Diseño de Algoritmos - Algoritmos II

Nazareno Aguirre, Sonia Permigiani, Gastón Scilingo,
Simón Gutiérrez
Departamento de Computación
Facultad de Ciencias Exactas, Físico-Químicas y Naturales
Universidad Nacional de Río Cuarto

Clase 2: Fuerza Bruta

Fuerza Bruta

Es una técnica directa, que generalmente está basada en la definición del problema, y de los conceptos involucrados en la definición.

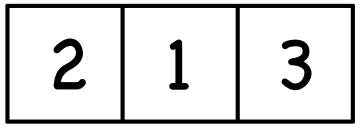
Ejemplos:

- Computar a^n ($a > 0$, n un entero no negativo)
- Computar n!
- Multiplicar dos matrices
- Buscar la clave de un valor dado en una lista

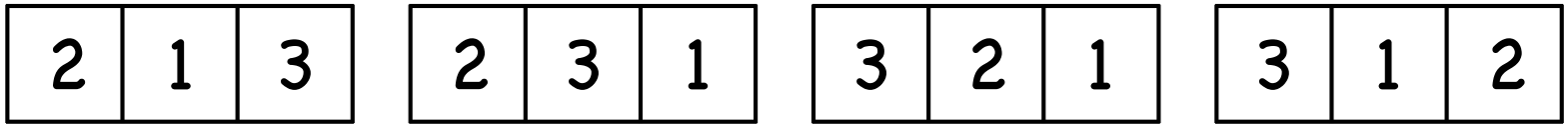
Ejemplo: Ordenamiento por Fuerza Bruta

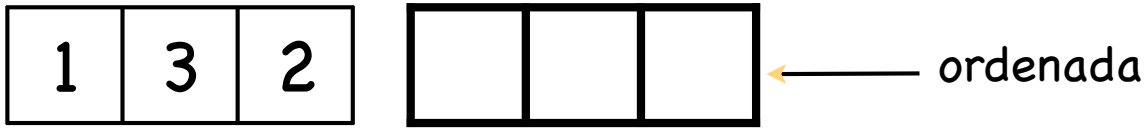
Caso extremo: Calcular permutaciones del arreglo de entrada, hasta conseguir una permutación que esté ordenada.

Ejemplo:



permutaciones

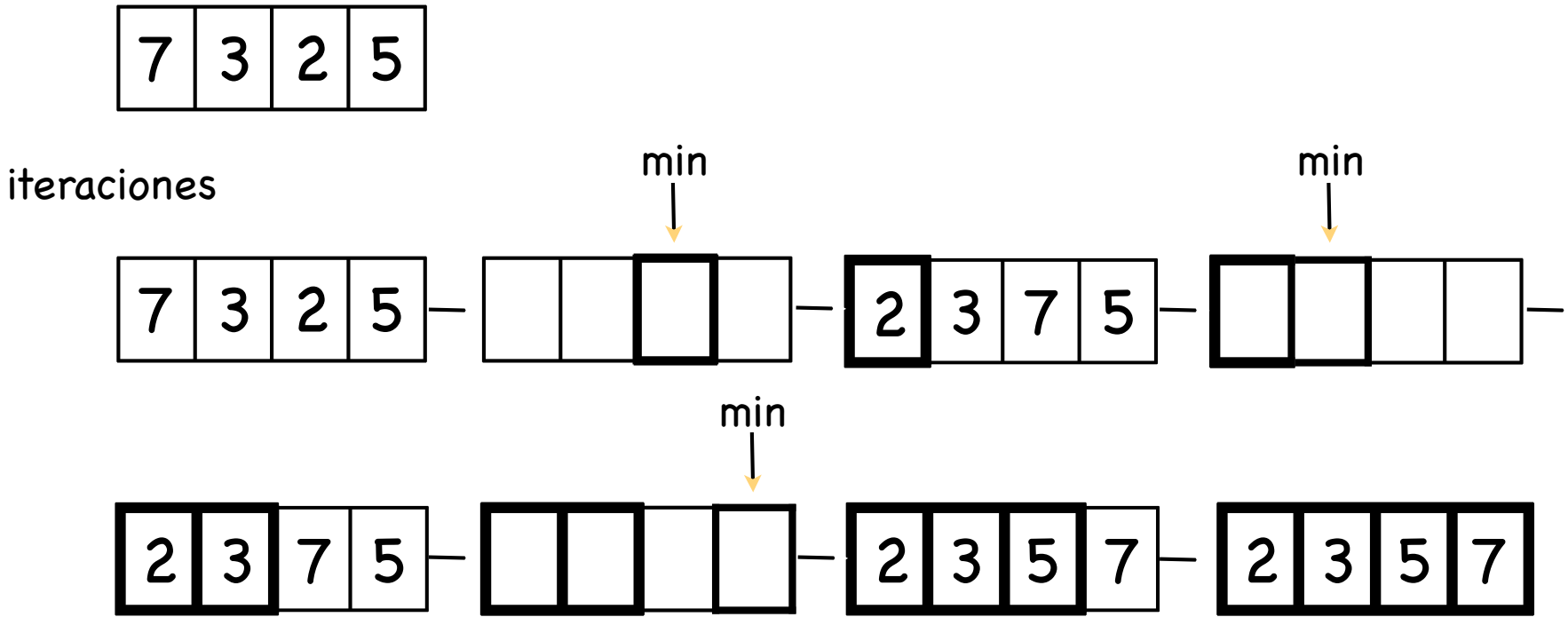




Ejemplo: Ordenamiento por Fuerza Bruta

Alternativa (Selection Sort): Visitar el arreglo hasta encontrar el elemento más chico, e intercambiarlo por el primer elemento. Luego, comenzando con el segundo elemento, buscar el elemento más chico a su derecha, e intercambiarlo con el segundo elemento. En general, en la iteración i (0<=i<=n-2), buscar el elemento más chico en A[i..n-1], e intercambiarlo con A[i].

Ejemplo:



Análisis de Selection Sort

```
Algoritmo SelectionSort(A[0..n-1])
   // Ordena un arreglo mediante el proceso de selección.
   // Entrada: un arreglo A[0..n-1] de elementos comparables
   // Salida: el arreglo A[0..n-1], ordenado de manera ascendente.
   for i ← 0 to n-2 do
      min ← i
      for j ← i+1 to n-1 do
         if A[j]<A[min] then min ← j
      swap(A[i],A[min])
```

- Cuál es la eficiencia en tiempo y espacio de este algoritmo?
- Es este algoritmo estable?
- Es este algoritmo correcto? (invariantes de ciclos?)

Ejemplo 2: Matching de Cadenas por Fuerza Bruta

Problema: Dadas un par de cadenas t y p, determinar si p es subcadena de t.

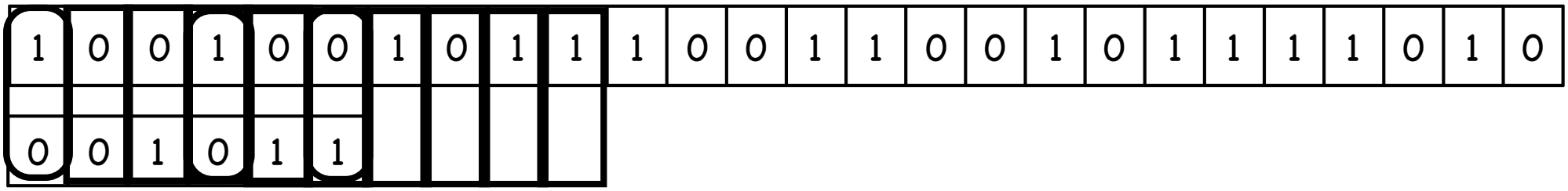
Solución por fuerza bruta:

- Paso 1 Alinear el patrón al comienzo del texto
- Paso 2 Moverse de izquierda a derecha, comparando caracter por caracter el texto (t) con el patrón (p), hasta que
  - todos los caracteres buscados se correspondan con el texto (búsqueda exitosa), o
  - se encuentre una discrepancia
- Paso 3 Mientras el patrón no se encuentre, y el texto no se haya agotado, realinear el patrón una posición hacia la derecha, y volver al paso 2

Ejemplo: Matching de Cadenas por Fuerza Bruta

Ejemplo:

t: 10010010111001100101111010 p: 001011



OK!

Análisis de String Matching por Fuerza Bruta

```
Algoritmo BruteForceStringMatch(char T[0..n-1], char P[0..m-1])
    // Chequea si P es una subcadena de T.
    // Entrada: un arreglo T[0..n-1] de n caracteres, que representa un texto, y un
    // arreglo P[0..m-1], que representa un patró a buscar en el texto.
    // Salida: el índice del primer caracter en el texto que comienza en una
    // subcadena que matchea con P. -1 si P no es subcadena de T.
    for i ← 0 to n-m do
        j ← 0
        while j < m and P[j] = T[i+j] do
            j ← j+1
        if j=m then return i
    endfor
    return -1
```

- Cuál es la eficiencia en tiempo?

Ejemplo 3:Encontrar el Par de Puntos Más Cercanos

Problema: Encontrar los dos puntos más cercanos en un conjunto de n puntos (en el plano cartesiano).

Solución por fuerza bruta:

- Computar la distancia entre cada par de puntos (diferentes) del conjunto y retornar los índices de aquellos puntos cuya distancia sea la menor.

Análisis de Cálculo de Pares Más Cercanos por Fuerza Bruta

Algoritmo BruteForceClosestPoints(P)
// Entrada: una lista (representación de conjuntos) P de n (n>=2) puntos (x_i,y_i).
// Salida: Los índices index1 e index2 del par de puntos de P más cercanos
// entre sí.
dmin ← +infinito
for i ← 1 **to** n-1 **do**
  for j ← i+1 **to** n **do**
    d ← sqrt($(x_i-x_j)^2 + (y_i-y_j)^2$)
    if d < dmin **then** dmin ← d; index1 ← i; index2 ← j
return index1, index2

- Cuál es la eficiencia en tiempo? Cómo podríamos mejorarlo?

Pros y Contras de la Fuerza Bruta

Pros

- amplia aplicabilidad
- simplicidad
- provee algoritmos razonables para una clase importante de problemas (e.g., multiplicación de matrices, ordenamiento, búsqueda, string matching)

Contras

- rara vez da como resultado algoritmos eficientes
  - algunos algoritmos de ordenamiento son inaceptablemente lentos
- no tan "constructiva" como otras técnicas de diseño

Búsqueda Exhaustiva

Una solución por fuerza bruta a un problema que involucra una búsqueda de un elemento con alguna propiedad particular, usualmente entre una cantidad combinatoria de objetos (e.g., permutaciones, combinaciones, subconjuntos de un conjunto, etc).

Método:

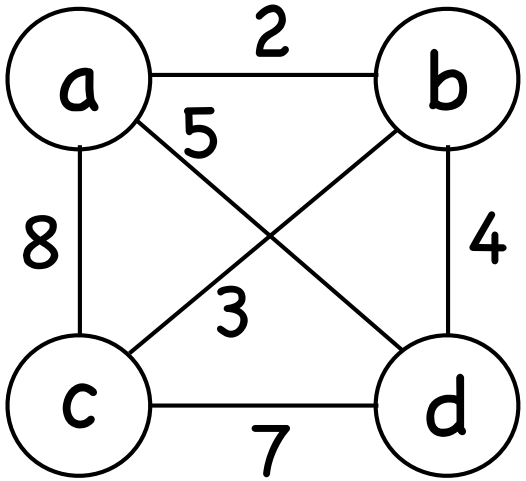
- generar una lista de todas las posibles soluciones al problema de una manera sistemática
- evaluar las potenciales soluciones una por una, desechando las que no sirvan; en el caso de problemas de optimización, se debe llevar cuenta del mejor objeto encontrado hasta el momento
- cuando se termine la búsqueda, retornar la solución encontrada

El Problema del Agente Viajero

Dadas n ciudades con distancias conocidas entre cada par, encontrar el tour más corto que pase por todas las ciudades exactamente una vez, y que vuelva al lugar de partida.

Si se observa que los datos de entrada pueden formalizarse como un grafo (no dirigido) conexo con pesos en los arcos, el problema equivale a encontrar el circuito hamiltoniano mínimo en el grafo.

Ejemplo:



El Problema del Agente Viajero

Para resolverlo por fuerza bruta, consideremos todos los circuitos posibles, y sus pesos correspondientes:

Tour	Costo
a→b→c→d→a	2+3+7+5 = 17
a→b→d→c→a	2+4+7+8 = 21
a→c→b→d→a	8+3+4+5 = 20
a→c→d→b→a	8+7+4+2 = 21
a→d→b→c→a	5+4+3+8 = 20
a→d→c→b→a	5+7+3+2 = 17

Las soluciones son los circuitos con costo 17.

El Problema de la Mochila

El problema de la mochila es un problema clásico de optimización. Consideremos el caso en el que contamos con n items, cada uno con peso w_i y valor v_i. Dada una mochila de capacidad (peso máximo soportado) W, encontrar el subconjunto de items que quepa en la mochila, y que maximice la suma de los valores del subconjunto.

Ejemplo: Supongamos que tenemos una mochila de capacidad W=16, y los siguientes items:

item	peso	valor
1	2	$20
2	5	$30
3	10	$50
4	5	$10

El Problema de la Mochila por Fuerza Bruta

Para resolverlo por fuerza bruta, consideremos todos los posibles subconjuntos de items:

Subconjunto	Peso total	Valor total
{1}	2	$20
{2}	5	$30
{3}	10	$50
{4}	5	$10
{1,2}	7	$50
{1,3}	12	$70
{1,4}	7	$30
{2,3}	15	$80
{2,4}	10	$40
{3,4}	15	$60
{1,2,3}	17	NO REALIZABLE
{1,2,4}	12	$60
{1,3,4}	17	NO REALIZABLE
{2,3,4}	20	NO REALIZABLE
{1,2,3,4}	22	NO REALIZABLE

Consideraciones Finales sobre Fuerza Bruta

Los algoritmos de búsqueda exhaustiva poseen en general tiempos de corrida razonables sólo para instancias pequeñas del problema a resolver.

En algunos casos, existen alternativas mucho mejores (e.g., caminos más cortos, árbol abarcador de costo mínimo, problemas de asignación, etc.).

En muchos casos, la búsqueda exhaustiva (o sus variantes) es la única forma conocida de resolver ciertos problemas de manera exacta.