



**DECSAI**

**Departamento de Ciencias de la Computación e I.A.**

Universidad de Granada



# Análisis y Diseño de Algoritmos

## Introducción

# Análisis y Diseño de Algoritmos

- Concepto de algoritmo
- Resolución de problemas
- Clasificación de problemas
- Algorítmica
- Análisis de la eficiencia de los algoritmos
- Técnicas de diseño de algoritmos



# Concepto de algoritmo



## Definición de algoritmo

Secuencia ordenada de pasos exentos de ambigüedad tal que, al llevarse a cabo con fidelidad, dará como resultado que se realice la tarea para la que se ha diseñado en un tiempo finito.

Un algoritmo nos permite obtener la solución del problema para el que esté diseñado.



# Concepto de algoritmo



## Propiedades de un algoritmo

- **Finitud:**  
La ejecución de un algoritmo ha de terminar después de un número finito de etapas.
- **Precisión:**  
Cada etapa ha de estar especificado rigurosamente. La ejecución de un algoritmo no ha de dejar espacio para la interpretación, la intuición o la creatividad.



# Concepto de algoritmo



## Características de un algoritmo

### ■ Entradas:

Un algoritmo tiene cero o más entradas  
(cantidades que se le dan inicialmente antes de que comience su ejecución).

### ■ Salidas:

Un algoritmo tiene una o más salidas  
(cantidades que tienen una relación específica con las entradas).



# Resolución de problemas

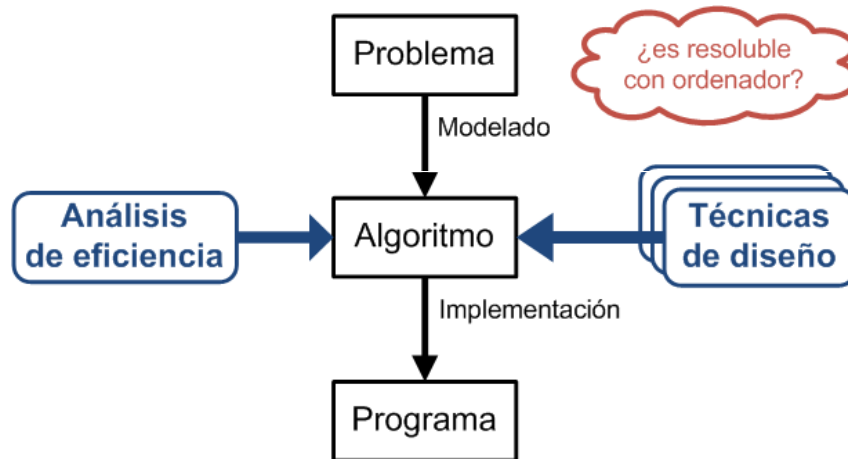


## Resolución de problemas con ordenador:

- Diseñar un algoritmo para el problema.
- Expresar el algoritmo como un programa.
- Ejecutar el programa.



# Resolución de problemas



# Clasificación de problemas



- **Años 30**  
Problemas computables y no computables.
- **Años 50**  
Complejidad de los problemas computables (búsqueda de algoritmos más eficaces).
- **Años 70**  
Clasificación de los problemas computables: P y NP.



# Clasificación de problemas



## Clases P y NP

### ■ Clase P

Problemas resolubles en tiempo polinómico con una máquina de Turing determinística (esto es, el tiempo de ejecución del algoritmo en un ordenador viene descrito por una fórmula polinómica).

### ■ Clase NP [**Non-Deterministic Polynomial-time**]

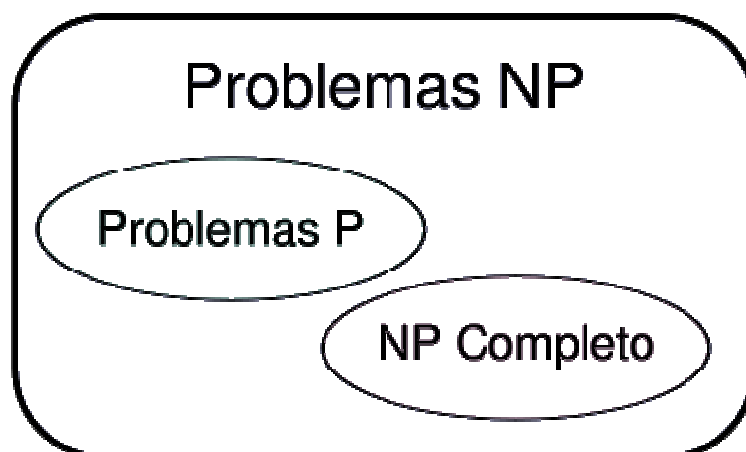
Problemas resolubles en tiempo polinómico con una máquina de Turing **no determinística**.



# Clasificación de problemas



## Clase P $\subseteq$ Clase NP



# Clasificación de problemas



## Reducción de problemas y complejidad

- Si reducimos un problema (A) a otro (B), podemos obtener una solución con un algoritmo para el problema B y transformar esa solución para convertirla en una solución para el problema A.
- **¿P=NP?** Si encontráramos un algoritmo polinómico para un problema NP-completo, sabríamos que todos los problemas de la clase NP se pueden resolver en tiempo polinómico.



# Algorítmica



La algorítmica, como disciplina de estudio de los algoritmos, ha de considerar:

- El diseño de algoritmos.
- La validación de algoritmos.
- El análisis de algoritmos.





## Diseño de algoritmos

- Creación de algoritmos (parte creativa).
- El diseño de algoritmos no se puede dominar si no se conocen las técnicas de diseño de algoritmos (métodos y heurísticas que han demostrado ser útiles en la práctica).



## Validación de algoritmos

- Demostración de que las respuestas dadas por el algoritmo son correctas para todas las posibles entradas.
- Único método válido: Demostración formal.





## Análisis de algoritmos

- Determinación de los recursos (espacio, tiempo) que consumen los algoritmos en la resolución de problemas.
- El análisis de algoritmos permite comparar algoritmos alternativos con criterios cuantitativos (y elegir el algoritmo más adecuado para resolver un problema).



## Análisis de la eficiencia



- **Problema:**  
Determinar las características del algoritmo que sirvan para evaluar su rendimiento.  
  
p.ej. Tiempo requerido para la ejecución del algoritmo en términos del número de veces que se ejecuta cada etapa del algoritmo.
- **Alternativas** (complementarias):
  - Enfoque empírico
  - Enfoque teórico
  - Enfoque híbrido

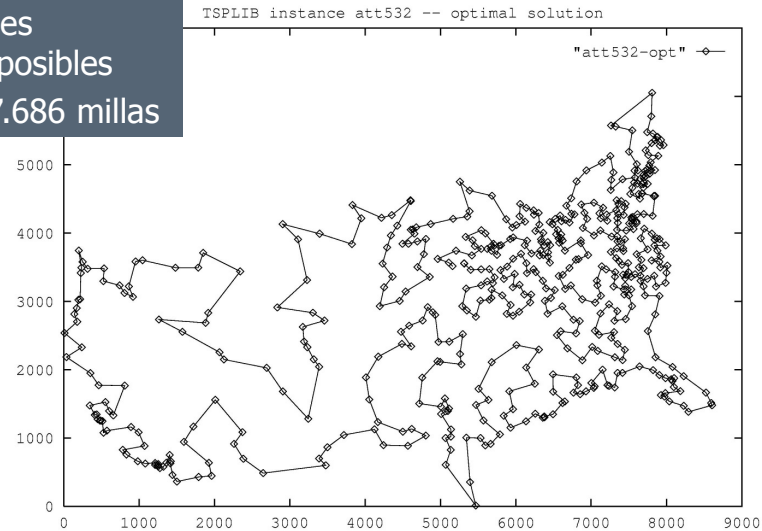


# Técnicas de diseño



## Ejemplo: **Problema del viajante de comercio (TSP)**

532 ciudades  
532! soluciones posibles  
Solución óptima: 27.686 millas



# Técnicas de diseño



## Ejemplo: **Problema del viajante de comercio (TSP)**

Si tenemos  $N$  ciudades e intentamos solucionar el problema por fuerza bruta (comprobando todas las soluciones posibles), tendremos que comprobar  $N!$  posibles soluciones.

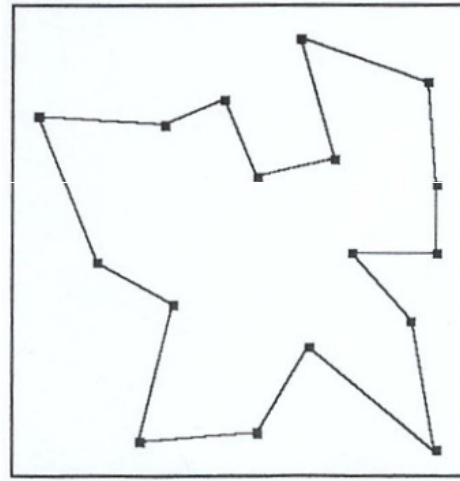
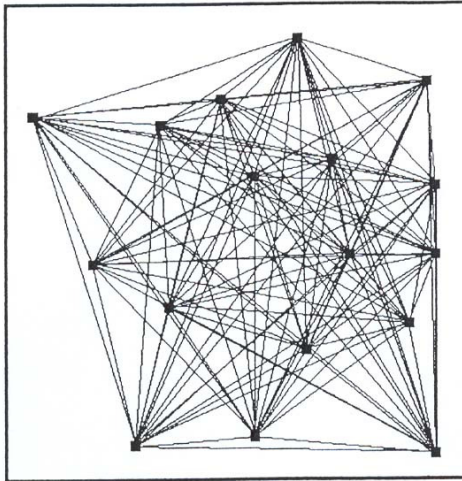
**Se necesitan técnicas que nos permitan diseñar algoritmos eficientes.**



# Técnicas de diseño



## Ejemplo: **Problema del viajante de comercio (TSP)**



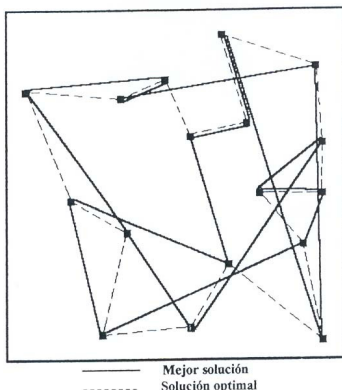
17 ciudades  
Solución óptima = 226.64  
Soluciones posibles =  $17! = 355\,687\,428\,096\,000$



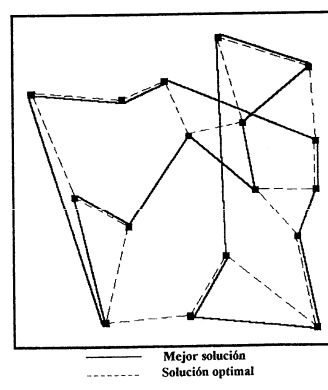
# Técnicas de diseño



## Solución aproximada con un algoritmo iterativo:



Iteración 0:  
Mejor solución = 403.7



Iteración 25:  
Mejor solución = 303.86

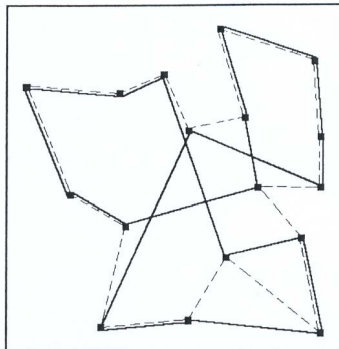
17 ciudades  
Solución óptima = 226.64  
Soluciones posibles =  $17! = 355\,687\,428\,096\,000$



# Técnicas de diseño



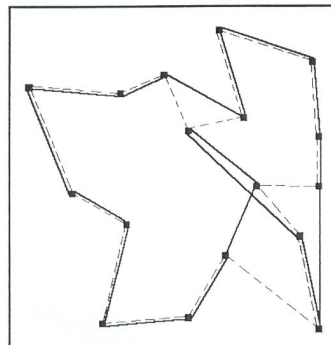
Solución aproximada con un algoritmo iterativo:



Mejor solución  
Solución óptima

Iteración 50:

Mejor solución = 293.6



Mejor solución  
Solución óptima

Iteración 100:

Mejor solución = 256.55

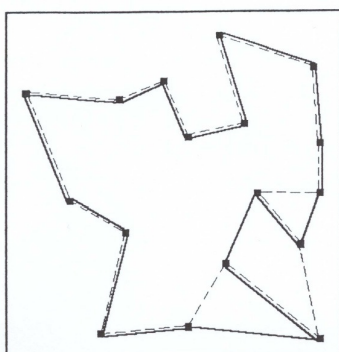
17 ciudades  
Solución óptima = 226.64  
Soluciones posibles =  $17! = 355\,687\,428\,096\,000$



# Técnicas de diseño



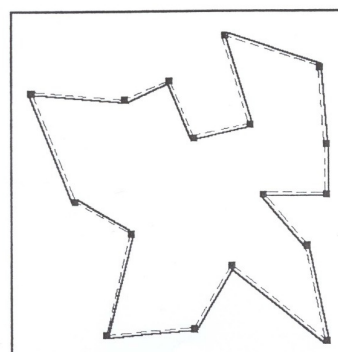
Solución aproximada con un algoritmo iterativo:



Mejor solución  
Solución óptima

Iteración 200:

Mejor solución = 231.4



Mejor solución  
Solución óptima

Iteración 250:

Mejor solución = 226.64

17 ciudades  
Solución óptima = 226.64  
Soluciones posibles =  $17! = 355\,687\,428\,096\,000$





## Técnicas de diseño de algoritmos

- “Divide y vencerás”
- Algoritmos greedy
- Programación dinámica
- Exploración de grafos
  - Branch & Bound (ramificación y poda)
  - Backtracking (“vuelta atrás”)
- Algoritmos probabilísticos
- Metaheurísticas

