# Algorítmica
# Práctica #4

Algoritmos Programación Dinámica

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

# TSP y propiedades de Programación Dinámica

**Problema N-Etápico**: Sí se cumple, ya que para obtener el ciclo hamiltoniano mínimo se debe decidir a qué ciudad viajar de primero, luego a cual otra ir de segundo y así sucesivamente hasta que finalmente se hayan visitado todas las ciudades una sola vez y se complete el ciclo regresando a la ciudad de origen.

**Verificación del Principio de Optimalidad de Bellman:** se verifica ya que si se tiene un ciclo hamiltoniano minimal que comienza y finaliza, por ejemplo, en la ciudad 1, este ciclo consiste de un camino de la ciudad 1 a otra ciudad j  junto a otro camino que parte de j visitando el resto de ciudades una sola vez y terminando en 1, si el ciclo posee el costo mínimo por lo tanto también el camino de j a 1 es de costo mínimo, ya que si existiese otro camino diferente de j  a 1 que fuese de coste aún menor, este ya hubiera sido elegido en el primer lugar, por lo tanto se cumple el Principio de Optimalidad de Bellman.

# Ecuación de Recurrencia

$$g(i,S) = \begin{cases} D_{[i,1]} & \text{si } S = \emptyset\ . \\ \text{Min}_{j \in S}(D_{[i,j]} + g(j, S - \{j\})) & \text{en caso contrario.} \end{cases}$$

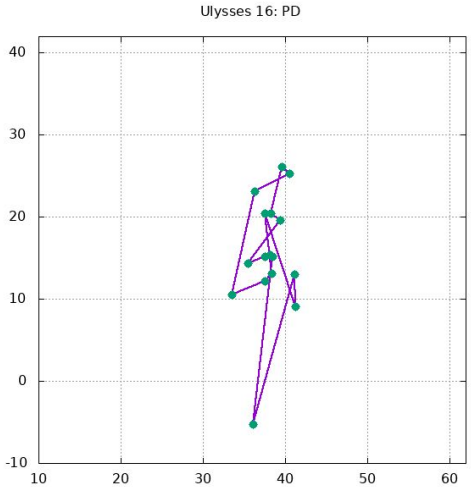
De aquí, podemos dar con la ecuación que dará el camino mínimo:

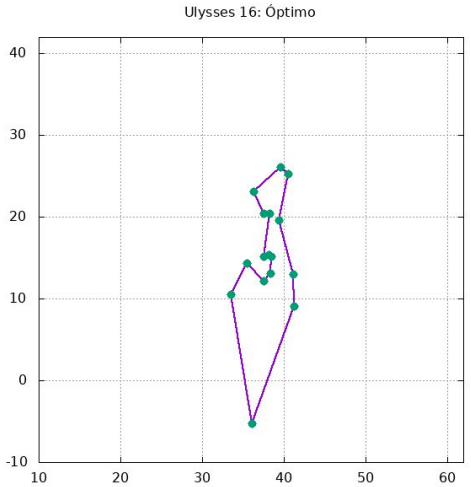
$$g(1, N - \{1\}) = \text{Min}_{2 \geq j \geq n}(D_{[1,j]} + g(j, S - \{j\}))$$

# PSEUDOCÓDIGO DE LA SOLUCIÓN

```
funcion g(n, s) //Siendo n un entero que representa un nodo y s el vector que
   contiene los dem s nodos del grafo
inicio
    distancia_min := INT_MAX
    distancia_min_aux := INT_MAX
    s_aux //Vector de int auxiliar

    si(!s.vacio())
        si(calculado(g(n,s)))
            distancia_min := distancia menor del mapa
        fin
        si (!calculado(g(n,s)))
            para i:=0 mientras i< s.size()
                s_aux = s
                s_aux.borrar(i)
                distancia_min_aux := distancias[n][s[i]] + g(s[i], s_aux)

                si(distancia_min_aux < distancia_min)
                    distancia_min := distancia_min_aux
                fin
            fin
            a adir_mapa(distancia_min)
        fin
    fin
    si(s.vacio())
        distancia_min := distancias[nodo][0]
    fin

    devolver distancia_min
fin
```

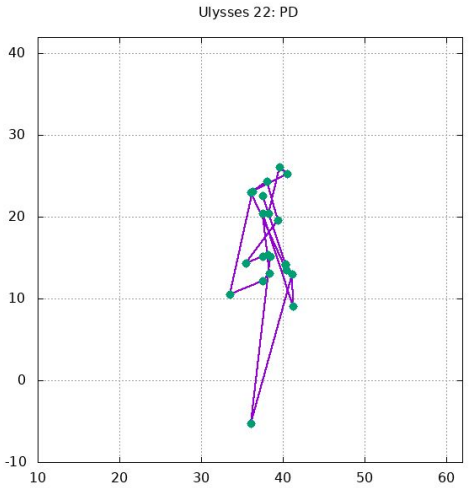
# Escenarios de Ejecución

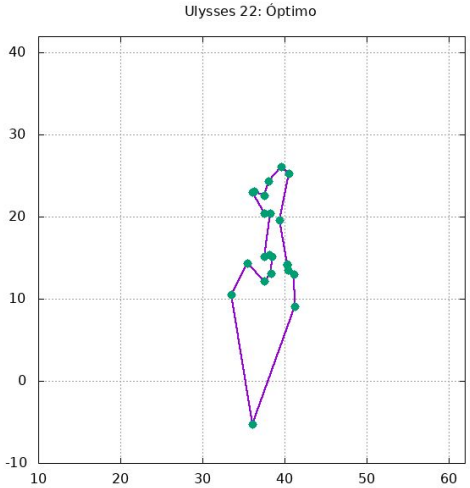



| TSP | Óptimo | PD | Similitud |
|---|---|---|---|
| Ulysses 16 | 73 | 71 | 97.26% |

# Escenarios de Ejecución




| TSP | Óptimo | PD | Similitud |
|---|---|---|---|
| Ulysses 22 | 74 | 72 | 97.30% |

# Comparación Greedy

| Ulysses16 | Tiempo de Ejecución (s) | Camino |
|---|---|---|
| Programación Dinámica | 6.312 | 71 |
| Cercanía | 0 | 103 |
| Inserción | 0 | 105 |
| Aristas | 0 | 90 |

# Comparación Greedy

| Ulysses22 | Tiempo de Ejecución (s) | Camino |
|---|---|---|
| Programación Dinámica | 1029.45 | 72 |
| Cercanía | 0 | 93 |
| Inserción | 0 | 107 |
| Aristas | 0 | 91 |

# TSP Comparación

| TSP | PD | Cercanía | | Inserción | | Aristas | |
|---|---|---|---|---|---|---|---|
| | | Camino | Diferencia | Camino | Diferencia | Camino | Diferencia |
| Ulysses 16 | 71 | 103 | 45,07% | 105 | 47,88% | 90 | 26,76% |
| Ulysses 22 | 72 | 93 | 29,17% | 107 | 48,61% | 91 | 26,38% |