```
void shortestPath(int v, int cost[][MAX_VERTICES],
                  int distance[], int n, short int found[])
{/* distance[i] represents the shortest path from vertex v
    to i, found[i] is 0 if the shortest path from i
    has not been found and a 1 if it has, cost is the
    adjacency matrix */
   int i,u,w;
   for (i = 0; i < n; i++) {
      found[i] = FALSE;
      distance[i] = cost[v][i];
   }
   found[v] = TRUE;
   distance[v] = 0;
   for (i = 0; i < n-2; i++) {
      u = choose(distance,n,found);
      found[u] = TRUE;
      for (w = 0; w < n; w++)
         if (!found[w])
            if (distance[u] + cost[u][w] < distance[w])
               distance[w] = distance[u] + cost[u][w];
   }
}
```

프로그램 6.9: 하나의 출발점에서 최단 경로

```
int choose(int distance[], int n, short int found[])
{/* find the smallest distance not yet checked */
   int i, min, minpos;
   min = INT_MAX;
   minpos = -1;
   for (i = 0; i < n; i++)
      if (distance[i] < min && !found[i]) {
         min = distance[i];
         minpos = i;
      }
   return minpos;
}
```

프로그램 6.10: 최저 비용 간선의 선택

```
void allCosts(int cost[][MAX_VERTICES],
              int distance[][MAX_VERTICES], int n)
{/* compute the shortest distance from each vertex
    to every other, cost is the adjacency matrix,
    distance is the matrix of computed  distances */
  int i,j,k;
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      distance[i][j] = cost[i][j];
  for (k = 0; k < n; k++)
    for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
        if (distance[i][k] + distance[k][j] <
                                        distance[i][j])
          distance[i][j] =
          distance[i][k] + distance[k][j];
}
```

프로그램 6.12: 모든 쌍의 최단 경로를 구하는 함수