Data Structures and Algorithms

Spring 2009-2010

Outline

1. Sorting Algorithms (contd.)
   - $o(n^2)$ Algorithms
   - Solving Divide-and-Conquer and the Master Theorem

Outline

1. Sorting Algorithms (contd.)
   - $o(n^2)$ Algorithms
   - Solving Divide-and-Conquer and the Master Theorem

Heapsort

- Have seen that we can build a heap in linear time using $n/2$ calls of `trickleDown()`
- If we now call `deleteMin()` n times, we will get the elements in smallest-to-largest (non-decreasing) order
- Where to store them as we peel them off?
- Do we need an extra array?
- No. Each time a `deleteMin()` is called, place the result in the slot just opened up by the shrunk array
- Problem with this is that after n `deleteMin()`s the elements now appear largest-to-smallest (non-increasing)
- Two solutions:
  1. Use a (max)heap with the `deleteMax()` operation
  2. After the n `deleteMin()`s *reverse* the contents of the array to get it smallest-to-largest; reversing an array can be done in $O(n)$-time using just $O(1)$ additional storage

Mergesort

- Mergesort strategy: To sort an array of numbers
  - Sort the first half of the array
  - Sort the second half of the array
  - Merge the two halves of the array
- Running time analysis:
  - Sorting an array comprising one element takes unit time
  - To sort n numbers we perform two sorts on $n/2$ numbers and then merge the resulting arrays
  - Merging two arrays of length $n/2$ each takes time n since we can make one

Mergesort (contd.)

- Recurrence relation:

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(n/2) + n \end{aligned}$$

- Assume n is a power of 2 ($n = 2^k, k = \log n$)
- Can argue then, that if n is not a power of 2, it falls between two powers of two: $2^{\lfloor k \rfloor} < n < 2^{\lceil k \rceil}$
- Two techniques to solving the recurrence

$$T(n) = 2T(n/2) + n$$

Mergesort Solution 1

Divide through by n,

$$\frac{T(n)}{n} = \frac{T(n/2)}{\frac{n}{2}} + 1$$

and letting $T'(n) = T(n)/n$

$$\begin{aligned} T'(n) &= T'(n/2) + 1 \\ &= T'(n/4) + 1 + 1 \\ &\vdots \\ &= T'(1) + \underbrace{1 + \cdots + 1}_{k} \end{aligned}$$

Thus,

$$\begin{aligned} T'(n) &= T(n)/n = 1 + \log n \\ T(n) &= n + n \log n = O(n \log n) \end{aligned}$$

Mergesort Solution 2

$$\begin{aligned} T(n) &= 2T(n/2) + n, \quad \text{and so,} \\ T(n/2) &= 2T(n/4) + n/2 \end{aligned}$$

Therefore

$$\begin{aligned} T(n) &= 2(2T(n/4) + n/2) + n \\ &= 4T(n/4) + 2n \\ &\vdots \\ T(n) &= 2^k T(n/2^k) + k \times n \end{aligned}$$

When $k = \log n$, $n/2^k = 1$ (since $n = 2^k$ for some k)

$$\begin{aligned} T(n) &= nT(1) + n \log n \\ &= n + n \log n \\ &= O(n \log n) \end{aligned}$$

Did you know?

$$
\begin{aligned}
a^{\log_c n} = (c^{\log_c a})^{\log_c n} &= c^{\log_c a \log_c n} \\
&= c^{\log_c n \log_c a} = (c^{\log_c n})^{\log_c a} = n^{\log_c a}
\end{aligned}
$$

In summary

$$a^{\log_c n} = n^{\log_c a}$$

Outline

1. Sorting Algorithms (contd.)
   - $o(n^2)$ Algorithms
   - Solving Divide-and-Conquer and the Master Theorem

General Solutions to Divide and Conquer

- How do we solve recurrences of the form

$$
\begin{aligned}
T(n) &= aT(\frac{n}{c}) + bn \\
T(1) &= b
\end{aligned}
$$

- Since we are dividing by c, we will assume that n is a power of c; that is, $n = c^k$ or, $k = log_c n$
- Of course, this is not the case in general but it is **always** the case that n is sandwiched between two powers of c: $c^k \leq n \leq c^{k+1}$

General Solutions to Divide and Conquer (contd.)

$$
\begin{aligned}
T(n) &= aT(\frac{n}{c}) + bn \\
&= a[aT(\frac{n}{c^2}) + b\frac{n}{c}] + bn \\
&= a^2T(\frac{n}{c^2}) + b\frac{a}{c}n + bn \\
&= a^2[aT(\frac{n}{c^3}) + \frac{bn}{c^2}] + b\frac{a}{c}n + bn \\
&= a^3T(\frac{n}{c^3}) + b\frac{a^2}{c^2}n + b\frac{a}{c}n + bn \\
&\vdots \\
&= a^{\log_c n}T(1) + bn[\frac{a^{log_c n-1}}{c^{log_c n-1}} + \cdots + \frac{a}{c} + 1]
\end{aligned}
$$

General Solutions to Divide and Conquer (contd.)

$$
\begin{aligned}
T(n) &= a^{\log_c n} T(1) + bn \sum_{i=0}^{log_c n-1} \left(\frac{a}{c}\right)^i \\
&= a^{\log_c n} b \frac{n}{n} + bn \sum_{i=0}^{\log_c n-1} \left(\frac{a}{c}\right)^i, \qquad \text{since } T(1) = b \\
&= bn \frac{a^{\log_c n}}{c^{\log_c n}} + bn \sum_{i=0}^{\log_c n-1} \left(\frac{a}{c}\right)^i, \qquad \text{since } c^{\log_c n} = n \\
&= bn \sum_{i=0}^{\log_c n} \left(\frac{a}{c}\right)^i \\
&= bn \left[\frac{\left(\frac{a}{c}\right)^{\log_c n+1} - 1}{\frac{a}{c} - 1} \right], \qquad \frac{a}{c} \neq 1
\end{aligned}
$$

General Solutions to Divide and Conquer (contd.)

- There are three cases to consider with this expression for $T(n)$
  1. $a = c$
  2. $a < c$
  3. $a > c$
- In each case we will reduce the expression as much as we can and then argue that since n is always bounded from above and below by a power of c, then $T(n)$ will be "Θ" of the function of n that we derive.
- See also the Master Theorem page on Wikipedia

General Solutions to Divide and Conquer (contd.)

Case ❶ $\frac{a}{c} = 1$ $(a = c)$ then

$$T(n) = bn\,[\log_c n + 1] = \Theta(n \log_c n)$$

General Solutions to Divide and Conquer (contd.)

Case ❷ $\frac{a}{c} < 1$ $(a < c)$ then

$$\lim_{n\to\infty} \left(\frac{a}{c}\right)^{\log_c n+1} \to 0$$

$$\begin{aligned} bn\left[\frac{\left(\frac{a}{c}\right)^{\log_c n+1} - 1}{\frac{a}{c} - 1}\right] &= bn\left[\frac{1 - \left(\frac{a}{c}\right)^{\log_c n+1}}{1 - \frac{a}{c}}\right] \\ &\approx bn\left[\frac{1}{\frac{c-a}{c}}\right] \\ &= bn\frac{c}{c-a} \\ &= \Theta(n) \end{aligned}$$

General Solutions to Divide and Conquer (contd.)

Case ❸ $\frac{a}{c} > 1$ $(a > c)$ then

$$\lim_{n\to\infty}\left(\frac{a}{c}\right)^{\log_c n+1} \to \infty$$

so the remaining terms are insignificant, and the running time should grow very rapidly

Using the "Did you know?" fact, $a^{\log_c n} = n^{\log_c a}$

$$\begin{aligned} T(n) &= bn\left(\frac{a}{c}\right)^{\log_c n+1} \\ &= bn\frac{a}{c}\left(\frac{a}{c}\right)^{\log_c n} \\ &= \frac{ab}{c}n\frac{n^{\log_c a}}{n} = \Theta(n^{\log_c a}) \end{aligned}$$