CS 340 - lecture 20 - Oct 23

Master Theorem Let $a \geq 1$, $b > 1$, $f: \mathbb{N} \to \mathbb{N}$, $T: \mathbb{N} \to \mathbb{N}$ with $T(N) = a \cdot T(\frac{N}{b}) + f(N)$

Let $z = \log_b a$.

(1) If $f(N) = O(N^x)$ for some $x < z$, then $T(N) = \Theta(N^z)$

(2) If $f(N) = \Theta(N^z)$, then $T(N) = \Theta(N^z \log N)$

(3) If $f(N) = \Omega(N^x)$ for some $x > z$ and if there are $n_0 \in \mathbb{N}$, $c < 1$ such that $a \cdot f(\frac{N}{b}) \leq c \cdot f(N)$ for all $N \geq n_0$, then $T(N) = \Theta(f(N))$

Example 36 continued.

(b) $T(N) = 8 \cdot T(\frac{N}{2}) + 1000 \cdot N^2$

$a = 8$, $b = 2$, $f(N) = 1000 \cdot N^2$

$z = \log_b a = \log_2 8 = 3$

$f(N) = \Theta(N^2)$

$f(N) = O(N^x)$ for some $x < z$ (here $x = 2$, $z = 3$)

M.T. (1) $\Rightarrow$ $T(N) = \Theta(N^z) = \Theta(N^3)$.

(c) $T(N) = 3 \cdot T\left(\frac{N}{4}\right) + N \log N$

$a = 3,\ b = 4,\ f(N) = N \log N$

$z = \log_b a = \log_4 3 \approx 0.793$

$f(N) = \Omega(N) = \Omega(N^1)$, $1 > z$

$f(N) = \Omega(N^x)$ for some $x > z$, namely $x = 1$

$$a \cdot f\left(\frac{N}{b}\right) = 3 \cdot \frac{N}{4} \log \frac{N}{4} \leq \frac{3}{4} N \log N$$
$$= c \cdot N \log N$$

for some $c < 1$, namely $c = \frac{3}{4}$.

M.T. (3) $\Rightarrow T(N) = \Theta(f(N)) = \Theta(N \log N)$.

(d) $T(N) = T\left(\frac{2N}{3}\right) + 1$

$a = 1,\ b = \frac{3}{2},\ f(N) = 1$

$z = \log_b a = \log_{3/2} 1 = 0$

$f(N) = 1 = N^0 = N^z = \Theta(N^z)$

M.T. (2) $\Rightarrow T(N) = \Theta(N^z \log N) = \Theta(N^0 \log N) = \Theta(\log N)$

3.5. QUICKSORT

Fastest known generic sorting alg. in practice, for C++.

$T_{avg}(N) = O(N \log N)$

$T_{worst}(N) = O(N^2)$, but worst cases are "unlikely" to occur

Algorithmic steps : QuickSort(list)

(1) If length(list) ≤ 1, return

(2) p = pivot element in list

(→ various methods for picking pivot are possible)

(3) listLeft = array of all l in list with l<p
k elements

listRight = array of all l in list with l>p
N − k − 1 elements

4) return list, where

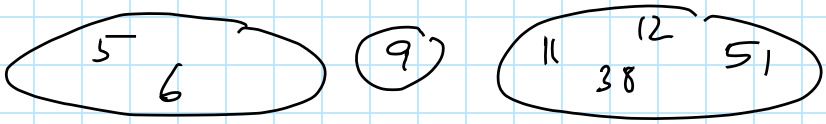
(list[0], ..., list[k-1]) = QuickSort(listLeft)

list[k] = p

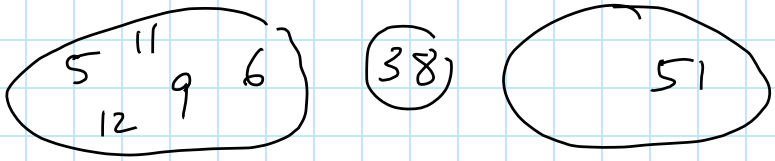
(list[k+1], ..., list[N-1]) = QuickSort(listRight)

e.g., list 5, 11, 38, 9, 12, 51, 6

suppose p = 9



p = 38



How to pick the pivot p?

(a) first element in list ?

not a good choice if input is "almost sorted"

(b) randomly chosen

safe strategy,

(c) middle element : pivot = list[(first + last)/2]

where first = leftmost index → of subarray to be sorted
last = rightmost index

(d) median-of-three :

pivot = median of list[first]
list[last]
list[(first + last)/2]

(what to do if list has only 2 elements ?)

How to do Step (3) (Partitioning) ?

3.1. Swap pivot into list[last] // hide pivot

3.2. i = first ; j = last - 1 ;

3.3. while (i ≤ j)

```
{
    while ((i ≤ last) and (list[i] < pivot)) {i++};
    while ((j ≥ first) and (list[j] ≥ pivot)) {j--};
    if (i < j)   swap (list[i], list[j]);
}
```

3.4. swap (list[i], list[last])