CS2040S: Data Structures and Algorithms

Recitation 7

Goals:

- Model problems using graphs
- Transform graphs to capture all problem states
- Identify and formulate shortest paths problems
- Modify SSSP algorithms or cast problems as standard SSSP problems
- Appreciate the representational power of graphs

Empire *State* of Delivery

In the USA, and most parts of the world, a right-hand traffic system is enforced (Singapore follows the left-hand traffic system). In the US, when the green light is active for drivers at an intersection, they must either proceed straight or make a left turn. If the red light is active, drivers must first come to a complete stop and should they wish to make a right turn, they may after giving way to oncoming vehicles and when it is safe to do so. This is known as the "right turn on red" rule which generally applies to most intersections across the US unless indicated otherwise (slip roads are not as prevalent in US as they are in Singapore). Therefore, it is generally the case that vehicles approaching a four-way intersection in the US may turn in all three directions (left, straight, right). However to address the Vehicle Routing Problem, delivery companies such as the United Parcel Service (UPS) found that minimizing left-turns in their vehicle routes saves them as much as 10m gallons of fuel (20,000 worth of tonnes carbon dioxide emissions) and improved delivery throughput by 350,000 each year, despite the longer routes [1]. The intention of such a strategy is simple: to cut down time spent at intersections waiting for the green light (which also wastes fuel) and to minimize risks of accidents due to turning.

It's summer break, and you are interning at *Manhattan Eats*, a hot food delivery startup operating in New York City. After significant complains of slow deliveries by customers, the company is determined to implement a vehicle routing algorithm rather than leave that to its drivers — the way things are currently run. Knowing that you took CS2040S, your manager tasked you to design a better algorithm for planning delivery routes. Following UPS' successful strategy, the company decides to follow a similar policy:

- Strictly no left turns policy: At every intersection, the vehicle may only either proceed straight or make a right turn (U-turns are entirely out of the question here!)

[1] The Conversation: "Why UPS drivers don't turn left and you probably shouldn't either"

- Minimal right turns policy: The number of right turns made in the entire route must be kept at a minimum and it does not matter what the total distance of the route is

Notice that whether a turn is left or right depends on the car's direction as it enters the intersection.

One notable characteristic, amongst many others, of Manhattan is how its roads are laid out to form an uniform grid where roads running North-South are known as *avenues* and roads running East-West are known as streets. Indeed, such a city layout is what makes vehicle routing all the more critical for the company's success — one wrong turn could mean a very dissatisfied customer! An example of Manhattan's intersection-filled layout is shown in Figure 1.

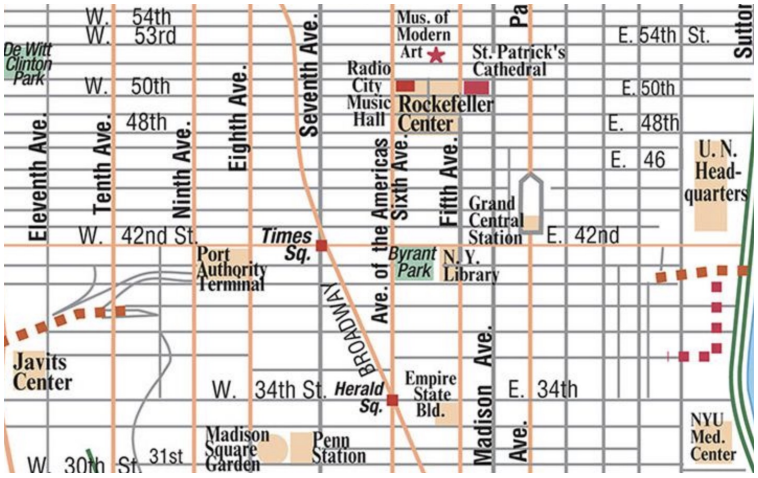

Figure 1: A partial map of Midtown Manhattan from aaccessmaps.

Problem 1. You are provided with a square grid map of Manhattan's streets, avenues and intersections. Given the destination intersection and the source intersection from where the vehicle will be setting off along with its initial direction (North, South, East, West), determine the best route for the driver according to your company's policy. The vehicle may only travel from intersection to intersection via streets and avenues.

For the sake of simplicity, you may assume all roads serve two-way traffic (this is no true in reality).

Single Player Shortest Victory

Player Unknown Battlegrounds (PUBG) is a popular online multiplayer battle royale game. In the classic solo mode of this game, the winner is the lone survivor. Players in a match are first airdropped into a free-for-all battlefield with no initial inventory. Weapons, ammunition, armours and tools are scattered across the battlefield. Once in a while, special supplies are dropped onto the battlefield at random locations. These special supplies contain powerful armaments and tools which grant players significant advantages in the game. As the game progresses, the play area also shrinks and eventually converge onto a small circle. Players who stay outside the play area suffer consistent damage until their characters perish. This is to force last survivors to come out of hiding and confront each other in a final battle.

Suppose you are currently playing a PUBG match on the popular Erangel map. As the match progresses into the final moments, you must strategically plan out your route to the final play area so that you may reach it in the least time. However, you are ill-equipped for the final battle and therefore have to visit at least one of the supply drops along the way. To help yourself find such an optimal route, you put on your CS2040S thinking cap and modelled the map as a graph in which the vertices are locations and the edge weight of an edge (u, v) represents the *travel time* from location u to v. This is depicted in Figure 2. Notice that the length of edges drawn do not correlate to their weight: some long distances entail short travelling times because vehicular transport is available while some short distances entail long travelling times because it involves climbing uphill.

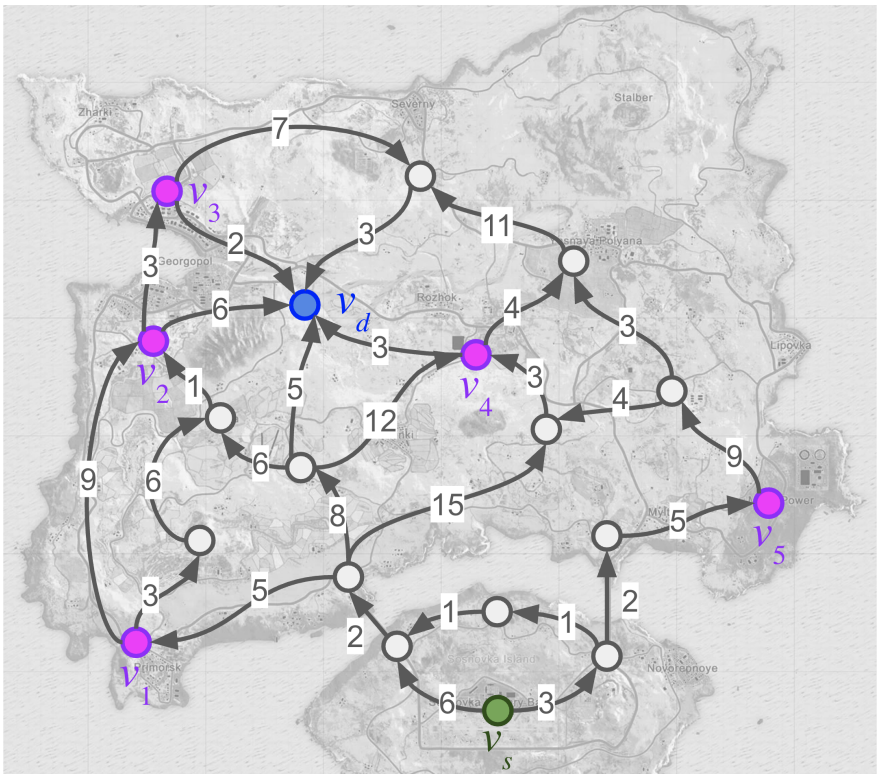

Figure 2: Travel times graph of Erangel map.

Problem 2. With reference to Figure 2, you have a *directed weighted* graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. Your current location is at v_s (green vertex) and the final play area is converging onto v_d (blue vertex). There are $v_1, v_2, \ldots, v_k$ supply drops scattered on the map (magenta vertices). Your goal is to find the *quickest* route starting from v_s and ending at v_d such that you visit *at least* one supply drop $v_i \in \{v_1, \ldots, v_k\}$.

StonksX Trader

One day you decided to get into the Foreign Exchange Market (Forex). You registered as a trader in the *StonksX* exchange, a popular exchange for global currencies. There are currently n currencies being traded in the exchange. To help you with analysis, you created a matrix R containing all exchange rates where $R[i, j]$ is the amount of currency j you can get for one unit of currency i. Note that exchange rates are not symmetric: $R[i, j] \neq R[j, i]$. Alas, there is "no arbitrage" possible in the *StonksX* exchange, meaning that if you start with one currency and then convert it to another, and then another, and so on, and then back to the first currency, you will end up with *no more* money then what you started with.

You begin the trading day with an account full of Singapore Dollars (SGD), and you want to end the day with only Great Britain Pounds (GBP). Find the sequence of conversions that will maximize the amount of GBPs you have at the end of the trading day.

Problem 3. How do you model this problem as a graph? What are its vertices and edges? Is it weighted or non-weighted? Are the edges directed or non-directed?

Problem 4. In your graphical model, which of its property reflects the "no arbitrage" rule? What would happen if such a rule is not enforced by the exchange?

Problem 5. What graph algorithm will you use to solve the problem? Which modifications are necessary?

Challenge question: How can you use the algorithm without modifying it (i.e. in the form that was presented to you in lecture)?