ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
UIT-HCM

Báo cáo cuối kỳ môn Phân tích và Thiết kế Thuật toán

Lớp: CS112.M11.KHTN

Nhóm 2: Nguyễn Duy Đạt & Lê Đoàn Phúc Minh

1. Computation Thinking

Abstraction -> Pattern recognition -> Decomposition -> Algorithm design -> Testing -> Evaluation

2. Kỹ thuật phân tích độ phức tạp thuật toán

a) Phương pháp tổng quát

+ Đánh giá giải thuật dựa trên tính đúng đắn, tính đơn giản và tính nhanh chóng (thời gian thực thi).

+ Đo thời gian thực hiện của chương trình dựa trên dữ liệu đầu vào, phụ thuộc vào tập lệnh của máy tính và kĩ năng người lập trình.

b) Phương pháp cụ thể thường sử dụng

+ Phương pháp Master Theorem và Akra-Bazzi:

-> Cả hai phương pháp đều giúp tính các hàm truy hồi, tuy nhiên điểm khác nhau là Akra-Bazzi có thể tính trong trường hợp hàm con có kích thước khác nhau còn Master Theorem thì không.

+ Phương pháp Aggregate:

-> Tính trung bình thời gian của 1 chuỗi các hàm, không quan tâm đến thời gian của 1 hàm, là 1 hướng tiếp cận độ phức tạp thời gian khấu trừ.

c) Các ký hiệu dùng trong phân tích độ phức tạp

+ Big Theta (Θ): Dùng cho phân loại thuật toán cấp tăng trưởng tiệm cận

+ Big Oh (O): Dùng cho xây dựng cận trên cấp tăng trưởng tiệm cận trên

+ Big Omega (Ω): Dùng cho xây dựng cận dưới cấp tăng trưởng tiềm cận dưới

3. Kiểm tra tính đúng và hiệu năng chương trình bằng bộ Test

Trong quá trình thử nghiệm chương trình, ngoài việc chương trình phải chạy đúng bộ test đã cho thì còn phải đáp ứng được việc chạy trong một khoảng thời gian nhất định với một lượng bộ nhớ giới hạn.

Việc thử nghiệm phần mềm được thực hiện bằng một số công cụ chuyên dụng.

Các thành phần của một trình chấm:

+ Trình sinh test

+ Lời giải 1 (Sử dụng phương pháp trâu bò để đảm bảo tính đúng đắn)

+ Lời giải 2 (Sử dụng giải pháp tối ưu của bài làm)

+ Trình so test

4. Tổng quan về các phương pháp thiết kế thuật toán

+ Brute Force: Đây là phương pháp thiết kế thuật toán cơ bản và đơn giản nhất trong số tất cả các phương pháp thiết kế phổ biến. Với phương pháp thiết kế này, chúng ta đơn giản chỉ là thử tất cả các phương án hợp lệ cho một bài toán cần được giải quyết cho đến khi tìm ra được phương án đúng hoặc không tìm thấy phương án.

+ Divide and Conquer (Chia để trị): Ý tưởng của phương pháp thiết kế này là chia bài toán mẹ thành các bài toán con với dữ liệu được phân chia ra cho chúng. Sau đó, giải quyết các bài toán con này một cách độc lập và gộp cái bài toán con này lại với nhau để tìm ra lời giải cuối cùng. Phương pháp thiết kế này thường được gặp trong một số thuật toán tối ưu như Merge Sort, Quick Select, FFT,….

+ Greedy Approach (Chiến thuật tham lam): Đây là phương pháp thiết kế thuật toán giải quyết bài toán bằng cách lựa chọn phương án tối ưu nhất ở mỗi bước đi nhằm tìm ra được phương án tối ưu nhất ở bước cuối cùng. Do tính chất tham lam của phương pháp thiết kế, việc thuật toán đưa ra kết quả tối ưu nhất là không đảm bảo. Phương pháp này thường bắt gặp trong một số thuật toán như Prim, Kruskal và Dijkstra.

+ Backtracking (Kỹ thuật quay lui): Phương pháp thiết kế thuật toán này được cải tiến từ phương pháp Brute Force. Trong quá trình đánh giá từng bước với phương pháp này, nếu như lựa chọn hiện tại không khả thi thì thuật toán sẽ quay lui về bước trước với lựa chọn khác.

+ Branch and Bound (Kỹ thuật nhánh cận): Tương tự như Backtracking, phương pháp này cũng được cải tiến từ Brute Force. Tuy nhiên, trong quá trình đánh giá từng bước với phương pháp này, nếu như cấu hình hiện tại không có kết quả khả quan so với cấu hình tốt nhất hiện thời thì thuật toán sẽ thử cấu hình khác và "bỏ rơi" ngay cấu hình hiện tại mà không cần phải tiếp tục xem xét nó.

+ Dynammic Programming (Quy hoạch động): Ý tưởng chính của phương pháp này là sử dụng phương pháp ghi nhớ để lưu lại kết quả của các trường hợp đã được xử lý nhằm tránh xử lý lại sau này. So với Chia để trị, phương pháp này sẽ xử lý tốt hơn đối với các bài toán có dữ liệu các bài toán con Overlap lên nhau. Tuy nhiên, trong trường hợp ngược lại thì phương pháp thiết kế này sẽ không phù hợp. Thuật toán tìm đường đi ngắn nhất Bellman-Ford và thuật toán tìm đoạn mảng có tổng lớn nhất Kadane là hai trong số những thuật toán có sử dụng phương pháp lập trình quy hoạch động.

5. Một số thuật toán và ứng dụng của đồ thị

+ Thuật toán Dijkstra: Tìm đường đi ngắn nhất từ 1 đỉnh đến các đỉnh còn lại của đồ thị có hướng không có cạnh mang trọng số không âm.

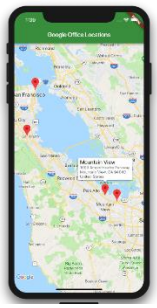
+ Thuật toán BFS/DFS: Tìm kiếm theo chiều rộng/chiều sâu của đồ thị.

+ Thuật toán Prim: Tìm cây khung nhỏ nhất của đồ thị vô hướng có trọng số liên thông.

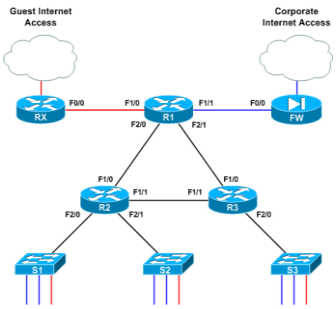
+ Thuật toán Ford-Fulkerson: Tính toán luồng cực đại trong 1 mạng vận tải.

+ Một số ứng dụng thực tế hằng ngày:

Google Map



Định tuyến mạng



Internet

6. Hình học và ứng dụng

+ Tích có hướng: kiểm tra 3 điểm có theo thứ tự kim đồng hồ hay không.

+ Hàm tính khoảng cách: Euclid, Manhattan, Minkowski, Chebyshev

+ Phân loại đa giác có thể dựa trên: độ lồi, số cạnh, tính đối xứng, sự hỗn hợp hình

+ Bài toán kiểm tra 1 điểm nằm trong hay ngoài đa giác: kiểm tra số giao điểm của tia từ điểm đó song song với trục hoành giao với cạnh của đa giác.

+ Thuật toán tìm bao lồi: Tìm 1 tập điểm nhỏ nhất mà tất cả các điểm đều nằm trong tập đó.

+ **Ứng dụng:**

-> Các khái niệm cơ bản về hình học là cơ sở xây dựng 1 số thuật toán Machine Learning như K-means, KNN, SVM

-> Thuật toán Affine trong đồ họa máy tính

-> Phân tích hệ thống thông tin địa lý

7. Kĩ năng điều hành thảo luận

+ Học hỏi được các nhóm có kiểu trình bày:

Làm Quiz -> Đưa ra vấn đề -> Tương tác liên tục trong thảo luận

+ Những yếu tố khác: Giọng nói, nội dung slide (không nên để nhiều chữ và màu font, nền phải tương thích).