COMP 250

Lecture 14

recurrences 2: mergesort & quicksort
logarithms: some review, some new

Oct. 12, 2016
Recall: Mergesort

partition

mergesort

merge

mergesort
mergesort(list){
    if list.length == 1
        return list
    else{
        mid = (list.size - 1) / 2
        list1 = list.getElements(0,mid)
        list2 = list.getElements(mid+1, list.size-1)
        list1 = mergesort(list1)
        list2 = mergesort(list2)
        return merge( list1, list2 )
    }
}

\[ t(n) = c \cdot n + 2 \cdot t \left( \frac{n}{2} \right) \]
What if $n$ is not even?

e.g. $t(13) = c \times 13 + t(6) + t(7)$

In general, one should write the recurrence as:

$$t(n) = c \times n + t\left(\lfloor \frac{n}{2} \rfloor\right) + t\left(\lceil \frac{n}{2} \rceil\right)$$

round down                       round up

In COMP 250, one typically assumes $n = 2^k$ for recurrences that involve $t\left(\frac{n}{2}\right)$.

The more general recurrence has roughly the same solution.

See A2 Q2a,b where you should assume $n = 2^k$. 
\begin{align*}
t(n) &= 2t\left(\frac{n}{2}\right) + cn \\
&= 2 \left( 2t\left(\frac{n}{4}\right) + c\frac{n}{2} \right) + cn \\
&= 4 \cdot t\left(\frac{n}{4}\right) + cn + cn \\
&= 4 \left( 2t\left(\frac{n}{8}\right) + c\frac{n}{4} \right) + cn + cn \\
&= 8 \cdot t\left(\frac{n}{8}\right) + cn + cn + cn \\
&= 2^k \cdot t\left(\frac{n}{2^k}\right) + cnk \\
&= n \cdot t(1) + c \cdot n \log n, \quad \text{when } n = 2^k
\end{align*}
Recall this slide from lecture 12:

Q: How many operations are required to mergesort a list of size \( n \)?

A: \( O( n \log_2 n ) \)
Recall from lecture 12.

Computers perform \( \sim 10^9 \) operations per second.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( n \log_2 n )</th>
<th>( n^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 10^3 )</td>
<td>( 10^4 )</td>
<td>( 10^6 )</td>
</tr>
<tr>
<td>( 10^6 )</td>
<td>( \sim 10^7 )</td>
<td></td>
</tr>
<tr>
<td>( 10^9 )</td>
<td>( \sim 10^{10} )</td>
<td></td>
</tr>
</tbody>
</table>
Quicksort

pivot

partition

concatenate

quicksort

1 2 3 4 5 6 7

1 10 11 13 16

10 11 13 16

8

10 3 1 7 2 5 4

3 6 1 7 2 5 4

1 2 3 4 5 6 7

1 2 3 4 5 6 7 8

8

10 3 1 7 2 5 4

3 6 1 7 2 5 4

1 2 3 4 5 6 7
Quicksort worst case example

<table>
<thead>
<tr>
<th>1</th>
<th>10</th>
<th>8</th>
<th>11</th>
<th>6</th>
<th>3</th>
<th>7</th>
<th>16</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>10</th>
<th>11</th>
<th>13</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>pivot</td>
<td>quicksort</td>
<td>quicksort</td>
<td>concatenate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

partition

Quicksort worst case

\[ t(n) = cn + t(n - 1) \]

From last lecture....
Quicksort worst case

\[ t(n) = c \cdot n + t(n - 1) \]

From last lecture....

\[ t(n) = c \cdot \frac{n(n + 1)}{2} \]

which is \( O(n^2) \)
So why is quicksort “quick” (in practice)?

• It can be done “in place”

• Worst case can be made very rare, as follows...
How to reduce the chances of worst case? (unbalanced partition)

“Median of three” technique:

Let the pivot be the median of first, middle, and last elements in list.

e.g. \( \text{pivot} = \text{median}(1, 13, 4) = 4 \)
(size-1)/2

\[
\begin{array}{c}
0 \\
10 \\
8 \\
11 \\
6 \\
13 \\
7 \\
16 \\
2 \\
5 \\
3 \\
\text{pivot 4}
\end{array}
\]

\[
\begin{array}{c}
1 \\
2 \\
3 \\
\text{partition} \\
4 \\
10 \\
8 \\
11 \\
6 \\
13 \\
7 \\
16 \\
5
\end{array}
\]

Works well in practice, especially if the list is already nearly sorted.

(Details omitted.)
Logarithms
(review)

Definition: for any $b > 0$, the log function is the inverse of the exponential function.

$$\log_b b^x \equiv x$$

$$b^{\log_b x} \equiv x$$
Some Useful Properties (assume \( a, b, c > 0 \))

\[
\log_b a^c = c \log_b a
\]

\[
\log_b (ac) = \log_b a + \log_b c
\]

\[
\log_b a = (\log_b c)(\log_c a)
\]

\[
a^{\log_b c} \equiv c^{\log_b a}
\]

Exercise: Try to prove these relationships using the definition on the previous slide. (See lecture notes for proofs.)
Midterm test 1

Mean: 81%
Median: 85%

About 10% did not write the exam.