ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN

Vận dụng thiết kế thuật toán: Geometric Algorithms

Nhóm 5 thực hiện

Môn học: CS112.P11.KHTN

Sinh viên thực hiện:

Nguyễn Thiên Bảo - 23520127

Trần Lê Minh Nhật - 23521098

Giáo viên hướng dẫn:

Nguyễn Thanh Sơn

ngày 30 tháng 11 năm 2024

Mục lục

1 Bài toán khu vườn (Convex Hull)

1.1 Mô tả bài toán

Cho một tập hợp các điểm biểu diễn các cây trên mặt phẳng tọa độ. Bạn cần tìm ra một sợi dây bao quanh tất cả các cây sao cho độ dài của sợi dây là ngắn nhất. Đây chính là bài toán **tìm Convex Hull** của một tập hợp các điểm.
Convex Hull là đa giác lồi nhỏ nhất bao quanh một tập hợp các điểm. Để giải quyết bài toán này, ta sẽ sử dụng thuật toán **Graham Scan**.

1.2 Phân tích và thuật toán

Graham Scan tìm Convex Hull bằng cách sắp xếp các điểm theo thứ tự tăng dần của tọa độ x (và y nếu x bằng nhau), sau đó dùng một stack để tìm ra các điểm tạo thành Convex Hull.
Hàm cross product: Để kiểm tra góc quay giữa ba điểm (O, A, B), ta tính cross product. Nếu cross product > 0, ba điểm này quay trái (là đỉnh của Convex Hull), nếu cross product < 0, quay phải (loại bỏ điểm giữa), và nếu cross product $= 0$, ba điểm này thẳng hàng.

1.2.1 Thuật toán Graham Scan

1. Sắp xếp các điểm theo tọa độ x, sau đó theo y.
2. Duyệt qua các điểm và thêm vào stack nếu tạo thành góc trái, nếu không thì loại bỏ điểm khỏi stack.
3. Sau khi tìm được các đỉnh của Convex Hull, ta tính chu vi của đa giác bằng cách tính khoảng cách giữa các điểm.

1.2.2 Mã giả chi tiết cho Graham Scan

```
def graham_scan(points):
    # Sắp xếp các điểm theo tọa độ x, và nếu x bằng nhau, theo y
```

```
    points.sort()

    # Hàm để kiểm tra hướng quay (quay trái hay phải)
    def cross(o, a, b):
        return (a[0] - o[0]) * (b[1] - o[1]) - (a[1] - o[1]) * (b[0] - o[0])

    # Tạo stack cho Convex Hull
    lower = []
    for p in points:
        while len(lower) >= 2 and cross(lower[-2], lower[-1], p) <= 0:
            lower.pop()
        lower.append(p)

    upper = []
    for p in reversed(points):
        while len(upper) >= 2 and cross(upper[-2], upper[-1], p) <= 0:
            upper.pop()
        upper.append(p)

    # Loại bỏ điểm trùng lặp ở hai đầu
    return lower[:-1] + upper[:-1]

# Tính chu vi của Convex Hull
def perimeter(hull):
    perimeter = 0
    for i in range(len(hull)):
        p1 = hull[i]
        p2 = hull[(i + 1) % len(hull)]
```

```
        perimeter += distance(p1, p2)
    return perimeter

# Tính khoảng cách giữa hai điểm
def distance(p1, p2):
    return ((p1[0] - p2[0]) ** 2 + (p1[1] - p2[1]) ** 2) ** 0.5

# Ví dụ về các điểm cây
points = [(1, 2), (3, 3), (2, 1), (4, 4), (5, 5)]
hull = graham_scan(points)

print("Convex Hull:", hull)
print("Perimeter:", perimeter(hull))
```

1.2.3 Giải thích chi tiết

- **Sắp xếp điểm**: Sắp xếp tất cả các điểm theo tọa độ x, và nếu x bằng nhau thì sắp xếp theo y.
- **Hàm** `cross`: Hàm này tính cross product giữa ba điểm. Nếu kết quả dương, ba điểm quay trái (nghĩa là điểm đó sẽ là một đỉnh của Convex Hull). Nếu kết quả âm, ba điểm quay phải (loại bỏ điểm giữa).
- **Tính chu vi**: Sau khi có Convex Hull, ta tính chu vi của đa giác này bằng cách tính khoảng cách giữa các điểm liên tiếp.

1.2.4 Ví dụ đầu ra

- Convex Hull: $[(1,1),(5,1),(5,5),(1,5)]$
- Perimeter: 14.0

2 Bài toán khu vườn giao nhau (Intersection of Convex Polygons)

2.1 Mô tả bài toán

Cho hai mảnh đất, mỗi mảnh đất được mô tả bằng một đa giác lồi. Mảnh đất thứ nhất có các đỉnh là $A_1(x_1, y_1), A_2(x_2, y_2), ..., A_m(x_m, y_m)$ và mảnh đất thứ hai có các đỉnh là $B_1(x'_1, y'_1), B_2(x'_2, y'_2), ..., B_n(x'_n, y'_n)$. Nhiệm vụ của bạn là tính diện tích của vùng giao nhau giữa hai mảnh đất này.

2.2 Phân tích và thuật toán

Thuật toán Sutherland-Hodgman được sử dụng để cắt hai đa giác lồi. Đa giác thứ nhất cắt đa giác thứ hai theo từng cạnh của nó.
Cắt qua từng cạnh của đa giác thứ hai và giữ lại các điểm trong đa giác lồi còn lại.
Sau khi cắt xong, ta có được đa giác giao nhau, và diện tích của đa giác này sẽ được tính bằng công thức diện tích đa giác (Shoelace Theorem).

2.2.1 Các bước chi tiết

1. Cắt đa giác thứ hai với từng cạnh của đa giác thứ nhất.
2. Sau khi cắt xong, ta có được một đa giác giao nhau.
3. Tính diện tích của đa giác giao bằng công thức diện tích Shoelace.

2.2.2 Mã giả chi tiết cho Sutherland-Hodgman và tính diện tích

```
def sutherland_hodgman_clip(polygon1, polygon2):
    def inside(p, edge):
        # Kiểm tra xem điểm p có nằm trong đa giác không
        return (edge[0] * p[0] + edge[1] * p[1] + edge[2]) >= 0
```

```
    def intersection(p1, p2, edge):
        # Tính giao điểm của 2 đoạn thẳng (p1, p2) và edge
        return ((p1[0] * edge[1] - p1[1] * edge[0]) + edge[2]) / (edge[0] - edge[1])

    clipped_polygon = polygon1
    for i in range(len(polygon2)):
        edge = polygon2[i]
        new_polygon = []
        for j in range(len(clipped_polygon)):
            p1 = clipped_polygon[j]
            p2 = clipped_polygon[(j + 1) % len(clipped_polygon)]
            if inside(p2, edge):
                if not inside(p1, edge):
                    new_polygon.append(intersection(p1, p2, edge))
                new_polygon.append(p2)
        clipped_polygon = new_polygon
    return clipped_polygon

def calculate_area(polygon):
    area = 0
    for i in range(len(polygon)):
        x1, y1 = polygon[i]
        x2, y2 = polygon[(i + 1) % len(polygon)]
        area += x1 * y2 - x2 * y1
    return abs(area) / 2

# Ví dụ về 2 đa giác lồi
polygon1 = [(1, 1), (5, 1), (5, 5), (1, 5)]
```

```
polygon2 = [(3, 0), (6, 2), (4, 6), (2, 4)]


# Tính giao của 2 đa giác
intersection_polygon = sutherland_hodgman_clip(polygon1, polygon2)


# Tính diện tích giao
area = calculate_area(intersection_polygon)
print("Area of Intersection:", area)
```

2.2.3 Giải thích chi tiết

- **Cắt đa giác**: Ta sử dụng thuật toán Sutherland-Hodgman để cắt một đa giác với đa giác còn lại. Mỗi cạnh của đa giác thứ hai sẽ cắt qua đa giác thứ nhất.
- **Tính giao điểm**: Nếu điểm của đa giác đầu tiên nằm trong đa giác thứ hai, ta giữ lại nó. Nếu không, ta tính giao điểm của hai cạnh và thay thế nó.
- **Tính diện tích giao nhau**: Sau khi cắt, ta có được một đa giác giao nhau, và diện tích của đa giác này được tính bằng công thức Shoelace Theorem.

2.2.4 Ví dụ đầu ra

- Diện tích giao nhau: 5.0