Lắp đặt cảm biến - Lời Giải

Nhóm 3

November 2025

Tóm tắt đề bài

Cho lưới $n \times n$. Có m cảm biến, cảm biến i đặt tại (x_i, y_i), chi phí c_i. Một cảm biến phủ ô (u, v) nếu $\max(|x_i - u|, |y_i - v|) \leq R$ (bán kính Chebyshev R). Hãy chọn một tập cảm biến sao cho phủ toàn bộ lưới và tổng chi phí nhỏ nhất.

- **Input**: một dòng n m R, sau đó m dòng (x_i y_i c_i).
- **Output**: dòng 1 in k (số cảm biến được chọn), dòng 2 in k chỉ số của cảm biến, dòng 3 là tổng chi phí.

1 Thuật toán (Heuristic ngẫu nhiên đơn giản)

Mỗi vòng lặp, ta thực hiện hai bước:

1. **Chọn một ô chưa phủ ngẫu nhiên**: lấy ngẫu nhiên một ô trên bảng chưa được phủ bởi các cảm biến đã chọn.
2. **Chọn cảm biến "phù hợp nhất" phủ được ô đó**: từ các cảm biến còn lại có phủ ô này, chọn cảm biến có tỉ lệ $\frac{\text{cost}}{\text{num of covered squares}}$ **nhỏ nhất**. Thêm cảm biến vào nghiệm và cập nhật tập ô đã phủ.

Lặp lại cho đến khi tất cả các ô đều được phủ hoặc không còn cảm biến nào có thể cải thiện.

Để có được kết quả tốt nhất, ta sẽ thực hiện thuật toán này nhiều lần và đặt cận thời gian, để đảm bảo có được kết quả tốt nhất trong giới hạn thời gian.

2 Mô tả chi tiết cách cài đặt

Gọi board là tập tất cả các ô, covered là tập đã phủ, và với mỗi cảm biến i, tiền xử lý tập cover[i] các ô nó phủ được. Ở mỗi bước:

- Lấy random_pos (random_pos $\in$ board random_pos $\notin$ covered).
- Trong các cảm biến còn lại có random_pos $\in$ cover$[i]$, tính $\text{gain}_i =$ cover$[i]$ và tỉ lệ $\rho_i = \dfrac{c_i}{\text{gain}_i}$. Chọn i có ρ_i nhỏ nhất (nếu hòa, ưu tiên chi phí nhỏ hơn).
- Thêm i vào tập chọn, cập nhật covered $\leftarrow$ covered $\cup$ cover$[i]$ và loại i khỏi danh sách ứng cử.

3 Độ phức tạp

Gọi $N = n^2$ là số ô. Tiền xử lý mỗi cảm biến xây tập phủ trong $O(R^2)$ (thực tế bị chặn bởi ô hình vuông $(2R{+}1)^2$ cắt với biên). Một vòng lặp xét tối đa m cảm biến còn lại, mỗi cảm biến ước lượng gain bằng phép trừ tập trong python trong $O(\min\{N, (2R{+}1)^2\})$. Với giới hạn nhỏ $n \le 50$, $m \le 100$, cách cài đặt tập Python là chấp nhận được. Khi đặt cận thời gian 2–3 giây, tổng thời gian thực tế vẫn rất nhanh.

4 Cài đặt (Python)

```python
# Nhom 3
import sys, time, random

cover = []
def get_cover(n, x, y, R):
    s = set()
    x1, x2 = max(1, x - R), min(n, x + R)
    y1, y2 = max(1, y - R), min(n, y + R)
    for i in range(x1, x2 + 1):
        for j in range(y1, y2 + 1):
            if max(abs(x - i), abs(y - j)) <= R:
                s.add((i, j))
    return s

def greedy_cover(n, sensors, R):
    board = set((i, j) for i in range(1, n + 1) for j in range(1, n + 1))
    cover_sets = []
    for idx, (x, y, c) in enumerate(sensors):
        cover_sets.append(cover[idx])

    covered = set()
    selected = []
    total_cost = 0
```

```
    while covered != board:
        best = None
        best_ratio = None
        random_pos = random.choice(list(board - covered))

        for st, cost, idx in cover_sets:
            if (random_pos not in st):
                continue
            new = len(st - covered)
            if new == 0:
                continue
            ratio = cost / new

            if best is None or ratio < best_ratio \
                or (abs(ratio-best_ratio) < 1e-12 and cost < best[1]):
                best = (st, cost, idx)
                best_ratio = ratio
        if best is None:
            break
        st, cost, idx = best

        selected.append(idx)
        total_cost += cost
        covered |= st
        cover_sets = [(_st, _cost, _idx) \
            for (_st, _cost, _idx) in cover_sets if _idx != idx]

    return selected, total_cost

def solve():
    input = sys.stdin.readline
    n, m, R = map(int, input().split())
    sensors = [tuple(map(int, input().split())) for _ in range(m)]

    for idx, (x, y, c) in enumerate(sensors):
        cover.append((get_cover(n, x, y, R), c, idx + 1))

    time_limit = 2.9
    start_time = time.time()
    best_selected = []
    best_cost = float('inf')
    found = False

    while time.time() - start_time < time_limit:
        selected, total_cost = greedy_cover(n, sensors, R)
```

```
        if total_cost < best_cost:
            best_selected = selected[:]
            best_cost = total_cost
            found = True

    selected_sorted = sorted(best_selected)
    if not found:
        print(0)
        print()
        print(-1)
        return

    print(len(selected_sorted))
    for x in selected_sorted:
        x += 1

    if selected_sorted:
        print(" ".join(map(str, selected_sorted)))
    else:
        print()
    print(best_cost)

if __name__ == "__main__":
    solve()

```