Algoritmos y estructuras de datos

Ordenamiento por mezcla

Francisco Javier Zaragoza Martínez

Universidad Autónoma Metropolitana Unidad Azcapotzalco
Departamento de Sistemas

26 de julio de 2024

Citas

Robert A. Heinlein

Grok significa entender tan profundamente que el observador se vuelve una parte de lo observado; mezclar, combinar, casar, perder su identidad en la experiencia de grupo.

Lee Krasner

Yo mezclo lo que llamo lo orgánico con lo que llamo lo abstracto, que es lo que estás llamando lo geométrico.

Edward Norton Lorenz

Vemos que cada superficie es en realidad un par de superficies, de modo que, donde parecen mezclarse, en realidad hay cuatro superficies.

Problema abstracto

Dado un arreglo A con n elementos, se desea reacomodar sus elementos de modo que $A_0 \leq A_1 \leq \cdots \leq A_{n-1}$.

Problema concreto

Dado un arreglo `int a[MAX]` con `int n` elementos en las posiciones `a[0], ..., a[n-1]` se desea reacomodar sus elementos de modo que `a[0] <= a[1] <= ... <= a[n-1]`.

Ordenamiento interno

Un problema, mil soluciones

El problema de ordenamiento interno es uno de los más estudiados en la computación y para el que se conocen muchos algoritmos. Algunos son:

- Ordenamiento por cuenta (*Counting sort*).
- Ordenamiento de burbuja (*Bubble sort*).
- Ordenamiento por inserción (*Insertion sort*).
- Ordenamiento por mezcla (*Merge sort*).
- Ordenamiento por selección (*Selection sort*).
- Ordenamiento por partición (*Quicksort*).
- Ordenamiento por árbol (*Tree sort*).
- Ordenamiento por montículo (*Heapsort*).

Inserción directa

Idea: Inserta cada elemento en su lugar

Ejemplo

3 1 4 1 5 9 2 6 5 3 5 8 9 7 9 3

Inserción directa

Idea: Inserta cada elemento en su lugar

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3

Inserción directa

Idea: Inserta cada elemento en su lugar

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	**3**	4	1	5	9	2	6	5	3	5	8	9	7	9	3

Inserción directa

Idea: Inserta cada elemento en su lugar

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3

Inserción directa

Idea: Inserta cada elemento en su lugar

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3

Inserción directa

Idea: Inserta cada elemento en su lugar

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3

Inserción directa

Idea: Inserta cada elemento en su lugar

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3

Inserción directa

Idea: Inserta cada elemento en su lugar

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3
1	1	2	3	4	5	9	6	5	3	5	8	9	7	9	3

Inserción directa

Idea: Inserta cada elemento en su lugar

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3
1	1	2	3	4	5	9	6	5	3	5	8	9	7	9	3
1	1	2	3	4	5	6	9	5	3	5	8	9	7	9	3

Inserción directa

Idea: Inserta cada elemento en su lugar

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	3	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3
1	1	3	4	5	9	2	6	5	3	5	8	9	7	9	3
1	1	2	3	4	5	9	6	5	3	5	8	9	7	9	3
1	1	2	3	4	5	6	9	5	3	5	8	9	7	9	3
1	1	2	3	4	5	5	6	9	3	5	8	9	7	9	3

Inserción directa

Idea: Inserta cada elemento en su lugar

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	**3**	4	1	5	9	2	6	5	3	5	8	9	7	9	3
1	**3**	**4**	1	5	9	2	6	5	3	5	8	9	7	9	3
1	**1**	**3**	**4**	5	9	2	6	5	3	5	8	9	7	9	3
1	**1**	**3**	**4**	**5**	9	2	6	5	3	5	8	9	7	9	3
1	**1**	**3**	**4**	**5**	**9**	2	6	5	3	5	8	9	7	9	3
1	**1**	**2**	**3**	**4**	**5**	**9**	6	5	3	5	8	9	7	9	3
1	**1**	**2**	**3**	**4**	**5**	**6**	**9**	5	3	5	8	9	7	9	3
1	**1**	**2**	**3**	**4**	**5**	**5**	**6**	**9**	3	5	8	9	7	9	3
1	**1**	**2**	**3**	**3**	**4**	**5**	**5**	**6**	**9**	5	8	9	7	9	3

Tenemos un arreglo ordenado `a[0] <= a[1] <= ... <= a[j-1]` en el que queremos insertar `a[j]` en su lugar, el cual se encuentra buscando de atrás para adelante.

```
void inserta(int j, int a[]) {
  int t = a[j];
  for (int i = j-1; i >= 0 && a[i] > t; i--)
    a[i+1] = a[i];
  a[i+1] = t;
}
```

Insertamos los elementos `a[1]`, ..., `a[n-1]` en ese orden en su lugar.

```
void insercion(int n, int a[]) {
  for (int j = 1; j < n; j++)
    inserta(j, a);
}
```

Insertamos los elementos `a[1]`, ..., `a[n-1]` en ese orden en su lugar.

```
void insercion(int n, int a[]) {
  for (int j = 1; j < n; j++)
    inserta(j, a);
}
```

Cantidad de comparaciones

La llamada a `inserta(j, a)` hace a lo mucho `j` comparaciones. En total se hacen a lo mucho $1 + 2 + \cdots + (n-1) = \frac{1}{2}n(n-1) \approx \frac{1}{2}n^2$ comparaciones.

Inserción directa

Implementación recursiva

Se ordenan recursivamente con inserción directa los primeros $n - 1$ elementos y luego se inserta el elemento `a[n-1]` en su lugar. Nota que si $n = 1$ no se hace nada (caso base).

```
void insercion(int n, int a[]) {
  if (n > 1) {
    insercion(n-1, a);
    inserta(n-1, a);
  }
}
```

Se ordenan recursivamente con inserción directa los primeros $n-1$ elementos y luego se inserta el elemento `a[n-1]` en su lugar. Nota que si $n = 1$ no se hace nada (caso base).

```
void insercion(int n, int a[]) {
  if (n > 1) {
    insercion(n-1, a);
    inserta(n-1, a);
  }
}
```

Cantidad de comparaciones

Si `insercion(n, a)` hace $T(n)$ comparaciones, entonces $T(n) \leq T(n-1) + (n-1)$ si $n > 1$ y $T(1) = 0$. En total son $\leq (n-1) + \cdots + 1 = \frac{1}{2}n(n-1) \approx \frac{1}{2}n^2$ comparaciones.

1. Reescribe `insercion` para que ordene decrecientemente.
2. Reescribe `insercion` para que inicie con el último elemento.
3. Reescribe `insercion` como dos ciclos anidados.
4. ¿Cuántas veces se lee del arreglo `a`?
5. ¿Cuántas veces se escribe en el arreglo `a`?
6. Considere la secuencia $h_0 = 1$, $h_{i+1} = 3h_i + 1$ para $i \geq 0$ y considere la mínima k tal que $h_{k+1} \geq n$. Aplique el ordenamiento por inserción varias veces, considerando en cada vez elementos a distancia $h_k, \ldots, h_0$. A este algoritmo se le conoce como ordenamiento de Shell (*Shellsort*).

Ordenamiento por mezcla

Idea: Ordena cada mitad del arreglo e intercala sus elementos

Ejemplo

3 1 4 1 5 9 2 6 5 3 5 8 9 7 9 3

Ordenamiento por mezcla

Idea: Ordena cada mitad del arreglo e intercala sus elementos

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6								

Ordenamiento por mezcla

Idea: Ordena cada mitad del arreglo e intercala sus elementos

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6								
		¿	¿	?	?										

Ordenamiento por mezcla

Idea: Ordena cada mitad del arreglo e intercala sus elementos

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6								
		¿	¿	?	?										
1	1	2	3	4	5	6	9								

Ordenamiento por mezcla

Idea: Ordena cada mitad del arreglo e intercala sus elementos

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6								
		¿	¿	?	?										
1	1	2	3	4	5	6	9								
								5	3	5	8	9	7	9	3

Ordenamiento por mezcla

Idea: Ordena cada mitad del arreglo e intercala sus elementos

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6								
		¿	¿	?	?										
1	1	2	3	4	5	6	9								
								5	3	5	8	9	7	9	3
										¿	¿	?	?		

Ordenamiento por mezcla

Idea: Ordena cada mitad del arreglo e intercala sus elementos

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6								
		¿	¿	?	?										
1	1	2	3	4	5	6	9								
								5	3	5	8	9	7	9	3
										¿	¿	?	?		
								3	3	5	5	7	8	9	9

Ordenamiento por mezcla

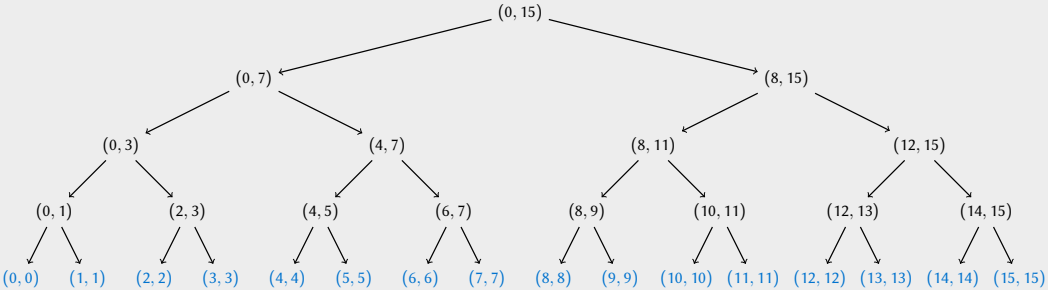
Idea: Ordena cada mitad del arreglo e intercala sus elementos

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6								
		¿	¿	?	?										
1	1	2	3	4	5	6	9								
								5	3	5	8	9	7	9	3
										¿	¿	?	?		
								3	3	5	5	7	8	9	9
1	1	2	3	3	3	4	5	5	5	6	7	8	9	9	9

Ordenamiento por mezcla

Árbol de recursión



Ordenamiento por mezcla

Implementación de la intercalación de dos mitades ordenadas

La primera mitad está de a[i] a a[m] y la segunda mitad está de a[m+1] a a[j].

```
void intercala(int i, int m, int j, int a[]) {
  int p = i, q = m+1, r = i;
  while (p <= m && q <= j) // mientras haya dos mitades
    if (a[p] <= a[q])      // menor en la primera mitad
      b[r++] = a[p++];     // b es un arreglo adicional
    else                   // menor en la segunda mitad
      b[r++] = a[q++];
  while (p <= m)           // resto de la primera mitad
    b[r++] = a[p++];
  for (p = i; p < r; p++)  // regresa todo al arreglo a
    a[p] = b[p];
}
```

Esta función deberá llamarse inicialmente como `mezcla(0, n-1, a)`. No se debe olvidar que `intercala` requiere un arreglo adicional `b` del mismo tamaño que `a`.

```
void mezcla(int i, int j, int a[]) {
  if (i < j) {                  // dos o mas elementos
    int m = i+(j-i)/2;          // calcula punto medio
    mezcla(i, m, a);            // ordena primera mitad
    mezcla(m+1, j, a);          // ordena segunda mitad
    intercala(i, m, j, a); // intercala las dos mitades
  }
}
```

Comparaciones de la unión

Con un vector de tamaño n, `intercala` hace $\leq n - 1$ comparaciones.

Comparaciones de la unión

Con un vector de tamaño n, `intercala` hace $\leq n - 1$ comparaciones.

Comparaciones de ordenamiento por mezcla

Si $T(n)$ es la cantidad de comparaciones que hace `mezcla` con un vector de tamaño n, entonces $T(1) = 0$ y $T(n) \leq 2T(n/2) + (n - 1)$ si $n > 1$.

Comparaciones de la unión

Con un vector de tamaño n, `intercala` hace $\leq n - 1$ comparaciones.

Comparaciones de ordenamiento por mezcla

Si $T(n)$ es la cantidad de comparaciones que hace `mezcla` con un vector de tamaño n, entonces $T(1) = 0$ y $T(n) \leq 2T(n/2) + (n-1)$ si $n > 1$.

Calculando $T(n)$ para las potencias de 2

$$\begin{array}{lcllll} T(2^1) & \leq & 2T(2^0) + (2-1) & = & 1 & = & 0 \cdot 2^1 + 1 \\ T(2^2) & \leq & 2T(2^1) + (4-1) & = & 5 & = & 1 \cdot 2^2 + 1 \\ T(2^3) & \leq & 2T(2^2) + (8-1) & = & 17 & = & 2 \cdot 2^3 + 1 \\ T(2^4) & \leq & 2T(2^3) + (16-1) & = & 49 & = & 3 \cdot 2^4 + 1. \end{array}$$

Calculando $T(n)$ para las potencias de 2

Demostremos por inducción en $k \geq 0$ que $T(2^k) \leq (k-1)2^k + 1$.

1. Para $k = 0$ se cumple pues $0 = T(2^0) \leq (0-1)2^0 + 1 = 0$.
2. Supongamos que para alguna $k \geq 0$ se cumple $T(2^k) \leq (k-1)2^k + 1$.
3. Entonces para $k+1$ tenemos que:

$$\begin{aligned} T(2^{k+1}) &\leq 2T(2^k) + 2^{k+1} - 1 \text{ (fórmula recursiva)} \\ &\leq 2[(k-1)2^k + 1] + 2^{k+1} - 1 \text{ (hipótesis)} \\ &= (k-1)2^{k+1} + 2 + 2^{k+1} - 1 \\ &= k2^{k+1} + 1. \end{aligned}$$

Esto es lo que queríamos demostrar.

Es decir, si $n = 2^k$, entonces $T(n) \approx n \log_2 n$.

Uso de memoria y cantidad de comparaciones

Propiedad	Burbuja	Inserción	Mezcla
Memoria	n	n	$2n$
Comparaciones	$\frac{1}{2}n^2$	$\frac{1}{2}n^2$	$n \log_2 n$

Tiempo de ejecución (1 ns por comparación)

n	Burbuja	Inserción	Mezcla
10^3	0.5 ms	0.5 ms	0.01 ms
10^4	50 ms	50 ms	0.1 ms
10^5	5 s	5 s	1 ms
10^6	500 s	500 s	20 ms

1. Escribe una función recursiva `int minimo(int i, int j, int a[])` que encuentre el mínimo de $(a_i, \ldots, a_j)$ usando la siguiente idea: Si $i = j$ entonces el mínimo es a_i. De lo contrario, calcula $m = \lfloor \frac{i+j}{2} \rfloor$ y resuelve recursivamente para $(a_i, \ldots, a_m)$ y $(a_{m+1}, \ldots, a_j)$. ¿Qué debes hacer con las respuestas de las llamadas recursivas?
2. Vuelve a hacerlo para ahora encontrar el máximo de $(a_i, \ldots, a_j)$.
3. Vuelve a hacerlo para ahora encontrar simultáneamente el mínimo y el máximo de $(a_i, \ldots, a_j)$.