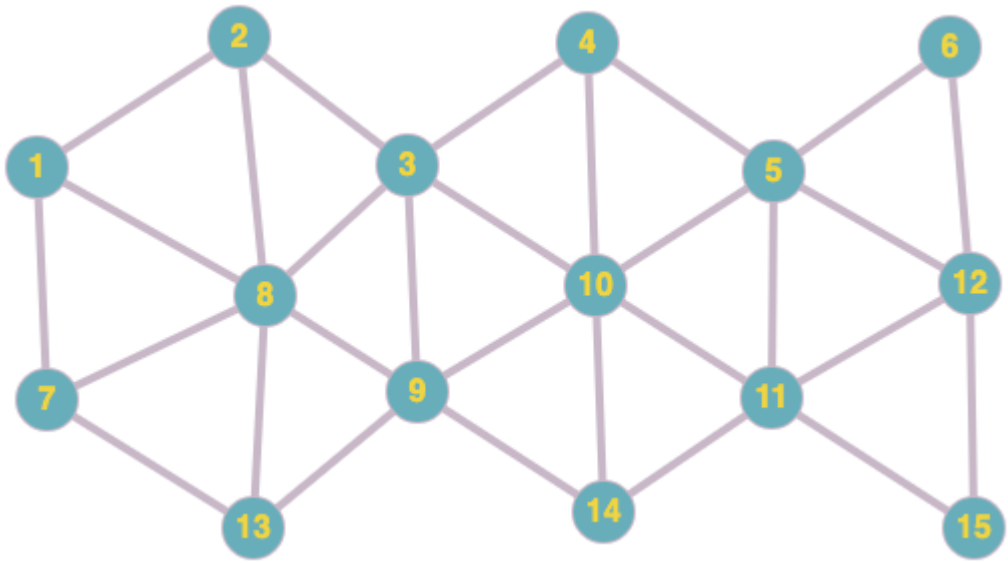
Hexagon Tiling Coverage: 15 Vertices

Charles Zhang

July 2

Environment




```
START = 1
V = 15
Reward = 100
alpha = 0.1
decay_gamma = 0.6    # discont factor
S = 1:V
Q = matrix(0, V, V)   # initialized VxV Q table
```

```
# adjacent matrix of the graph
Adj = cbind(c(0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0),
            c(1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0),
            c(0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0),
            c(0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0),
            c(0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0),
            c(0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0),
            c(1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0),
            c(1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0),
            c(0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0),
            c(0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0),
            c(0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1),
            c(0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1),
            c(0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0),
            c(0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0),
            c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0))
R = Adj-1
END = 7
```

```
R[c(1,8,13),END] = Reward
```

```
get_actions <- function(s) {
  a = c()
  for (i in 1:V) {
    if(R[s,i] != -1) a = c(a, i)
  }
  return(a)
}
```

```
tic()
rounds = 500
r = 1
while (r <= rounds) {
  s = sample(S, 1)   # random state
  while (TRUE) {
    action_space = get_actions(s)   # action space for S
    action <- sample(action_space, 1)   # random action to the next state
    s_next <- action    # next state S'
    actions_next = get_actions(s_next)   # action space for S'
    qs = c()
    for (i in actions_next) qs = c(Q[s_next,i], qs)   # list of all Q(S',a')
    # update by SARSA TD learning
    Q[s,action] <- Q[s,action]+ alpha*(R[s,action] + decay_gamma * max(qs)-Q[s,action])
    s = s_next    # update S
    if (s == END) break   # reach the final state
  }
  r <- r+1
}
```

Find Path based on Q table

```
path = c()
state = START
Q[Q == 0] <- 1000
while (length(path) < V)
{
  pre_state = state
  path = c(path, state)   # append the state
  state = match((min(Q[state,])), Q[state,])   # argmin Q(S, )
  Q[pre_state, ] = 1000     # clear the column and row of the appended state
  Q[, pre_state] = 1000     # by giving a large number
}
```

Running Time

```
toc()
```

```
## 0.463 sec elapsed
```

Coverage Path

```
path
```

```
##  [1]  1  2  3  4  5  6 12 15 11 10 14  9  8 13  7
```

Compare with previous GPS waypoints we got