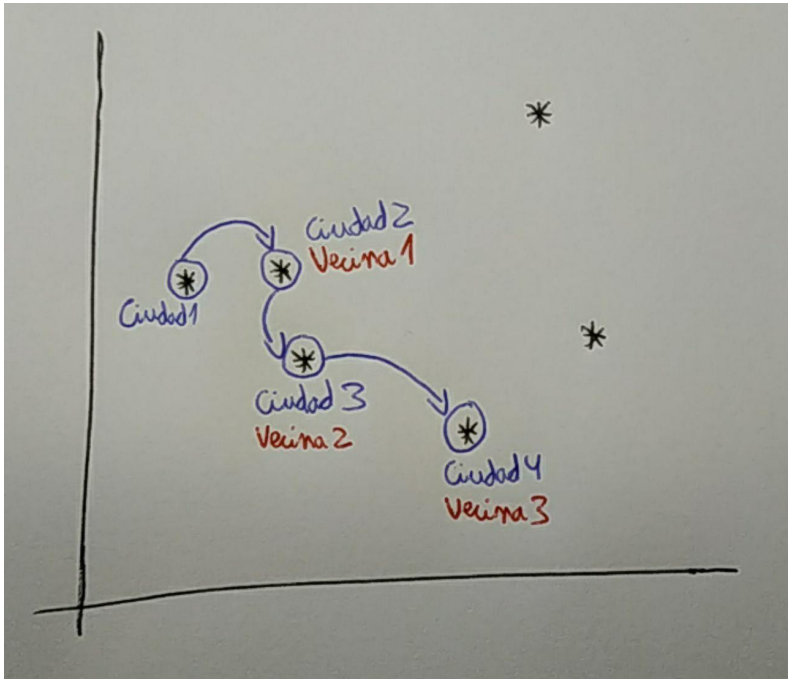
Práctica 3: Algoritmos Voraces (Greedy)

Grupo 4

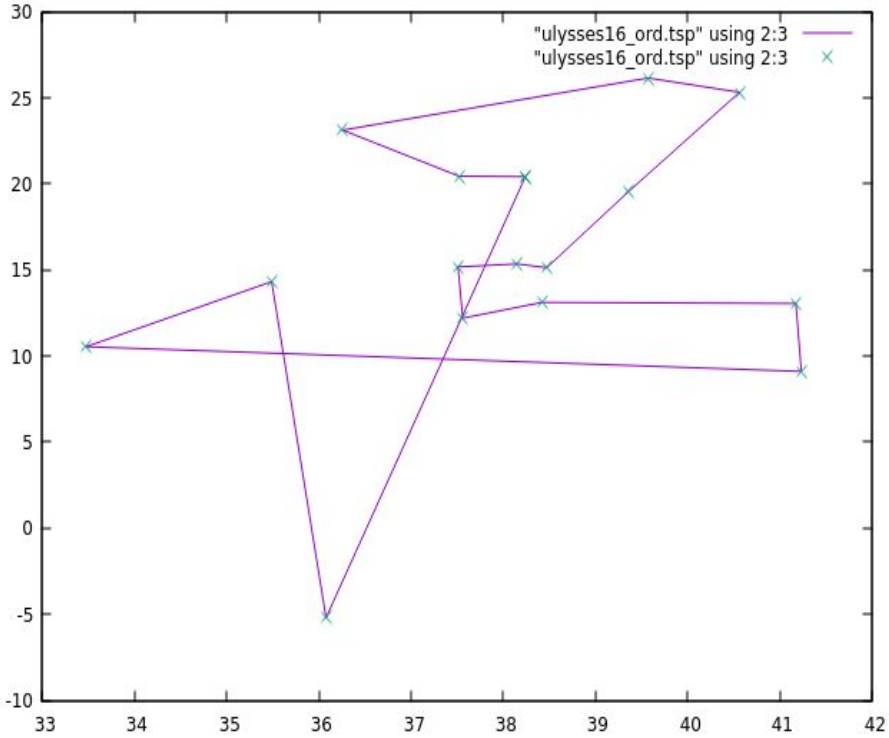
Integrantes: Raúl Rodríguez Pérez, Francisco Javier Gallardo Molina, Inés Nieto Sánchez, Antonio Lorenzo Gavilán Chacón

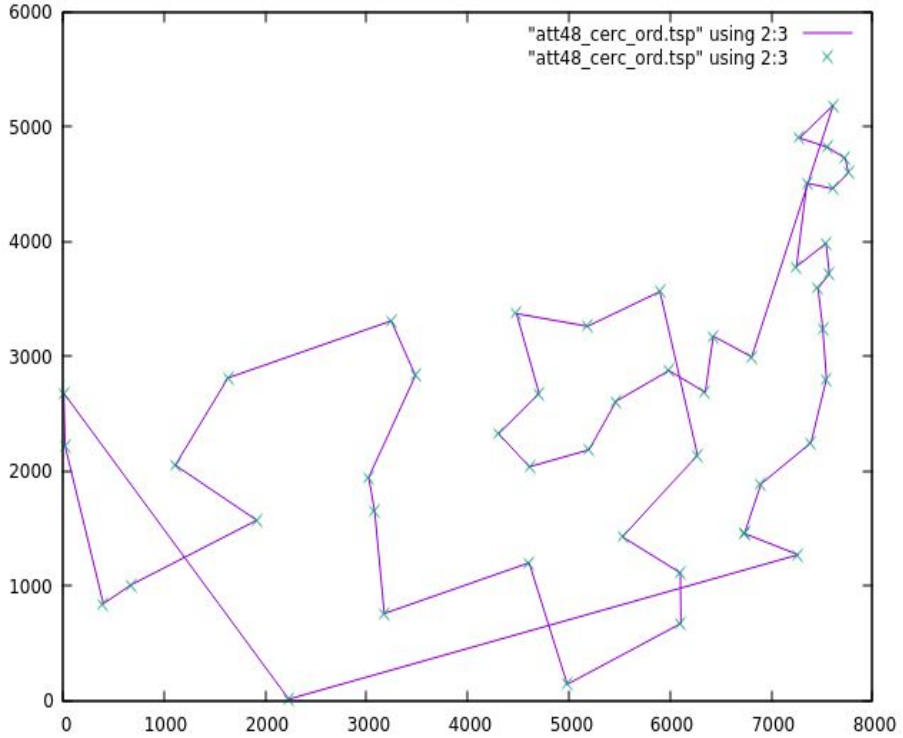
Viajante de comercio de cercanías



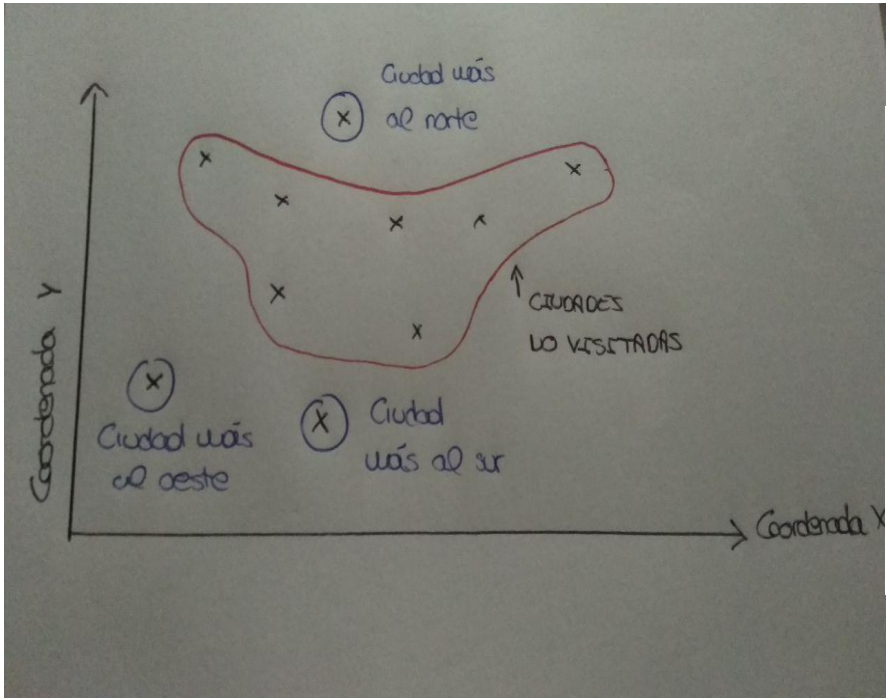

```
Require: Conjunto de ciudades C,S
x=0
S=C[0]
for  i = 1 to len(C) do
        if C[i] MenorDistancia then
                x = C[i]
                S.add(x)
                C[i].erase
        end if
end for
        return S
```

Viajante de comercio de cercanías



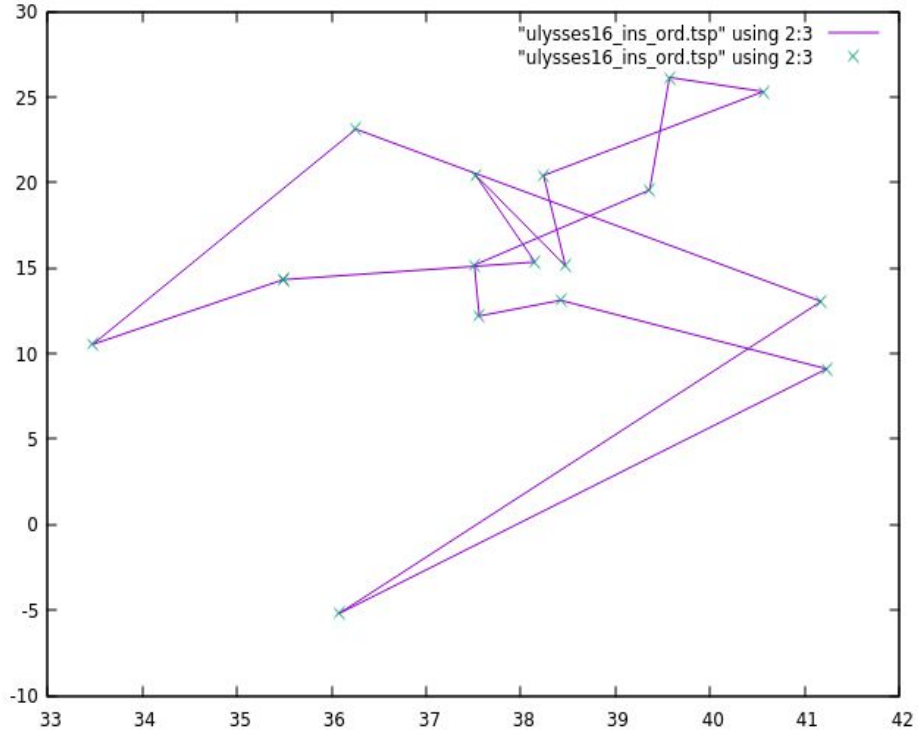


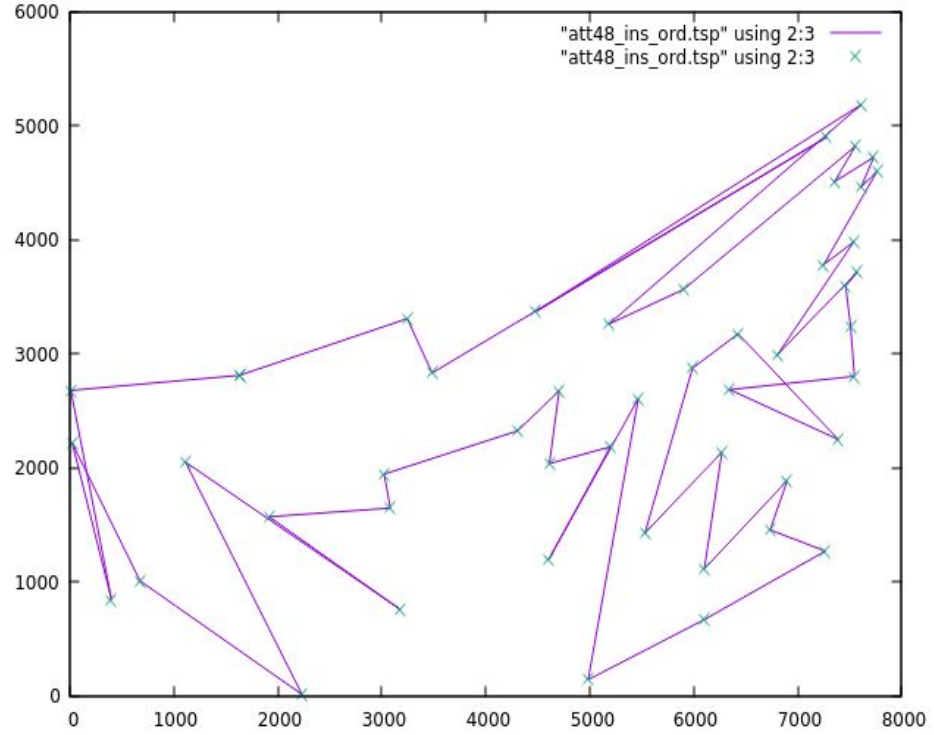
Viajante de comercio de inserción



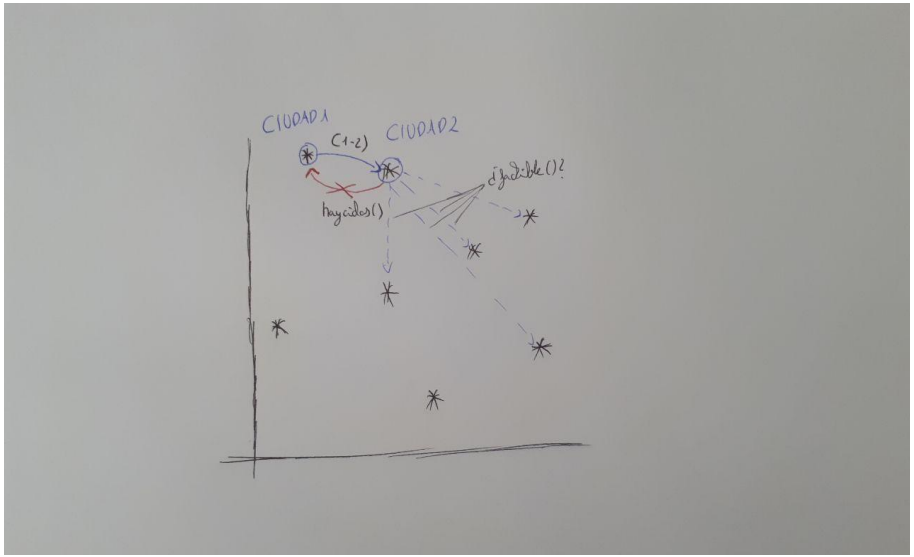

```
Require: Conjunto de ciudades C,S, distancia
x=0; S=CalcularRecorridoInicial();        // cálculo del triángulo
                                             de ciudades
while len(C)>0 do
        pair<int, list<int>::iterator> p = elegir_ciudad();
        insertar_ciudad(S,p.first, p.second);          // Insertar
                                                    cuidad no visitada
end while
distancia = distancia_total()             //Calcular la distancia
                                          | total del recorrido
        return S;
```

Viajante de comercio de inserción



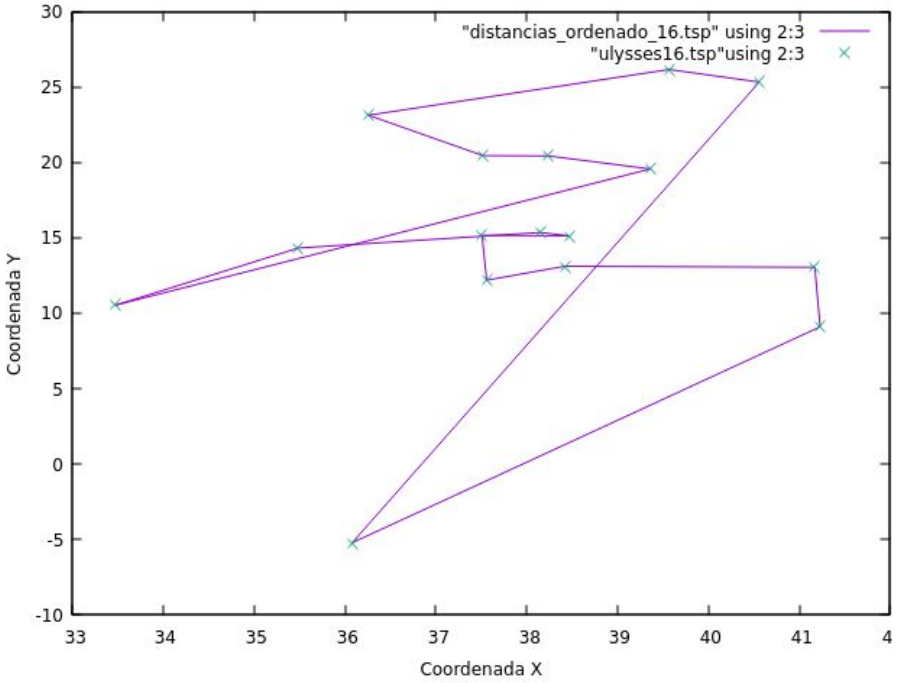


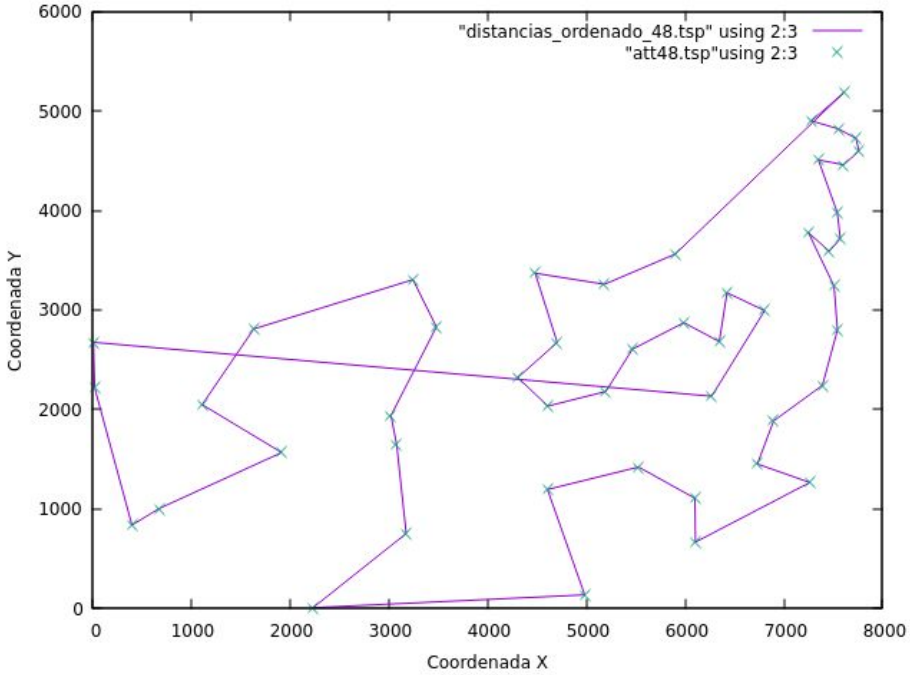
Viajante de comercio de distancias




```
Función recorrido (aristas, distancia)
    C = Ø;
    it ⟵ Iterador sobre aristas;
    aux.add (ciudad1, ciudad2);
    distancia ⟵ Distancia entre ciudad1 y ciudad2;
    it ++;
    while it distinto de it.end do
        if es factible (aux, (*it).second) and no hay ciclos (aux, (*it).second) then
            aux.add ((*it).second));
            distancia.add ((*it).first));
        end if
    end while
    aux ⟵ Añadir las dos ciudades con una arista para cerrar el ciclo;
    actualizar distancia;
    C ⟵ Calcular camino sobre aux;
    return C;
```

Viajante de comercio de distancias





Trabajadores y tareas

- Existen **n** personas y **n** trabajos.
- Cada persona **i** puede realizar un trabajo **j** con más o menos rendimiento: **B[i, j]**.
- **Objetivo:** asignar una tarea a cada trabajador (asignación uno-a-uno), de manera que se maximice la suma de rendimientos.

Tareas

Personas B	1	2	3
1	4	9	1
2	7	2	3
3	6	3	5

Ejemplo 1. (P1, T1), (P2, T3), (P3, T2)

B_{TOTAL}= 4+3+3= 10

Ejemplo 2. (P1, T2), (P2, T1), (P3, T3)

B_{TOTAL}= 9+7+5= 21


```
Require: Matriz de costos m
  for cont = 0 to size(m) – 1 do
    aux = m[0][0]; a = 0; b = 0; s = (0,0);
    for i = 0 to size(m) – 1 do
      for j = 0 to size(m) – 1 do
        if m[i][j] distinto a 999 and m[i][j] menor que aux then
          aux = m[i][j]; a = i; b = j;
        end if
      end for
    end for
    s.add(a,b);
    fila a – ésima de m <—— 999;
    columna b – ésima de m <—— 999;
  end for
  rerturn s
```