Ordenamiento por mezcla

Función ordenamezcla(L, n)

- Si $n \leq 1$ entonces regresa L.
- Parte L en dos listas L_1 y L_2 del mismo tamaño ($n_1 = n_2 = n/2$).
- Regresa mezcla(ordenamezcla(L_1, n_1), ordenamezcla(L_2, n_2)).

- Vamos a hacer dos suposiciones:
  - Que n es una potencia de 2.
  - Que la función mezcla puede mezclar dos listas ordenadas con un total de n elementos en tiempo $O(n)$.
- Sea $T(n)$ el tiempo de ejecución de ordenamezcla(L, n). Entonces

$$T(n) = \begin{cases} c & \text{si } n = 1 \\ 2T(n/2) + dn & \text{en otro caso} \end{cases}$$

para algunas constantes c y d.

Sustitución repetida

- Suponga que $n > 1$. Entonces:

$$\begin{aligned} T(n) &= 2T(n/2) + dn \\ &= 2(2T(n/4) + dn/2) + dn \\ &= 4T(n/4) + 2dn \\ &= 4(2T(n/8) + dn/4) + 2dn \\ &= 8T(n/8) + 3dn. \end{aligned}$$

- Aparece el patrón $T(n) = 2^i T(n/2^i) + idn$.
- Tomando $i = \log_2 n$ se tiene que

$$T(n) = 2^{\log_2 n} T(n/2^{\log_2 n}) + dn \log_2 n = dn \log_2 n + cn.$$

- Por lo tanto $T(n) \in O(n \log n)$.
- De nuevo, esto no es una demostración.

Algoritmo genérico

- Sea S una lista de n enteros distintos.

Función quicksort(S)

- Si $|S| \leq 1$ entonces
  - Regresa S
- Si no entonces
  - Escoge un elemento a de S ¿cómo?
  - Sean $S_<$, $S_=$ y $S_>$ los elementos de S que son $< a$, $= a$ y $> a$ respectivamente
  - Regresa (quicksort($S_<$), $S_=$, quicksort($S_>$))

- El elemento a se llama pivote.

Análisis del peor caso

- Quicksort es ligeramente mejor que ordenamiento por mezcla en el caso promedio.
- ¿Pero qué pasa en el peor caso?
- Por ejemplo, si $i = 1$ a cada paso entonces $T(1) = 0$ y $T(n) = T(n-1) + n - 1$ para $n \geq 2$.
- En este caso se puede demostrar que $T(n) \in \Theta(n^2)$.
- Así que quicksort es mucho peor que ordenamiento por mezcla en el peor caso.

Análisis del mejor caso

- Por ejemplo, si $i = n/2$ a cada paso entonces $T(1) = 0$ y para $n > 1$

$$T(n) = 2T(n/2) + n - 1.$$

- En este caso es fácil demostrar que $T(n) = n \log n + O(n)$.
- Por lo tanto, quicksort hace sólo 39 % más comparaciones en el caso promedio que en el mejor caso.

Complejidad del ordenamiento

- ¿Qué tan rápido se pueden ordenar n elementos?
- Eso depende de qué operaciones se permita hacer con ellos.
- Quicksort y ordenamiento por mezcla hacen comparaciones.
- Demostraremos que los algoritmos de ordenamiento basados en comparaciones deben hacer al menos $\Omega(n \log n)$ comparaciones.
- Existen algoritmos sencillos que hacen $O(n)$ operaciones (de otro tipo) para ordenar n elementos.

Cota inferior

- Todo algoritmo de ordenamiento debe ordenar su entrada.
- La entrada puede venir ordenada de $n!$ formas distintas.
- El algoritmo no sabe cuál de esas entradas es.
- Como cada comparación sólo puede tener dos valores posibles, el espacio de búsqueda se reduce a lo mucho a la mitad con cada una.
- Por lo tanto, si el algoritmo puede ordenar en un máximo de $T(n)$ comparaciones entonces

$$2^{T(n)} \geq n!$$

- Es decir $T(n) \geq \log n!$

Aproximando $\log n!$

- Observe que

$$(n!)^2 = (1 \cdot 2 \cdots n)(n \cdots 2 \cdot 1) = \prod_{k=1}^{n} k(n+1-k).$$

- La expresión $k(n+1-k)$ toma su valor mínimo cuando $k = 1$ o $k = n$ y toma su valor máximo cuando $k = (n+1)/2$.
- Por lo tanto

$$\prod_{k=1}^{n} n \leq (n!)^2 \leq \prod_{k=1}^{n} \frac{(n+1)^2}{4}.$$

- Es decir $n^{n/2} \leq n! \leq (n+1)^n / 2^n$.
- Equivalentemente $\frac{1}{2} n \log n \leq \log n! \leq n \log \frac{n+1}{2}$.
- Por lo tanto $\log n! \in \Theta(n \log n)$.

Conclusión

- Entonces $T(n) \in \Omega(n \log n)$.
- Por lo tanto, cualquier algoritmo de ordenamiento basado en comparaciones debe hacer $\Omega(n \log n)$ comparaciones en el peor caso.
- Ordenamiento por mezcla cumple con la cota pero quicksort es peor.
- También se puede demostrar que cualquier algoritmo de ordenamiento basado en comparaciones debe hacer $\Omega(n \log n)$ comparaciones en el caso promedio.
- Ordenamiento por mezcla y quicksort cumplen con esta cota.