§ Homework 6: Greedy Methods §

Phần 1: Lý thuyết phương pháp

Phương pháp tham lam là một chiến lược giải quyết vấn đề bằng cách lựa chọn lựa chọn tốt nhất tại mỗi bước, không quan tâm đến tác động của quyết định đó đến các bước sau. Ý tưởng chính của phương pháp này là chọn giải pháp tốt nhất tại mỗi bước và hy vọng rằng việc lựa chọn này sẽ dẫn đến kết quả tốt nhất cho toàn bộ vấn đề.

Lược đồ tổng quát của phương pháp tham lam bao gồm các bước sau:

1. Xác định tất cả các lựa chọn có thể để giải quyết vấn đề.
2. Xác định hàm mục tiêu cần tối ưu.
3. Chọn lựa chọn tốt nhất tại mỗi bước.
4. Lặp lại cho đến khi đạt được giải pháp tối ưu.

Diagram:

```
Greedy (A) ≡     //Optimal solution by greedy on domain A
    S = ∅ ;
    while (A ≠ ∅ and !Is_solution(S))
        x = BestSelect(A);
        A = A \ {x};
        if (Acceptable(S, x))
            Integrate(S, x);
    endwhile;
   return S;
 End.
```

Ưu điểm của phương pháp tham lam là đơn giản và dễ hiểu, thường được áp dụng cho các vấn đề tối ưu hóa và giải quyết các vấn đề thực tế trong thời gian ngắn. Ngoài ra, phương pháp tham lam thường cho kết quả tốt trong những trường hợp khi các lựa chọn độc lập với nhau và không có sự tương tác giữa chúng.

Tuy nhiên, phương pháp tham lam có nhược điểm là có thể dẫn đến việc lựa chọn các giải pháp không tối ưu cho toàn bộ vấn đề. Điều này xảy ra khi quyết định tại mỗi bước được lựa chọn dựa trên thông tin có sẵn và không đánh giá được tác động của nó đến các bước sau này. Do đó, phương pháp tham lam thường không cho kết quả tốt khi các lựa chọn có sự phụ thuộc lẫn nhau hoặc các vấn đề phức tạp.

Phần 2: Bài tập lập trình

Flie code và đánh giá thuật toán trong folder ./src/Hw6/

Phần 3: Bài tập Anany book

9.1.11.

Cây bao trùm tối thiểu (Minimum Spanning Tree - MST) là một cây con của đồ thị vô hướng liên thông, có tổng trọng số của các cạnh là nhỏ nhất có thể. Tìm kiếm toàn diện (Brute-force Search) là phương pháp thường được sử dụng để tìm kiếm MST, tuy nhiên, phương pháp này sẽ gặp phải một số trở ngại sau:

+ Độ phức tạp tính toán: Với đồ thị có n đỉnh, số lượng cây bao trùm có thể có lên đến (n-2)^n, do đó việc áp dụng tìm kiếm toàn diện để tìm kiếm MST sẽ mất rất nhiều thời gian tính toán.

+ Tốn kém về chi phí: Tìm kiếm toàn diện yêu cầu phải duyệt qua tất cả các cây con của đồ thị, việc này tốn kém về chi phí bởi vì sẽ có rất nhiều lần tính toán trùng lặp.

+ Không hiệu quả: Khi đối với đồ thị lớn, phương pháp tìm kiếm toàn diện sẽ trở nên không hiệu quả do việc tính toán trên số lượng cây con quá lớn.

+ Không thực tế: Tìm kiếm toàn diện thường không được áp dụng trong thực tế do việc tìm kiếm MST đòi hỏi tính toán phức tạp, thời gian và chi phí cao.

Vì vậy, để xây dựng cây bao trùm tối thiểu, các thuật toán tối ưu hơn như Kruskal, Prim, Boruvka,.. đã được phát triển để giải quyết các trở ngại trên.

9.1.10.

Khái niệm cây khung nhỏ nhất (Minimum Spanning Tree - MST) là áp dụng cho đồ thị vô hướng có trọng số và không yêu cầu đồ thị phải liên thông.

Trong trường hợp đồ thị không liên thông, MST chỉ được xác định trên từng thành phần liên thông riêng biệt của đồ thị. Trong quá trình áp dụng thuật toán Prim hoặc Kruskal để tìm cây khung nhỏ nhất, các cạnh sẽ được thêm vào theo từng bước một, đồng thời kiểm tra tính kết nối của đồ thị. Khi tất cả các đỉnh đã được kết nối với nhau, cây khung nhỏ nhất đã được xác định.

Do đó, thuật toán Prim hoặc Kruskal đã bao gồm việc kiểm tra tính kết nối của đồ thị và tự động làm việc này trong quá trình xây dựng MST. Không cần phải kiểm tra khả năng kết nối của đồ thị trước khi áp dụng thuật toán.

9.2.7.

Ta chứng minh bằng phương pháp phản chứng (proof by contradiction).

Giả sử rằng cây khung được tìm bằng thuật toán Kruskal chứa một chu trình (cycle) C.

Vì C là chu trình nên nó phải chứa ít nhất một cạnh e không thuộc cây khung. Vì vậy, ta có thể loại bỏ cạnh e khỏi C để tạo thành một cây khung mới với trọng số nhỏ hơn.

Điều này mâu thuẫn với giả thiết của thuật toán Kruskal, vì Kruskal luôn chọn cạnh nhỏ nhất mà không tạo ra chu trình. Vì vậy, giả thiết ban đầu là sai và cây khung tìm được bởi Kruskal luôn không chứa chu trình.

9.2.9.

Thuật toán Prim và thuật toán Kruskal đều được sử dụng để tìm cây khung nhỏ nhất của một đồ thị liên thông có trọng số. Tuy nhiên, chúng có những khác biệt trong cách thực hiện và độ phức tạp thời gian và không gian của chúng.

Độ phức tạp thời gian:

+ Thuật toán Prim: Độ phức tạp thời gian của thuật toán Prim là O(E log V), với E là số lượng cạnh của đồ thị và V là số lượng đỉnh của đồ thị. Trong trường hợp đồ thị có tính chất đặc biệt, ví dụ như đồ thị hoàn toàn liên thông, độ phức tạp có thể giảm xuống đến O(E + V log V).

+ Thuật toán Kruskal: Độ phức tạp thời gian của thuật toán Kruskal là O(E log E), với E là số lượng cạnh của đồ thị. Trong trường hợp đồ thị có tính chất đặc biệt, ví dụ như đồ thị hoàn toàn liên thông, độ phức tạp có thể giảm xuống đến O(E log V).

Độ phức tạp không gian:

+ Thuật toán Prim: Độ phức tạp không gian của thuật toán Prim là O(V^2), với V là số lượng đỉnh của đồ thị. Trong trường hợp sử dụng heap để lưu trữ các cạnh, độ phức tạp không gian có thể giảm xuống đến O(E + V log V).

+ Thuật toán Kruskal: Độ phức tạp không gian của thuật toán Kruskal là O(E), do cần lưu trữ danh sách các cạnh và các tập hợp con.

Tóm lại, thuật toán Prim thường được sử dụng trong các đồ thị có số lượng đỉnh lớn hơn số lượng cạnh, trong khi thuật toán Kruskal thường được sử dụng trong các đồ thị có số lượng cạnh lớn hơn số lượng đỉnh. Ngoài ra, việc sử dụng heap để lưu trữ các cạnh trong thuật toán Prim có thể làm giảm độ phức tạp thời gian và không gian của thuật toán.