có thể lập qua cạnh của S và tìm khoảng cách ngắn nhất từ gốc tọa độ tới từng cạnh của S .

Cài đặt

Các bạn có thể tham khảo cách cài đặt dưới đây.

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  /// KACTL
5
6  #define rep(i, a, b) for(int i = a; i < (b); ++i)
7  #define all(x) begin(x), end(x)
8  #define sz(x) (int)(x).size()
9  typedef long long ll;
10 typedef pair<int, int> pii;
11 typedef vector<int> vi;
12
13 template <class T> int sgn(T x) { return (x > 0) - (x < 0); }
14 template<class T>
15 struct Point {
16     typedef Point P;
17     T x, y;
```

```

18     explicit Point(T x=0, T y=0) : x(x), y(y) {}
19     bool operator<(P p) const { return tie(x,y) < tie(p.x,p.y); }
20     bool operator==(P p) const { return tie(x,y)==tie(p.x,p.y); }
21     P operator+(P p) const { return P(x+p.x, y+p.y); }
22     P operator-(P p) const { return P(x-p.x, y-p.y); }
23     P operator*(T d) const { return P(x*d, y*d); }
24     P operator/(T d) const { return P(x/d, y/d); }
25     T dot(P p) const { return x*p.x + y*p.y; }
26     T cross(P p) const { return x*p.y - y*p.x; }
27     T cross(P a, P b) const { return (a-*this).cross(b-*this); }
28     T dist2() const { return x*x + y*y; }
29     double dist() const { return sqrt((double)dist2()); }
30     // angle to x-axis in interval [-pi, pi]
31     double angle() const { return atan2(y, x); }
32     P unit() const { return *this/dist(); } // makes dist()==1
33     P perp() const { return P(-y, x); } // rotates +90 degrees
34     P normal() const { return perp().unit(); }
35     // returns point rotated 'a' radians ccw around the origin
36     P rotate(double a) const {
37         return P(x*cos(a)-y*sin(a),x*sin(a)+y*cos(a)); }
38     friend ostream& operator<<(ostream& os, P p) {
39         return os << "(" << p.x << ", " << p.y << ")"; }
40 };
41
42 template<class P> bool onSegment(P s, P e, P p) {
43     return p.cross(s, e) == 0 && (s - p).dot(e - p) <= 0;
44 }
45
46 template<class P>
47 int sideOf(P s, P e, P p) { return sgn(s.cross(e, p)); }
48
49 using P = Point<double>;
50 bool inHull(const vector<P>& l, P p, bool strict = true) {
51     int a = 1, b = sz(l) - 1, r = !strict;
52     if (sz(l) < 3) return r && onSegment(l[0], l.back(), p);
53     if (sideOf(l[0], l[a], l[b]) > 0) swap(a, b);
54     if (sideOf(l[0], l[a], p) >= r || sideOf(l[0], l[b], p) <= -r)
55         return false;
56     while (abs(a - b) > 1) {
57         int c = (a + b) / 2;
58         (sideOf(l[0], l[c], p) > 0 ? b : a) = c;
59     }
60     return sgn(l[a].cross(l[b], p)) < r;
61 }
62
63 double segDist(P s, P e, P p) {
64     if (s==e) return (p-s).dist();
65     auto d = (e-s).dist2(), t = min(d,max(.0,(p-s).dot(e-s)));
66     return ((p-s)*d-(e-s)*t).dist()/d;
67 }
68

```





```

69  /// END KACTL
70
71  template<class P>
72  vector<P> minkowskiSum(vector<P> a, vector<P> b) {
73      rotate(begin(a), min_element(begin(a), end(a)), end(a));
74      rotate(begin(b), min_element(begin(b), end(b)), end(b));
75      int n = a.size(), m = b.size();
76      vector<P> h(n + m + 1); h[0] = a[0] + b[0];
77      int t = 1;
78      for (int i = 0, j = 0; i < n || j < m; ) {
79          if (i == n) j++;
80          else if (j == m) i++;
81          else {
82              P pa = a[(i + 1) % n] - a[i], pb = b[(j + 1) % m] - b[j];
83              auto cr = pa.cross(pb);
84              if (cr >= 0) i++;
85              if (cr <= 0) j++;
86          }
87          h[t++] = (a[i % n] + b[j % m]);
88      }
89      return {h.begin(), h.begin() + t - (t >= 2 && h[0] == h[t - 1])};
90  }
91
92  int main() {
93      ios_base::sync_with_stdio(false);
94      cin.tie(nullptr);
95      int n, m; cin >> n >> m;
96      vector<P> a(n), b(m);
97      vector<vector<P>> poly(n, {P(0, 0)});
98      for (int i = 0; i < n; i++) {
99          cin >> a[i].x >> a[i].y;
100      }
101      for (int i = 0; i < m; i++) {
102          cin >> b[i].x >> b[i].y;
103          b[i] = P(0, 0) - b[i]; // lật B qua gốc tọa độ
104      }
105      auto sum = minkowskiSum(a, b);
106      if (inHull(sum, P(0, 0), false)) {
107          cout << "0.0\n";
108      } else {
109          double ans = numeric_limits<double>::max();
110          int s = sum.size();
111          for (int i = 0; i < s; i++) {
112              ans = min(ans, segDist(sum[i], sum[(i + 1) % s], P(0, 0)));
113          }
114          cout << fixed << setprecision(10) << ans << '\n';
115      }
116  }

```

Độ phức tạp của thuật toán là $O(n + m)$.

Bài tập áp dụng

- [CF 1195F](#)  .
- [CF 1841F](#)  .
- [Discover Singapore 2019, 300iq Contest - I](#)  .
- [ICPC World Finals 2021 - K](#)  .

Được cung cấp bởi [Wiki.js](#)