STA 372-6 Homework 6: Dynamic Programming

Evan Johnston - eaj628

1. Code to find shortest path:

```
##################################################################
#Problem 1

# create weighted adjacency matrix
#       To: 1  2  3  4  5  6  7  8  9  10
# From: 1  0  0  3  2  0  0  0  0  0  0
#        2  0  0  4  2  4  0  0  0  0  0
#        3  0  0  0  0  0  3  2  0  0  0
#        4  0  0  0  0  0  4  0  2  5  0
#        5  0  0  0  0  0  0  2  2  0  0
#        6  0  0  0  0  0  0  0  0  0  3
#        7  0  0  0  0  0  0  0  0  0  4
#        8  0  0  0  0  0  0  0  0  0  2
#        9  0  0  0  0  0  0  0  0  0  3
#        10 0  0  0  0  0  0  0  0  0  0

# for simplicity, apply igraph package to aid in path calculation
require ("igraph")

adj_mat1<-matrix(c(0, 0, 3, 2, 0, 0, 0, 0, 0, 0,
                   0, 0, 4, 2, 4, 0, 0, 0, 0, 0,
                   0, 0, 0, 0, 0, 3, 2, 0, 0, 0,
                   0, 0, 0, 0, 0, 4, 0, 2, 5, 0,
                   0, 0, 0, 0, 0, 0, 2, 2, 0, 0,
                   0, 0, 0, 0, 0, 0, 0, 0, 0, 3,
                   0, 0, 0, 0, 0, 0, 0, 0, 0, 4,
                   0, 0, 0, 0, 0, 0, 0, 0, 0, 2,
                   0, 0, 0, 0, 0, 0, 0, 0, 0, 3,
                   0, 0, 0, 0, 0, 0, 0, 0, 0, 0), nrow=10, byrow=T)

# create graph object
network<-graph_from_adjacency_matrix(adj_mat1, mode="directed", weighted=T)
plot(network, layout=layout_nicely(network, dim = 2))

# all paths from 1 to 10
paths1<-all_simple_paths(network, 1, 10)

# shortest path from 1 to 10
spath1<-shortest_paths(network, 1, 10)
spath1
# weight of shortest path from 1 to 10
distances(network, 1, 10)

# all paths from 2 to 10
paths2<-all_simple_paths(network, 2, 10)

# shortest path from 2 to 10
spath2<-shortest_paths(network, 2, 10)
spath2
distances(network, 2, 10)
```

STA 372-6 Homework 6: Dynamic Programming

Evan Johnston - eaj628

Output from above code:

```
> # shortest path from 1 to 10
> spath1<-shortest_paths(network, 1, 10)
> spath1
$vpath
$vpath[[1]]
+ 4/10 vertices:
[1]  1  4  8 10
```

```
> # shortest path from 2 to 10
> spath2<-shortest_paths(network, 2, 10)
> spath2
$vpath
$vpath[[1]]
+ 4/10 vertices:
[1]  2  4  8 10
```

Thus the shortest path from 1 to 10 is 1→4→8→10 with a total degree of 6, and the shortest path from 2 to 10 is 2→4→8→10 also with a degree of 6.

2. **Dynamic approach to car valuation**
   a. States: time
   b. Actions: sell or operate
   c. Bellman Equation:

$$V(t) = \begin{cases} \min\{(-resale_t + operating\ cost_t), 0\} & if\ t = 6 \\ \min\{\ (-resale_t + operating\ cost_t), V(t+1)\} & if\ 0 < t < 6 \end{cases}$$

   d. Output from code:

Decisions	Sell	Sell	Sell	Sell	Sell	Operate
Values (Costs)	-\$13,400	-\$11,000	-\$6,400	-\$3,600	-\$800	0

Note: values of costs does not consider the original purchase price.

3. **Travelling Salesman Variation**
   a. Net gains from travelling matrix M:

		To:		
		Indianapolis	Bloomington	Chicago
From:	Indianapolis	\$120	\$110	\$150
	Bloomington	\$70	\$160	\$100
	Chicago	\$100	\$90	\$170

   b. States: (time, place) where time $t \in \{1,2,3,4\}$ and place $c \in \{1,2,3\}$ corresponding to Indianapolis, Bloomington, and Chicago, respectively
   c. Actions: choose place $c \in \{1,2,3\}$
   d. Bellman Equation:

STA 372-6 Homework 6: Dynamic Programming

Evan Johnston - eaj628

$$V(t,c) = \begin{cases} \max\{M_{2,1}, M_{2,2}, M_{2,3}\} & if\ t = 1 \\ \max\{M_{c(t-1),1}, M_{c(t-1),2}, M_{c(t-1),3}\} & if\ 1 < t < 4 \\ \max\{M_{c(t-1),1}\} & if\ t = 4 \end{cases}$$