Pseudocódigo A*

```
void a_star (matriz m, vector h_list , int inicio, int obj ) {

    vector <pair<int, int>> stack
    vector<int> c_list // Costos acumulados
    vector<int> padres // Para saber el padre de cada nodo
    vector<bool> visitado // Para saber si ya visitamos nodos
    stack.push_back (inicio, h_list[inicio]) // Inicializamos stack con
    c_list[inicio]=0                          1er nodo

// Vamos sacando el nodo con f mínima del stack en
   cada iteración para explorar sus vecinos

while (!stack.empty) {

   pair<int, int> actual = stack.pop (mínimo)

   // Si no lo hemos visitado, lo procesamos
   if (!visitado [nodo [0]] {

      visitado [nodo ] = true

      // Si es el objetivo, terminamos el ciclo
      if (actual == obj ) break

      // recorremos sus vecinos
      for (j=0 ; j<m.size() ; j++) {

         // Si no es diagonal ni infinito ni ha sido visitado
         if (m [actual][j] != 0 && m [actual ][j] != oo && !visitado [j] ){

            // Calculamos nuevo costo
            int nuevoc = c_list [actual ] + m [actual ][j ]

            // Si es un nuevo camino más corto, actualizamos
            IF (nuevoc < c_list [j ]) {
               c_list[j ] = nuevoc
               padres[j ] = actual
               int nuevof = nuevoc + h_list [j ]
               stacks. push_back (j, nuevof)
            }
         }
      }
   }
}
```

```
// Una vez terminó el ciclo, recorrer la lista de
   padres desde el objetivo hasta el inicio

   vector<int> recorrido
   int i = obj

// Mientras el índice no sea el nodo inicial, agregamos al
   recorrido final

   while (i != inicio) {
      recorrido.push_back(i)
      i = padre[i]
   }
   // Agregamos el último nodo que falta (inicio)
   recorrido.push_back(inicio)

   // Imprimimos el recorrido al revés, dándonos el
      camino en orden

   print recorrido, costo
}
```