# Startup Tycoon

Algorithms with a Purpose

October 17, 2015

## 1 Introduction

Having recently graduated from X Combinator, your startup, Tuber, is struggling to disrupt the status quo in San Franhattan. According to Haxxor News, there's a growing demand for widgets in the city and being the street-smart entrepreneur you are, you decide to pivot your company to on-demand widget delivery. Using the $30 billion of venture capital, you decide to hire the best researchers out of the nearby Cranberry Melon University. However, you aren't alone – you will be joined by others who are also trying to disrupt this nascent market. To take over this niche in the market, your team must develop the best algorithm for delivering widgets! So get out there and deliver some widgets!

## 2 Terminology

| | |
|---|---|
| player | a single entity in a game, representing a team of up to 4 humans |
| game | an iteration of a set of orders and turns on a distinct map |
| turn | a single time step of a game during which an order may appear and a player may take an action |
| action | creating a station and/or responding to orders |
| order | a request for a widget originating from a home on the map |
| widget | the item being delivered by an order |
| home | a node on the map that can generate orders for widgets |
| station | a node on the map that can respond to orders for widgets |
| map | the map on which orders appear and actions occur, consisting of a collection of nodes, some of which are connected |

## 3 Rules

1. Every node can be reached from another through some path through the map. Nodes are connected with neighbors by **unweighted, undirected** edges.

2. Each station will be located at a **player-determined** node. The player may choose when to build their stations (including the first station), provided that they have enough capital.

3. Each turn will be capped at 0.5 seconds. Each game will end after **400** turns (a little over three minutes at most).

4. Orders will appear at certain time steps, according to a **Uniform Random Distribution** (in time) and will appear on the map a **Gaussian** distance away from randomly chosen hubs. (These hubs are unknown to the player)

   An order contains the following:

   - A destination node
   - A bounty for the request

5. You can respond to the request whenever you choose after receiving it (including the time step it was received) by returning a **set of nodes** from the station to the destination including the station and destination. Of course, once an order has timed out, you may no longer respond to it. The actual bounty received is a function of how long it takes for the widget to reach its destination (see Section 4: Scoring).

6. For a number of time steps equal to the length of the path returned, the edges on the path **cannot be used** for other requests. (Imagine your widget traveling through one edge per time step, during which the entire path is occupied while during transit.)

7. A **new station** can be built on any turn instantaneously and can begin serving on the same turn it is created as long as the order of the commands is correct. Stations can be built on any node in the map, including homes. The cost of each successive station will be higher than the previous one.

8. You can respond to more than one order on each turn if you wish.

## 4 Scoring

The scoring function for each game is the money earned by fulfilling orders, minus the cost of the stations built:

$$\sum_{n \in N} (s_n - t_n \times d) - \sum_{b \in B} c_b$$

$$\begin{aligned} N &= \text{set of fulfilled orders} \\ s_n &= \text{initial bounty of order } n \\ t_n &= \text{time steps elapsed in fulfilling order } n \\ d &= \text{decay factor (constant)} \\ B &= \text{set of stations built} \\ c_b &= \text{cost of building station } b \end{aligned}$$

The value of each order will decline linearly over time, and scores will reflect the time taken to deliver each fulfilled order. Note that continually unfulfilled orders will eventually time out upon reaching a value of 0.

## 5 Tournament

- The competition will take place in multiple rounds wherein each player's score is computed individually (i.e., there will not be multiple players competing simultaneously on the same map).
- The same map will be used for all players throughout a given round; however, a different map will be used for each successive round.
- The number of time steps may not be the same for all rounds, but will be defined in settings.py.
- 
- Scores will reset to 0 at the start of a round.
- The tournament winners shall be the top three players from the final round.

## 6 Getting Started

- READ THE README.md FILE!
- Read the code, namely:

  /src/game/player.py - You will implement your algorithm here.

  /src/game/state.py - All the state maintained for the game.

  /src/game/order.py - Representation of orders in the game.

  /src/game/settings.py - Constants and graphs used in the game.
- Familiarize your self with networkx. Start with how to get nodes, edges, and attributes of those nodes and edges. This documentation is linked to from the README.md.

## 7 Tips

- The practice maps will not be the same as the tournament maps, but maps will always be connected, unweighted, and undirected.
- As each map is a connected, unweighted, and undirected graph, graph algorithms — search and shortest-path in particular — will probably come in handy.
- You may also find it useful to look into and take ideas from various **streaming algorithms**. Wikipedia is a good place to start: `https://en.wikipedia.org/wiki/Streaming_algorithm`