**BÁO CÁO TIẾN ĐỘ PROJECT I**

**Sinh viên thực hiện:** Lê Thạch Cương

**GVHD:** Nguyễn Khánh Phương

Tuần 3

Bài 1:

Thuật toán Dijkstra tìm đường đi ngắn nhất từ 1 đỉnh đến các đỉnh còn lại của đồ thị: gọi đường đi ngắn nhất từ đỉnh bắt đầu đến các đỉnh còn lại là tập d[] và giả sử đỉnh bắt đầu là s, ta gán d[s] = 0 nghĩa là khoảng cách từ đỉnh bắt đầu đến chính nó là 0 và d[i] (đường đi đến các đỉnh còn lại) = INT\_MAX. Vì đồ thị đầy đủ lên giữa 2 đỉnh luôn có trọng số w. Nếu (j là tập đỉnh kề với đỉnh i) d[j] > d[i] + w thì cập nhật lại d[j] = d[i] + w. Làm như vậy cho đến khi xét hết tất cả đỉnh của đồ thị. Ta có thêm 1 mảng p để lưu những đỉnh mà đường đi đi qua. Với thuật toán không sử dụng priority queue thì với mỗi đỉnh i đang xét thì chúng ta phải duyệt toàn bộ những đỉnh kề còn lại với nó lên làm cho thuật toán mất thời gian khi đồ thị đầu vào có số đỉnh lớn. Với thuật toán có sử dụng priority queue ta chỉ cần đẩy đỉnh đã xét (i) và d[i] vào trong queue lên thời gian xử lý với đồ thị có số đỉnh lớn vẫn chạy rất nhanh.

* Độ phức tạp thuật toán với từng cách cài đặt:

|  |  |
| --- | --- |
| Dijkstra 1 | Dijkstra 2 |
| O(N^3) | O(N^2) |

|  |  |  |
| --- | --- | --- |
| Số đỉnh n | Dijkstra 1 | Dijkstra 2 |
| 10^2 | 0.015 | 0 |
| 10^3 | 15.777 | 0.022 |
| 10^4 | 259.03(minute) | 1.662 |

Bài 2: Bài toán luồng cực đại trong mạng

Thuật toán Edmons Karp: Nếu tồn tại một đường đi từ source đến sink, với điều kiện tất cả các cung trên đường đi đó vẫn còn khả năng thông qua, thì gửi đi một luồng dọc theo đường đi đó. Sau đó tìm một đường đi khác và tiếp tục như vậy. Một đường đi còn khả năng thông qua là một đường đi mà luồng qua đó còn khả năng tăng thêm hay còn gọi là đường tăng luồng. Trong thuật toán Edmons Karp thì đường tăng luồng được chọn sẽ là đường có ít cung nhất (Tìm bằng thuật toán BFS).

Độ phức tạp tính toán trong bài toán là O(VE^2) với V là tập đỉnh và E là tập cạnh

Bài 3: Cặp ghép cực đại

Ví dụ: Có n người và có m công việc, việc cần làm là tìm ra số cặp người – công việc lớn nhất sao cho không có người nào làm nhiều hơn 1 công việc(mỗi người có thể làm được nhiều việc nhưng chỉ được chọn 1) và 1 công việc không được nhiều hơn 1 người làm.

Bài toán cặp ghép cực đại cũng tương tự như ví dụ trên nhưng được lý thuyết hóa thông qua đồ thị hai phía. Yêu cầu bài toán là làm sao tìm được một cặp ghép cực đại giữa đồ thị 2 phía.

Bài 4: Bài toán Parking

Trong một bãi đỗ xe có rất nhiều ô tô và các điểm đỗ, các ô tô được chọn nhiều điểm đỗ xe nhưng chỉ được dừng lại duy nhất tại 1 điểm đỗ. Bài toán được mô phỏng thông qua một mảng 2 chiều sau đây.

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | |  | | --- | | {"XXXXXXXXXXXXX",  "X......XPPX",  "XC...P.XPPX",  "X......X..X",  "X....C....X",  "XXXXXXXXXXXXX"} | | |

Ở đây X là tường, P là chỗ đậu xe, C là ô tô và . là những chỗ có thể di chuyển (Có thể đi xuyên qua bãi đỗ này để sang bãi đỗ khác)

Thông thường ô tô thường chọn bãi đỗ xe gần nhất để vào tuy nhiên nó sẽ làm tổng thời gian toàn bộ ô tô vào bãi trống lại lớn. Như ví dụ trên nếu ô tô C ở vị trí hàng 5 vào bãi đỗ gần nhất thì ô tô ở hàng 2 mất đến 11 đơn vị thời gian để vào bãi đỗ xe gần nhất. Nếu ô tô ở hàng 5 vào bãi đỗ gần nhất bên trái thì tổng đơn vị thời gian mất chỉ là 5 cho cả 2 xe.

Vấn đề bài toán đặt ra là hãy trả về đơn vị thời gian ít nhất để tất cả ô tô có thể vào chỗ đỗ xe (như ví dụ trên là 5).

Thuật toán cho bài toán:

* Duyệt toàn bộ mảng trên và trả về những vị trí có thể di chuyển.
* Tạo ra 1 mảng lưu trữ thời gian cần thiết để từng xe đi tới điểm đỗ
* Sử dụng 1 hàm để kết nối các xe với các điểm đỗ
* Sử dụng 1 hàm để tìm thời gian đi nhỏ nhất cho tất cả các xe