B.LỰA CHỌN 2 - Thuật toán được cung cấp trong bài giảng

Phần 1. Lý thuyết phương pháp

Phương pháp tham lam (Greedy Algorithm) là một phương pháp giải quyết vấn đề bằng cách chọn lựa giải pháp tốt nhất đến thời điểm hiện tại mà không quay lại xem xét các giải pháp đã được chọn trước đó.
Lược đồ tổng quát của phương pháp tham lam bao gồm:

1. Khởi tạo: chọn một giải pháp ban đầu.
2. Lựa chọn: chọn lựa giải pháp tốt nhất ở thời điểm hiện tại.
3. Kiểm tra: kiểm tra xem giải pháp đã chọn có phải là giải pháp tối ưu hay không.
4. Lặp lại: lặp lại bước 2 và 3 cho đến khi đạt được giải pháp tối ưu. .

Ưu điểm của phương pháp tham lam:

1. Tính đơn giản và dễ hiểu.
2. Tốc độ thực thi nhanh.
3. Phù hợp cho các bài toán có kích thước lớn.

Nhược điểm của phương pháp tham lam:

1. Không đảm bảo tìm được giải pháp tối ưu cho mọi trường hợp.
2. Có thể rơi vào vòng lặp vô hạn nếu không chọn được giải pháp tốt nhất ở mỗi bước.
3. Không thể quay lại các giải pháp đã chọn trước đó để tìm kiếm giải pháp tối ưu hơn.

Phần 2: Bài tập lập trình

1.Bài toán đổi tiền (change coins)

Changecoins.py

2.Bài toán xếp balo 0-1

Knapback.py

3.Bài toán lập lịch sử dụng tài nguyên

TaskScheduling.py

Phần 3: Bài tập Anany book

Bài 9.1.10: chúng ta có thể nói rằng chúng ta không cần phải kiểm tra tính liên thông của đồ thị trước khi áp dụng thuật toán Prim, bởi vì thuật toán đã được thiết kế để làm việc trên đồ thị liên thông và có thể tự động xác định cây khung nhỏ nhất của đồ thị đầu vào.
Bài 9.1.11: Xây dựng cây bao trùm tối thiểu bằng phương pháp tìm kiếm toàn bộ có thể gặp phải những khó khăn về mặt tính toán và khả thi cho các đồ thị lớn. Số lượng cây bao trùm có thể có trong một đồ thị có thể rất lớn và kiểm tra từng cây riêng lẻ sẽ đòi hỏi một lượng thời gian và tài nguyên tính toán đáng kể. Ngoài ra, thuật toán cần xem xét tất cả các kết hợp cạnh có thể, dẫn đến các vấn đề về sử dụng bộ nhớ và phức tạp. Do đó, tìm kiếm toàn bộ không phải là một phương pháp thực tế để xây dựng cây bao trùm tối thiểu trong hầu hết các trường hợp. Thay vào đó, các thuật toán hiệu quả hơn như thuật toán Prim hoặc Kruskal thường được sử dụng.
Bài 9.1.13: Khi không có trọng số trên các cạnh của đồ thị liên thông, ta có thể gán trọng số bằng 1 cho mỗi cạnh và áp dụng thuật toán Prim để tìm cây bao trùm. Điều này sẽ cho chúng ta một cây bao trùm tối thiểu vì tất cả các cạnh có trọng số bằng nhau.
Tuy vậy thuật toán prim không phải là thuật toán tối ưu trong trường hợp này
Bài 9.2.3: Ta có thể thực hiện các bước sau để tìm cây bao trùm tối thiểu cho một đồ thị bất kỳ sử dụng thuật toán Kruskal:

1. Tìm tất cả các thành phần liên thông của đồ thị ban đầu
2. Tạo một danh sách rỗng để lưu trữ các cây bao trùm tối thiểu.
3. Với mỗi thành phần liên thông, sử dụng thuật toán Kruskal để tìm cây bao trùm tối thiểu.
4. Thêm cây bao trùm tối thiểu của từng thành phần liên thông vào danh sách được tạo ở bước 2.
5. Trả về danh sách các cây bao trùm tối thiểu.

Bài 9.2.5:

1. Sắp xếp các cạnh của đồ thị theo thứ tự giảm dần của trọng số.
2. Khởi tạo một danh sách rỗng để lưu trữ các cạnh thuộc cây bao trùm tối đa.
3. Với mỗi cạnh của đồ thị theo thứ tự đã sắp xếp, kiểm tra xem hai đỉnh của cạnh có cùng hoặc khác thành phần liên thông với các đỉnh đã được thêm vào danh sách ở bước 2.Thêm vào cạnh có trọng số lớn nhất trong các cạnh chưa thêm vào cây và không tạo thành chu trình.
4. Lặp lại bước 3 cho đến khi ta có một cây bao trùm.