ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Cấu trúc Rời rạc cho Khoa học Máy tính - CO1007

Báo cáo bài tập lớn

THUẬT TOÁN FORD-BELLMAN

Giảng viên hướng dẫn: Mai Xuân Toàn

Sinh viên thực hiện: 2352170 - Bùi Phan Khánh Duy

TP. Hồ Chí Minh, 08/2023

1 Giới thiệu

Bài toán người du lịch (TSM) được giải trên hai tập tin: `tsm.h` và `tsm.cpp`, tập tin đầu chứa khai báo nguyên mẫu hàm `Travelling(int n, int** C, char S)` với `n` là số đỉnh, `C` là ma trận trọng số và `S` là kí tự biểu thị đỉnh xuất phát. Tập tin tiếp theo chứa định nghĩa hàm `Travelling` và hàm `Try(int n, int** C, char S, int k)` với `k` là tham số gọi đệ quy quay lui.

Cho đến hiện tại vẫn chưa có thuật toán đủ nhanh (có độ phức tạp đa thức) cho bài toán này nên một trong những cách giải duy nhất của nó là liệt kê tất cả các chu trình có thể, sau đó tìm chu trình với chi phí ít nhất. Vì vậy, ta có thể sử dụng thuật toán quay lui. Ngoài ra, vì chuyến đi đi qua mỗi đỉnh đúng một lần sau đó trở về vị trí ban đầu, dãy biểu diễn các đỉnh trên đường đi sẽ là một hoán vị của n. Dễ thấy rằng bài toán trở về bài toán liệt kê hoán vị. Tuy nhiên, có tổng cộng $n!$ hoán vị, duyệt tất cả sẽ mất rất nhiều thời gian, vì thế ta sử dụng kĩ thuật nhánh cận để giảm bớt một số trường hợp đương nhiên không phải xét, làm tăng tốc độ của thuật toán.

2 Giải thích chi tiết

2.1 Liệt kê hoán vị, nhánh cận và tính chi phí

Hàm `Try` là hàm đệ quy đảm nhận việc liệt kê hoán vị. Ta truyền cho nó đầy đủ các tham số để hoạt động, ngoài ra còn có tham số k. Đầu tiên, ta sẽ duyệt qua toàn bộ các giá trị có thể, ở đây là từ đỉnh 0 (đỉnh A) đến đỉnh $n-1$, ta kiểm tra xem ta đã đi qua đỉnh đó chưa (nếu `used[i] = true` thì ta đã đi qua đỉnh i hay i nằm trong hoán vị), hoặc `C[h][i] > 0` (khi có đường đi trực tiếp từ h, đỉnh liền trước i trong hoán vị đến i), ta sẽ chọn đỉnh i là điểm đến thứ k trong chuyến đi. Nếu thỏa mãn toàn bộ các điều kiện, ta chọn i cho `perm[k]` với `perm` là mảng lưu hoán vị, đồng thời cộng `C[h][i]` cho chi phí hiện tại, đánh dấu `used[i] = true`, sau đó kiểm tra xem chúng ta đã hoàn thành hoán vị chưa (đủ tất cả các đỉnh), nếu đủ ta sẽ kiểm tra xem từ đỉnh cuối của hoán vị trở về đỉnh xuất phát, tổng chi phí có bé hơn chi phí tốt nhất chúng ta đã tìm thấy không, nếu bé hơn thì ta sẽ chọn đường đi mới này. Nếu vẫn chưa đủ hoán vị, ta tiếp tục chọn phần tử thứ $k+1$ cho hoán vị qua lời gọi `Try(n, C, S, k + 1)`, để thực hiện nhánh cận ta kiểm tra thêm xem chi phí hiện tại có lớn hơn chi phí tốt nhất ta tìm thấy chưa, nếu tốn hơn cả chi phí tốt nhất thì ta không phải tìm nữa, vì có đi nữa cũng chỉ tốn thêm. Trong trường hợp ngược lại, ta tiếp tục tìm đỉnh mới cho đến khi hoàn thành hoán vị, trở lại trường hợp trên. Cuối cùng, sau khi kết thúc một lượt đệ quy, ta phải bỏ đánh dấu

phần tử đã chọn, trả lại như trước để thử khả năng khác qua hai lời gọi `cost -= C[h][i]` và `used[i] = false`.

2.2 Lời gọi đệ quy và in kết quả

Hàm `Travelling` sẽ thay ta gọi hàm `Try` cùng các tham số và in ra kết quả. Vì đỉnh `S` đương nhiên nằm trong hoán vị, và còn là đỉnh đầu tiên, nên ta đánh dấu nó đã có mặt trong hoán vị tại vị trí đầu tiên. Sau đó ta gọi hàm `Try` để cập nhật đáp án cho mảng `bestPerm` và biến `bestCost` lần lượt lưu đường đi tốt nhất và tổng chi phí của nó. Cuối cùng ta in ra đáp án, thuật toán kết thúc.