Contenido

Contenido

Conteo de combinaciones

- Para escoger r objetos de entre n
  - se escoge el primer objeto y después se escogen $r-1$ objetos de entre los $n-1$ restantes o
  - no se escoge el primer objeto y después se escogen r objetos de entre los $n-1$ restantes.
- Por lo tanto
$$\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r}.$$
- El caso base es cuando $r = 0$ o $r = n$.

Tiempo de ejecución del algoritmo de burbuja

Función burbuja(A, n)

- Para $i \leftarrow 1$ a $n-1$ haz:
  - Para $j \leftarrow 1$ a $n-i$ haz:
    - Si $A_j > A_{j+1}$ entonces intercambia A_j con A_{j+1}.

- La llamada, la decisión y el regreso cuestan $O(1)$ cada una.
- El ciclo interno se ejecuta $n-i$ veces y cuesta $O(n-i)$.
- El ciclo externo cuesta
$$O\left(\sum_{i=1}^{n-1}(n-i)\right) = O\left(\sum_{i=1}^{n-1} i\right) = O\left(\frac{n(n-1)}{2}\right) = O(n^2).$$
- También se puede probar que toma tiempo $\Omega(n^2)$ y por lo tanto tiempo $\Theta(n^2)$.

Facilitando el análisis

- En principio, se podría calcular exactamente el tiempo o el número de pasos necesarios para ejecutar un algoritmo.
- Pero esto es generalmente innecesario.
- En lugar de esto, identifica la operación fundamental que se usa en el algoritmo.
- Generalmente el tiempo de ejecución es un múltiplo constante de la cantidad de operaciones fundamentales.
- Así que no hay necesidad de hacer un análisis línea por línea.
- Sólo es necesario calcular exactamente la cantidad de operaciones fundamentales.

Ejemplo

- En el ejemplo del algoritmo de la burbuja, la operación fundamental es la comparación.
- El tiempo de ejecución será un factor constante del número total de comparaciones.
- La decisión hace 1 comparación.
- El ciclo interno hace $n-i$ comparaciones.
- El ciclo externo hace $\sum_{i=1}^{n-1}(n-i)$ comparaciones.
- En total se hacen $n(n-1)/2$ comparaciones.
- Esto es $O(n^2)$.

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 $T(n) \geq \log n!$

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.