Algorítmica
Práctica #5

B&B y Vuelta Atrás

José María Gómez García
Fernando Lojano Mayaguari
Valentino Lugli
Carlos Mulero Haro

PROBLEMA: ITV

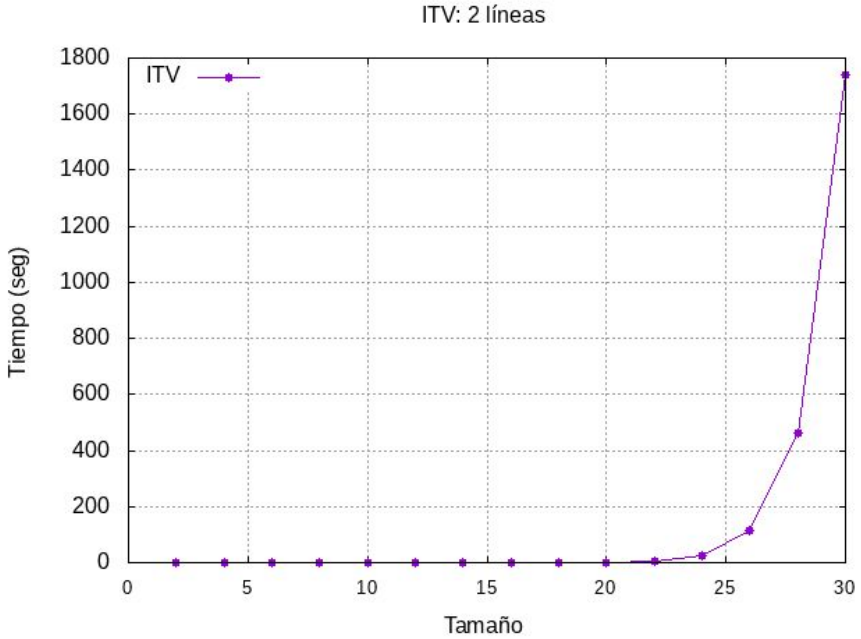
En el problema ITV queremos representar una estación de ITV que conste de varias líneas en las que se inspeccionan vehículos. Cada coche tendrá distintas características por lo que el tiempo de inspección varía según el vehículo. El objetivo del problema es atender a todos los vehículos en el menor tiempo posible.

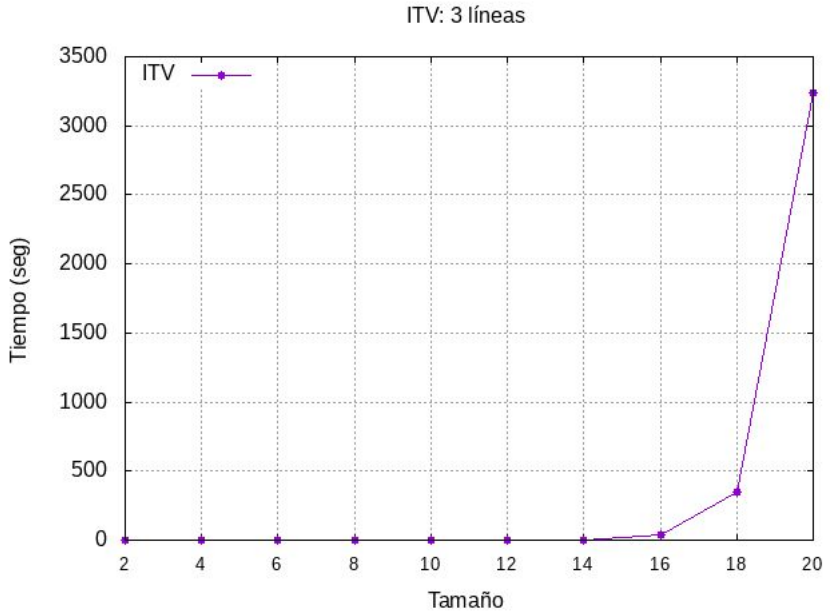
Para dar con una solución usando un algoritmo de vuelta atrás empezamos añadiendo un vehículo cualquiera a la primera línea (este será nuestro nodo raíz); a partir de este expandimos el árbol añadiendo el siguiente vehículo en cola a cada línea (cada vez que añadimos el siguiente vehículo en cola a una línea expandimos en un nodo); cuando no queden coches en cola quiere decir que hemos dado con una solución, para quedarnos con la solución que acabe en el menor tiempo posible seleccionamos aquella cuya línea con mayor tiempo de espera sea la menor de entre todas las líneas con mayor tiempo de espera de las soluciones.

PSEUDOCÓDIGO

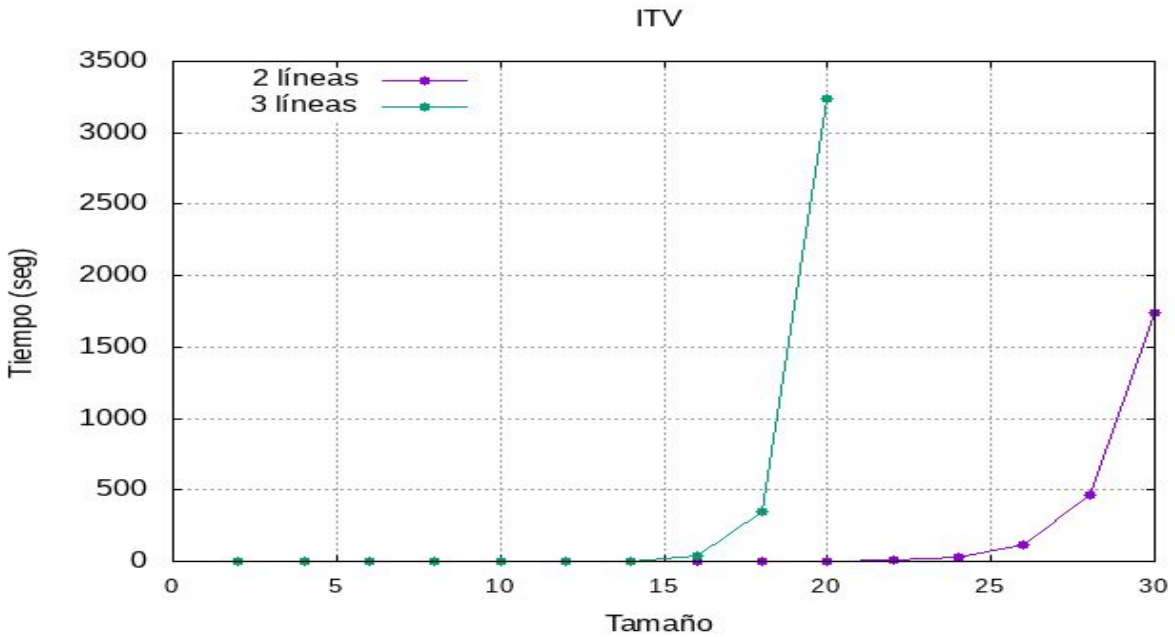
```
// coches_encola contiene los coches que todavia no estan en ninguna linea
// coches_listos contiene los coches en cada una de las lineas, por ejemplo, coches_listos[l][c] estaria accediendo al coche c de la linea l. Podemos empezar a resolver con un coche ya en la primera linea

funcion resolver_itv(choches_encola, coches_listos)

    if (!coches_encola.vacio)
        for i=0 while i < lineas
            coches_encola.push_back(coches_listos[0])
            coches_listos.erase(0)
            resolver(coches_encola, coches_listos)
        fin
    fi
    // No quedan coches en cola, todos estan listos, es un nodo hoja
    else
        tiempo_linea_max := int
        // Calculamos el tiempo en cada cola y nos quedamos con el mayor tiempo de espera
        for i=0 while i < lineas
            tiempo_linea[i] := tiempo(coches_listos[i])
            if tiempo_linea_max es menor que tiempo_linea[i] entonces
                tiempo_linea_max := tiempo_linea[i]
            fi
        fin
        // Si el mayor tiempo de espera es menor que la solucion que teniamos anteriormente este pasa a ser la solucion
        if tiempo_linea_max es menor que el tiempo_solucion entonces
            tiempo_solucion := tiempo_linea_max
            solucion := coches_listos
        fi
    esle
function
```

ANÁLISIS EMPÍRICO





ANÁLISIS EMPÍRICO: GRÁFICAS CONJUNTAS



ESCENARIOS DE EJECUCIÓN

Caso 1: 2 líneas de inspección

Tamaño	T L1	T L2	T Sol	Línea 1	Línea 2
2	4	7	7	1	2
4	22	23	23	3-1	4-2
6	37	38	38	5-4-2	6-3-1
8	48	47	48	7-6-5-2-1	8-4-3–
10	53	54	54	9-8-7-6-2	10-5-4-3-1
12	59	59	59	11-10-9-7-6-5-2	12-8-4-3-1
14	74	75	75	13-12-11-10-9-8-7-6-1	14-5-4-3-2
16	80	80	80	15-14-13-12-11-10-9-3-1	16-8-7-6-5-4-2
18	84	84	84	16-15-14-13-12-11-10-9-7-2-1	18-8-6-5-4-3
20	99	99	99	19-18-17-16-15-14-13-11-11-10-9-8	20-7-6-5-4-3-2-1
22	110	109	110	21-20-19-18-17-16-15-14-13-12-11-10-1	22-9-8-7-6-5-4-3-2
24	119	118	119	23-22-21-20-19-18-17-16-15-14-13-12-10	24-11-9-8-7-6-5-4-3-2-1
26	125	126	126	25-24-23-22-21-20-19-18-17-16-15-14-13-11	26-12-10-9-8-7-6-5-4-3-2-1
28	129	129	129	27-26-25-24-23-22-21-20-19-18-17-16-15-14-1	28-13-12-11-10-9-8-7-6-5-4-3-2
30	141	141	141	29-28-27-26-25-24-23-22-21-20-19-18-17-16-15-14-1	30-13-12-11-10-9-8-7-6-5-4-3-2

ESCENARIOS DE EJECUCIÓN

Caso 2: 3 líneas de inspección

Tamaño	T L1	T L2	T L3	T Sol	Línea 1	Línea 2	Línea 3
2	4	0	7	7	1	-	2
4	18	11	16	18	3	2-1	4
6	30	29	16	30	5-4	3-2-1	6
8	32	32	31	32	7-5-2-1	6-4	8-3
10	35	36	36	36	9-7-5-1	8-6-2	10-4-3
12	39	39	40	40	11-10-9-8-7-1	6-4-2	12-5-3
14	50	50	49	50	13-12-11-10-9-6	8-7-5-4	14-3-2-1
16	52	54	54	54	15-14-13-12-11-10-1	9-7-6-5-2	16-8-4-3
18	56	56	56	56	17-16-15-14-13-12-11-10	9-7-6-4-2	18-8-5-3-1
20	66	66	66	66	19-18-17-16-15-14-13-11	12-10-9-7-6-4-2	20-8-5-3-1

PROBLEMA: TSP

PSEUDOCÓDIGO

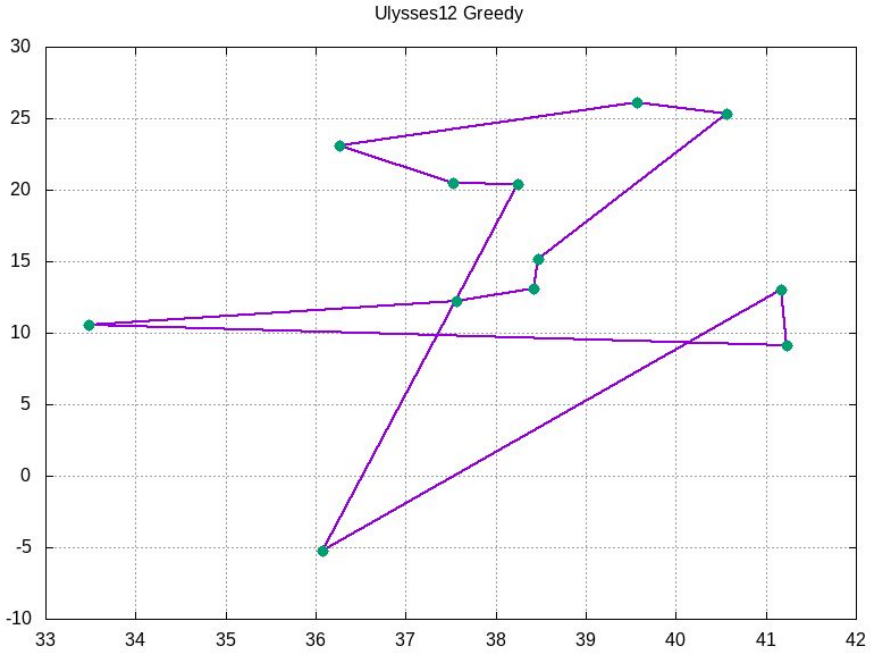
```
funcion resolver(cv, csv, dv, de)
cv_aux, csv_aux := vector        //vectores auxiliares de ciudades visitadas y sin visitar
dt, de_aux := int       //dt=distancia total
inicio

if(!cv.vacio)
  //ordenar por prioridad las ciudades sin visitar
  csv := prioridad(cv,back(), csv)
  cv_aux := cv

   desde i := 0 hasta i < csv.size()
      dt := 0
      cv_aux := cv
      //Se introduce la primera ciudad del vector sin visitar ordenado por prioridad
      cv_aux.push_back(csv[i])
      csv_aux := csv
      //Se elimina la ciudad del vector de ciudades sin visitar auxiliar
      csv_aux.erase(csv_aux.begin()+i)
      //Se incluye la distancia de la ciudad a la distancia total
      dt += dv + distancias[cv.back()][cv_aux.back()]
      //Se calcula la nueva distancia estimada
      de_aux = de - menorArista(cv_aux.back())

        if (dt + de_aux < ds)
           ne++                        //incremento nodos explorados
            resolver(cv_aux, csv_aux, dt, de_aux)       //entrada recursiva
         fi
         else n_podas++            //incremento nodos podados
  fin
fi
```

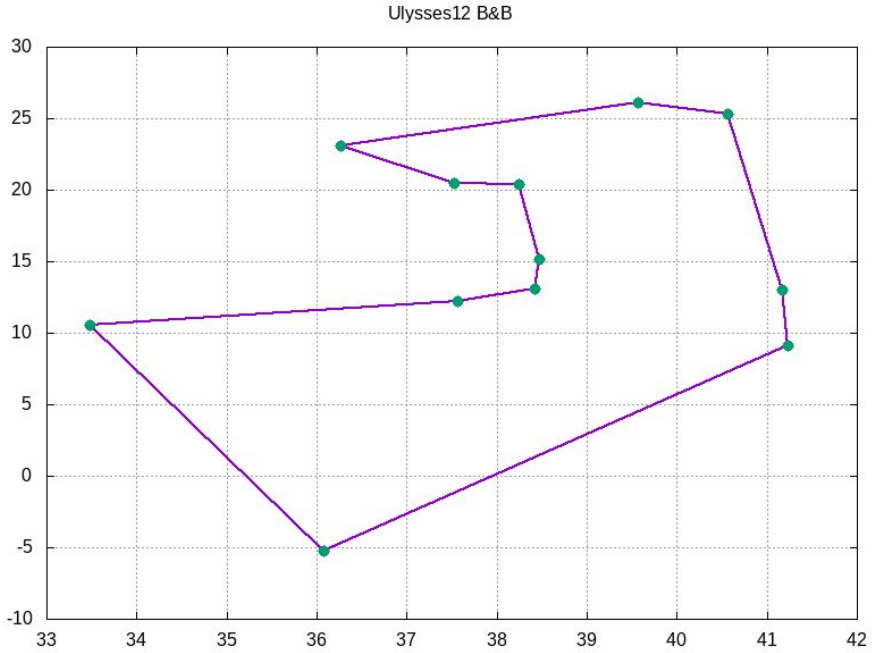
```
 else        //No quedan valores en ciudades visitadas para analizar
   dt := 0
   desde i:=0 hasta cv.size()-1
      dt += distancias[cv[i]][cv[i+1]]
   fin

   dt += distancias[cv.front()][cv.back()]

   if (dt < ds) // si distancia total menor que distancia solucion
      ds = dt  // Sustituye distancia solucion por distancia total
      cs = cv  // Sustituye camino solucion por ciudades visitadas
   fi
fin
```

-Variables a tener en cuenta:

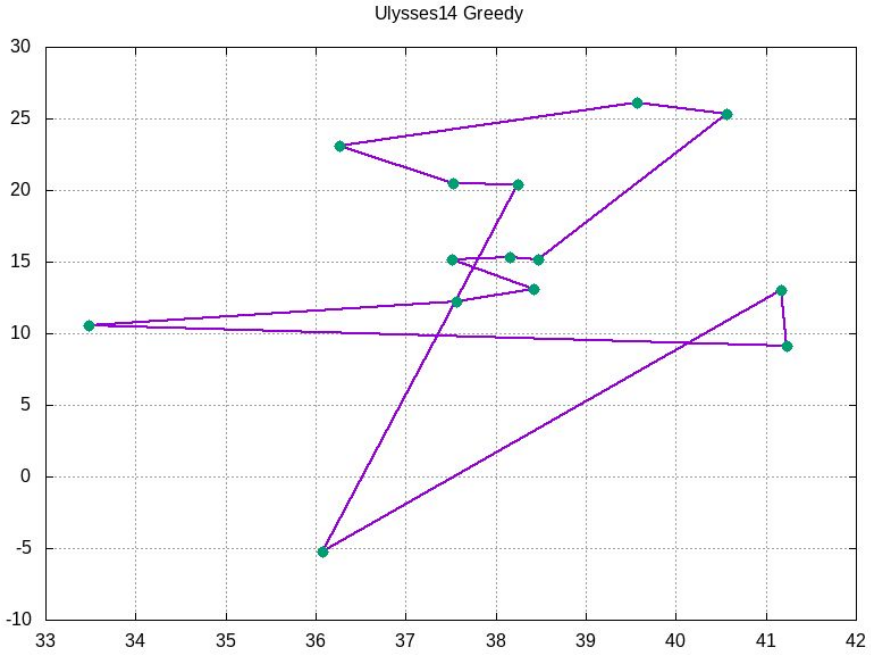
```
csv:=vector de ciudades sin visitar
cv:=vector de ciudades visitadas
dv:=distancia ciudades visitadas
de:= distancia estimada
ds:= distancia solucion
ne:= nodos explorados
n_podas := numero de podas realizadas
cs:= camino solucion
```

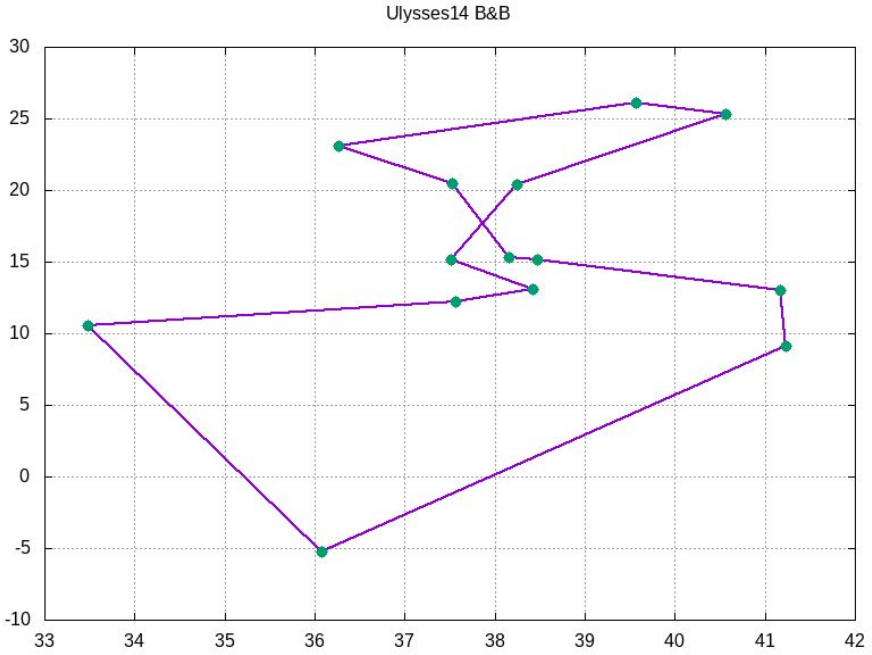
ESCENARIOS DE EJECUCIÓN



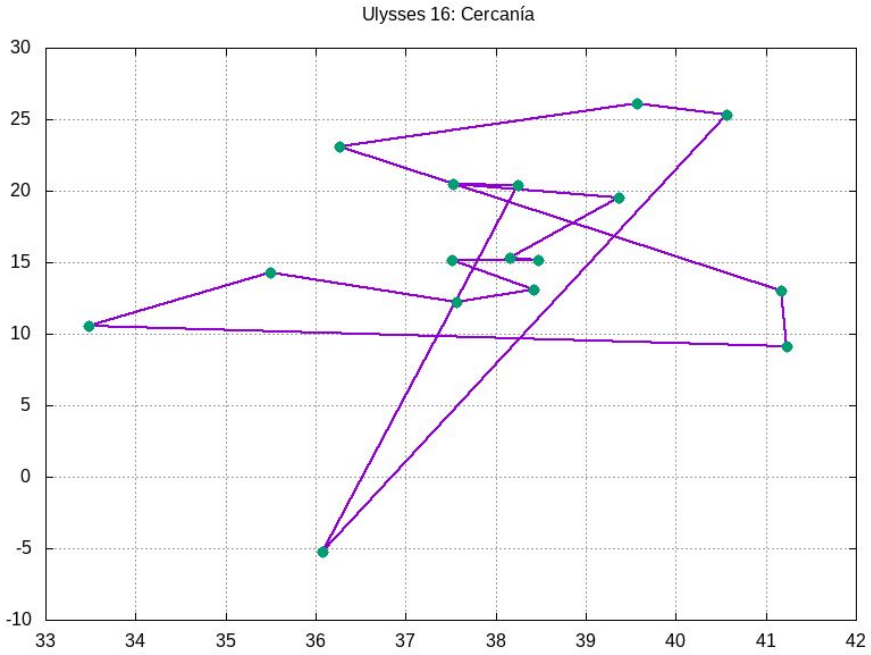


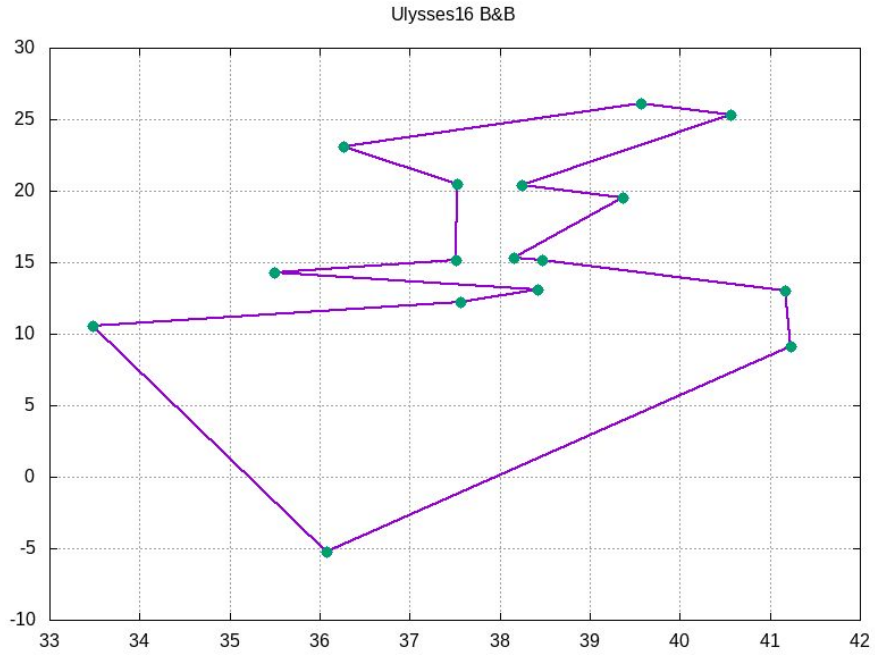
ESCENARIOS DE EJECUCIÓN



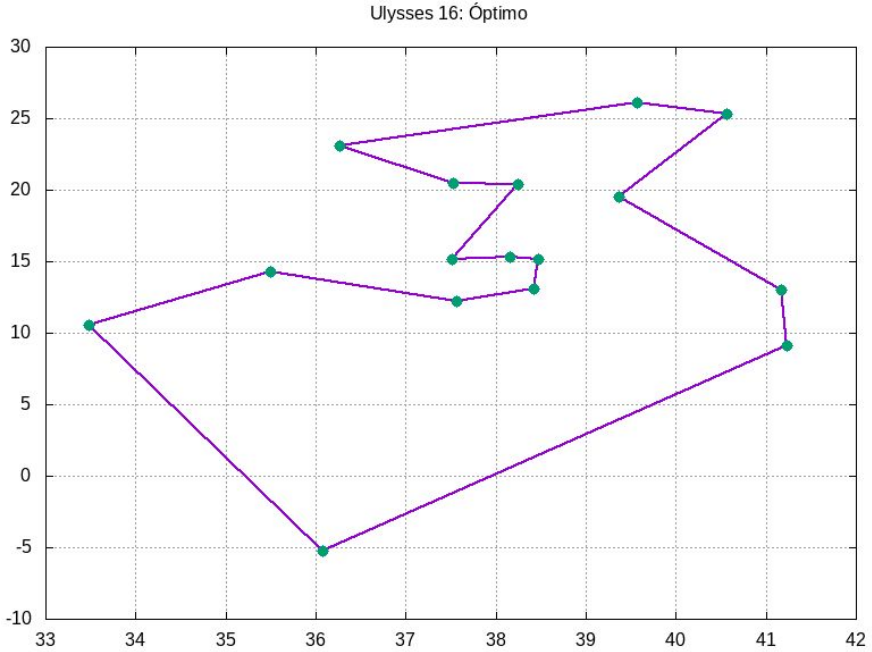


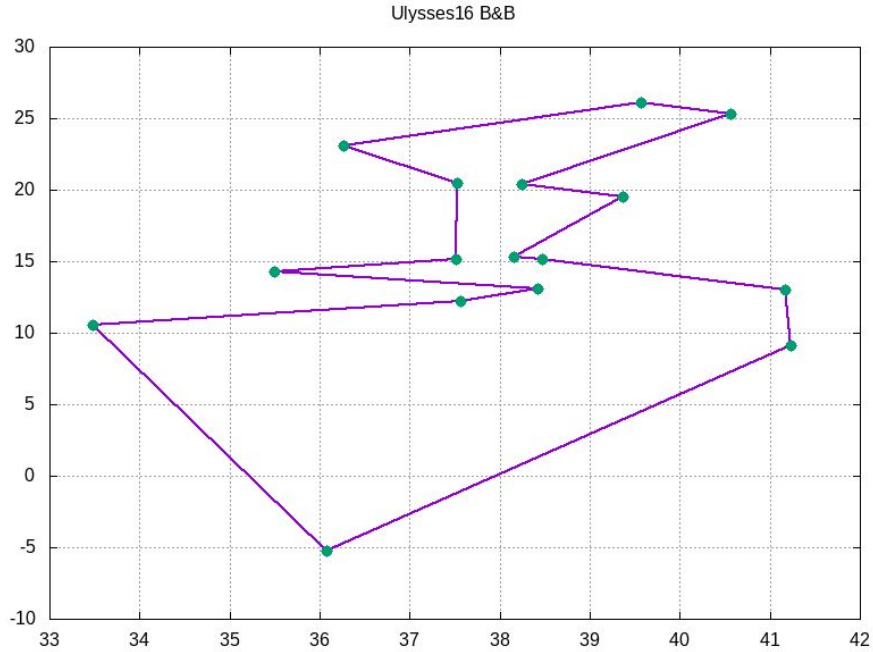
ESCENARIOS DE EJECUCIÓN





ESCENARIOS DE EJECUCIÓN





ESCENARIOS DE EJECUCIÓN

TSP	Greedy	B&B	Tiempo (s)	Nodos Expandidos	Número de podas	Tamaño máximo LNV
Ulysses6	36	35	0.000407	166	82	5
Ulysses8	40	36	0.009283	2246	2669	7
Ulysses12	83	68	13.0852	3701838	6443600	11
Ulysses14	84	68	49.8863	217594737	443383865	13
Ulysses16	103	71	3848.92	16799129630	39133662533	15

COMPARACIÓN ENTRE B&B Y BACKTRACKING

TSP	Tiempo Backtracking (seg)	Tiempo B&B (seg)
Ulysses6	3.5e-05	4.2e-05
Ulysses8	0.001272	0.000578
Ulysses12	9.87395	0.750011
Ulysses14	1614.15	46.6722
Ulysses16	338971.5	3861.45