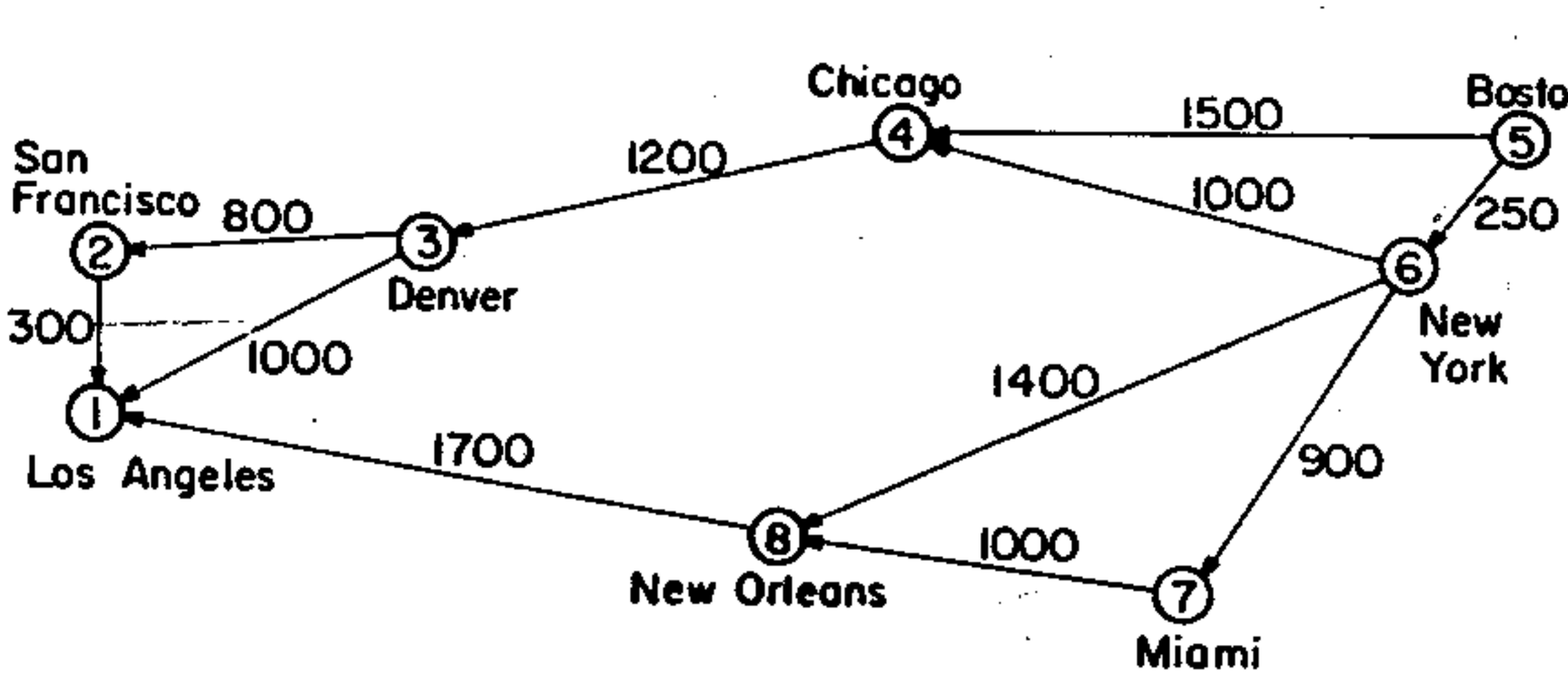
Algorithm for BFS

```
/* Traverse: recursive traversal of a graph */
void Traverse(int v)
{
        int w;

        visited[v] = TRUE;
        Visit(v);
        for (all w adjacent to v)
                if (!visited[w])
                        Traverse(w);
    }
```

```
/* BreadthFirst: perform breadth-first traversal of a graph. */
void BreadthFirst(Graph_type G)
{
        Vertexqueue_type Q;
        Boolean_type visited[MAX];
        int v, w;

        for (all v in G)
                visited[v] = FALSE;
        Initialize(Q);                          /* Set the queue to be empty.          */
        for (all v in G)
                if (!visited[v]) {
                        AddQueue(v, Q);
                        do {
                                DeleteQueue(v, Q);
                                visited[v] = TRUE;
                                Visit(v);
                                for (all w adjacent to v)
                                        if (!visited[w])
                                                AddQueue(w);
                        } while (!Empty(Q));
                }

}
```

Example



	1	2	3	4	5	6	7	8
1	0							
2	300	0						
3	1000	800	0					
4			1200	0				
5				1500	0	250		
6				1000		0	900	1400
7							0	1000
8	1700							0

Example

Vertex Iteration	S	Selected	*dist*	LA [1]	SF [2]	D [3]	C [4]	B [5]	NY [6]	M [7]	NO [8]
Initial		—		+∞	+∞	+∞	1500	0	250	+∞	+∞
1	5	6		+∞	+∞	+∞	1250	0	250	1150	1650
2	5,6	7		+∞	+∞	+∞	1250	0	250	1150	1650
3	5,6,7	4		+∞	+∞	2450	1250	0	250	1150	1650
4	5,6,7,4	8		3350	+∞	2450	1250	0	250	1150	1650
5	5,6,7,4,8	3		3350	3250	2450	1250	0	250	1150	1650
6	5,6,7,4,8,3	2		3350	3250	2450	1250	0	250	1150	1650
	5,6,7,4,8,3,2										

Shortest Path Algorithm

```
#define INFINITY ...
#define MAXNODES ...
#define MEMBER  1
#define NONMEMBER  0
shortpath (weight, s, t, pd, precede)
int weight[][MAXNODES];
int s, t, *pd, precede[];
{
    int distance[MAXNODES], perm[MAXNODES];
    int current, i, k, dc;
    int smalldist, newdist;

    /* initialization */
    for (i = 0; i < MAXNODES; ++i) {
        perm[i] = NONMEMBER;
        distance[i] = INFINITY;
    }   /*  end for */
    perm[s] = MEMBER;
    distance[s] = 0;
    current = s;
    while (current != t) {
        smalldist = INFINITY;
        dc = distance[current];
        for (i = 0; i < MAXNODES; i++)
            if (perm[i] == NONMEMBER) {
                newdist = dc + weight[current][i];
                if (newdist < distance[i]) {
                    /* distance from s to i through current is */
                    /*       smaller than  distance[i]          */
                    distance[i] = newdist;
                    precede[i] = current;
                }   /* end  if */
                /* determine the smallest distance */
                if (distance[i] < smalldist) {
                    smalldist = distance[i];

                    k = i;
                }   /* end if */
            }   /* end for ... if  */
        current = k;
        perm[current] = MEMBER;
    }   /*  end  while  */
    *pd = distance[t];
}   /* end  shortpath  */
```