---

1 Traveling Salesperson Problem (TSP)

1.1 Matrix D

For matrix D, the rows are indexed from v_1 to v_n. Columns are indexed by all subsets in the powerset of $V - \{v_1\}$, which is the set of all vertices without the first vertex.

Let A be a set such that A is included in the powerset of $V - \{v_1\}$. The element at $\mathrm{D}[v_i][A]$ should be the length of the shortest path from v_i back to v_1, passing through each vertex in A exactly once. For example, if $A = \{v_2\}$ and $v_i = v_4$, the element at $\mathrm{D}[v_4][v_2]$ should be the total length of $< v_4, v_2, v_1 >$.

1.2 Matrix P

The matrix P is the matrix which the optimal path can be reconstructed from. The element at $\mathrm{P}[v_i][A]$ will contain the next vertex after v_i in the optimal solution.

2 Example with Graph

We will be using this graph for the rest of the example.

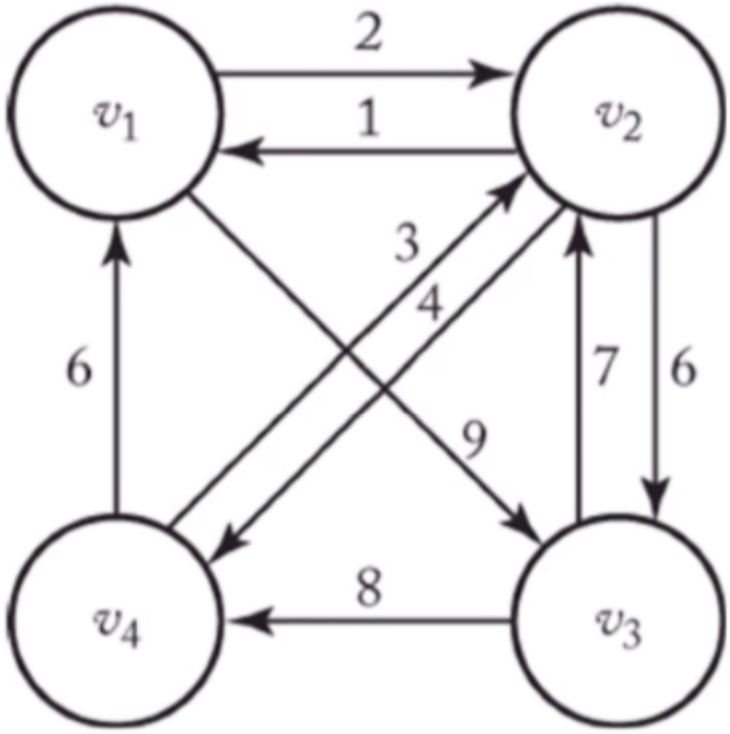

Figure 3.16

2.1 Matrix W

This is the weight matrix of the graph, calculated just as it was in Floyd's algorithm.

	v_1	v_2	v_3	v_4
v_1	0	2	9	∞
v_2	1	0	6	4
v_3	∞	7	0	8
v_4	6	3	∞	0

2.2 Matrix D

	$\{\}$*	$\{v_2\}$**	$\{v_3\}$**	$\{v_4\}$**	$\{v_2, v_3\}$***	$\{v_2, v_4\}$***	$\{v_3, v_4\}$***	$\{v_2, v_3, v_4\}$****
v_1	Don't consider v_1 because this is our starting/ending point.							21
v_2	1	X	∞	10	X	X	20	X
v_3	∞	8	X	14	X	12	X	X
v_4	6	4	∞	X	∞	X	X	X

If an X has been marked, it means we do not consider this position because v_i is being used as an intermediate vertex in A, and that is not possible. For example, D$[v_2][\{v_2\}]$ has an X because this position is asking for the path $< v_2, v_2, v_1 >$ and that is not possible since we only visit each vertex once.

2.2.1 How to Find Values of Matrix D

*For the null set/empty set({}):

This column of weights can be found simply by looking at the graph or the weight matrix W. This is because these values represent the path from v_i to v_1 using the null or empty set as intermediate vertices, or in other words, no intermediate vertices. For example, D[v_2][{}] is 1 because W[v_2][v_1] is 1. Another way to look at it would be to look at the graph and see the edge from v_2 to v_1 has the weight of 1.

In general: D[v_i][{}] = W[v_i][v_1]

For non-empty sets:

The values of these columns can be calculated by considering each vertex in A as the first intermediary vertex, v_j. From there, if there are anymore remaining vertices in $A - \{v_j\}$, use the column whose subset represents $A - \{v_j\}$ and use the value in the row that corresponds to v_j.

In general: D[v_i][A] = minimum(W[v_i][v_j] + D[v_j][$A - \{v_j\}$]

Examples:

**For sets of size 1:

Consider D[v_3][v_2].
The path that this position represents is $< v_3, v_2, v_1 >$.
We've already found the path $< v_2, v_1 >$, and it is being stored at D[v_2][{}].
We still need to find the path $< v_3, v_2 >$. We can find this either from looking at the graph or looking at the weight matrix W.
Looking at the graph, simply find the edge from v_3 and v_2 and observe its weight, 7.
Looking at the weight matrix W, look at W[v_3][v_2] and see that the value is also 7.
Add the two values together to get the total weight of the path $< v_3, v_2, v_1 >$, which is $7 + 1 = 8$.

***For sets of size 2:

Consider D[v_3][$\{v_2, v_4\}$].
The paths that this could represent are:
Path 1: $< v_3, v_2, v_4, v_1 >$
Path 2: $< v_3, v_4, v_2, v_1 >$
As you can see, the vertices after v_3 in the paths are the vertices from A, which is $\{v_2, v_4\}$. Path 1 starts with v_2 after v_3 and path 2 starts with v_4 after v_3. So, to generalize, each vertex in A will produce another possible path.

In order to calculate the total weight of path 1, we already have figured out $< v_2, v_4, v_1 >$. Looking it up on the table at D[v_2][$\{v_4\}$],

we see that the path's total weight is 10. Now, we need to find the total weight of the path $< v_3, v_2 >$. We can find this in the same way as smaller subsets of A. The weight is 7. For path 1, the total weight is $10 + 7 = 17$.

We do the same steps to calculate path 2, $< v_3, v_4, v_2, v_1 >$. $< v_4, v_2, v_1 >$ can be found at D$[v_4][\{v_2\}]$, which is 4. Looking up $< v_3, v_4 >$, we see that it is 8. So, the total weight of path 2 is $4 + 8 = 12$.

Now, compare the total weights of the paths. $12 < 17$ so the optimal path is path 2. Path 2's total weight, 12, is inserted at D$[v_3][\{v_2, v_4\}]$.

****For sets of size 3:

For increasing sizes, the process is the same as smaller sizes. However, seeing a set of size 3 might be helpful to further make this process concrete.

Consider D$[v_1][\{v_2, v_3, v_4\}]$. This is our only set of size 3 in this problem. As we've seen in sets of size 2, each vertex in the set A creates another possible path that needs to be tested.

There are three vertices in A so there are three possible paths that we need to check.

Path 1: v_1 followed by v_2
Path 2: v_1 followed by v_3
Path 3: v_1 followed by v_4

Technically, there are more than three paths that are tested, but in order to get to this step, the subpath following the second vertex have already been decided. For example, in path 2, we already know that the shortest path from v_3 to v_1 is $< v_3, v_4, v_2, v_1 >$. For proof, see the set of size 2 example.

Just as before, we need to test each possible path.

Path 1: Find $< v_2, ..., v_1 >$ by using the set A minus the chosen vertex. In this case, the chosen vertex is v_2 so we need to search for the set $\{v_3, v_4\}$. D$[v_2][\{v_3, v_4\}] = 20$. Looking up $< v_1, v_2 >$ in the same way we did in examples of smaller sizes, we find that the weight is 2. The total weight for path 1 is $20 + 2 = 22$.

Path 2: D$[v_3][\{v_2, v_4\}] = 12$ and $< v_1, v_3 > = 9$ The total weight of path 2 is $12 + 9 = 21$

Path 3: D$[v_4][\{v_2, v_3\}] = \infty$ and $< v_1, v_4 > = \infty$ The total weight of path 3 is $\infty + \infty = \infty$.

After comparing the total weights of all three paths, we see that path 2, with v_3 being the second vertex, is the most optimal path. The total weight that is inserted into D is 21.

2.3 Matrix P

Matrix P is filled with vertices so that the shortest path can be reconstructed going backwards through the sizes of A.

Our matrix P would look like this:

	$\{\}$	$\{v_2\}$	$\{v_3\}$	$\{v_4\}$	$\{v_2, v_3\}$	$\{v_2, v_4\}$	$\{v_3, v_4\}$	$\{v_2, v_3, v_4\}$
$\{v_1\}$	X	X	X	X	X	X	X	3
$\{v_2\}$	X	X	3	4	X	X	3	X
$\{v_3\}$	X	2	X	4	X	4	X	X
$\{v_4\}$	X	2	3	X	2	X	X	X

Matrix P can be updated as matrix D is being calculated. After updating a shortest path on any subset in D where you have to pick a v_j or second vertex, simply record that vertex into the corresponding location in P as in D. Again, locations where the same vertex would be counted twice are not considered.

2.4 Example of Backtracing Matrix P

Start at v_1 and grab the vertex located at P$[v_1][A]$. Notice that A is $V - v_1$ or $\{v_2, v_3, v_4\}]$. The vertex that was found is v_3.

Take the vertex that was found, in this case, v_3, and subtract it from A. Now, $A = V - v_1, v_3$ or $\{v_2, v_4\}$. P$[v_3][A]$ gives us 4, which corresponds to v_4.

In general, you would continue this process of subtracting and looking-up until A becomes an empty or null set.

P$[v_4][A] = 2$ or v_2

So, an optimal tour that was found is $< v_1, v_3, v_4, v_2, v_1 >$.

3 Pseudocode

```
// Global variables for Algorithm to update
// Length of the optimal tour
minLength

//2d array from which optimal tour can be constructed
// Rows of P are indexed from 1 to n
// cols of P are indexed by all subsets A of V={v1}
```

```
// P[i][A] is the index of the first vertx after v_i on the
// shortest path frpm v_i to v_1 that passes through all the
// vertices in A exactly once
P
function TSP(W):{ // The length of W is n
        D=[1....n][subeset of V-v_1]

        for(i=2;i≤n;i++)
                D[i][Ø]=W[i][1]
        for(k=1;k≤n-2;i++)
                for(all A ⊆ V-v_1 of size k)
                    for(i such that i ≠ 1 and v_1 ∉ A)
                        D[i][A] = minimum(W[i][j]+D[j][A-{v_i, v_j} ])
                        P[i][A] = value of j that gave the minimum
        D[1][V-{v_i}= minimum(W[1][j]+D[j][V-{v_i, v_j} ])
        P[1][V-{v_i}= value of j that gave the minimum
        minLength=D[1][V-{v_i}
        }
```