Pseudocódigo Agente Viajero

↱ Matriz adyacencia

```
void agenteViajero (vector<vector<int>> m, int inicio) {

  int n = m.size() // n nodos
  int mejorCosto = ∞
  vector <int> mejorCamino
  vector <bool> visitado
  vector<int> stack

  // Visitamos el 1er nodo
  stack.push_back(inicio)
  visitado[inicio] = true

  // Exploramos todos los niveles con branch and bound
  explorarVecinos(m, visitado, stack, nivel = 1,
                  nodoInicial = inicio, mejorCosto = 0,
                  mejorCamino)

  // Imprimimos los resultados
  print "mejor camino", inversa(mejorCamino), mejorCosto
}

void explorarVecinos (matriz, visitado, stack, costo, nodoActual,
                      nivel, inicio, mejorCosto, mejorCamino)

  // Si llegamos al último nivel, actualiza los valores
  if (nivel == m.size()) {
    int costoTotal = costo + m[nodoActual][inicio] // Porque volvemos
                                                   al nodo inicial
    // Si el costo es menor, actualiza mejor solución
    if (costoTotal < costo) {
      costo = costoTotal
      mejorCamino = stack
      mejorCamino.push_back(inicio)
    }
    return // Termina el proceso
  }
  else {

    ...
```

```
// Si no estamos en el último nivel, explora vecinos
for (int i=0; i<m.size(); i++) {

    // Revisar que no haya sido visitado y no sea diagonal
    if (!visitado[i] && m[nodoActual][i] != 0) {

        int nuevaCosto = costo + m[nodoActual][i]

        // Si es mejor el costo, explorar rama
        if (nuevoCosto < mejorCosto) {
            visitado[i] = true
            stack.push_back(i)
            explorarVecinos(m, visitado, stack, nuevoCosto,
                            nodoActual=i, nivel+1, inicio,
                            mejorCosto, mejorCamino)

        // Cuando ya exploramos todo, regresamos para
           ver otras posibilidades

        stack.pop(último)
        visitado[i] = false
    }
}
}
}
```