QuickSort

Note Title 10/15/2007

This is another example of a “divide and conquer” algorithm.

Step 1 (divide)

Select a “pivot” value and logically partition the list into two sub-lists:

L1: values less than the pivot

L2: values greater than the pivot

Your list is now: $\boxed{\text{L1,pivot,L2}}$

P

Step 2 (conquer)

Sort L1 and L2

SORTED!

QuickSort PseudoCode

Algorithm

Let's assume that our list L is held in an array and that we want to use as little extra space as possible.

```
QuickSort(array L, int first, int last) {
    if (first<last) {
        pivotpos = Partition(L,first, last)
        QuickSort(L, first, pivotpos-1)
        QuickSort(L,pivotpos+1,last);
    }
}
```

We will abbreviate pivotpos as pos_p.

NOTE: We would still need to write the partition algorithm. The easiest thing to code would probably be to pick the last value in the list as the pivot and then partition based on that.

Let's trace some examples

5, 7, 6, 1, 3, 2, 4 ⟸ Input

1. Choose 1

[1] 5 7 6 3 2 4

2. Choose 4

1 3 2 [4] 5 7 6

1 [2] 3   5 [6] 7

1 3 5 7

1, 2, 3, 4, 5

What is partition's runtime?

There are many ways to implement the partition algorithm, but in terms of data comparisons, what should its runtime be?

$n-1$

MORAL: Partitioning always takes n-1 time, but depending on how smart we are, the choice of pivot can impact the overall RT.

LAST

What is QuickSort's Runtime?

Start with $T(0) = T(1) = 0$

For the recurrence, what is:

- The worst case split?
- The best case split?
- The average/expected runtime?

Worst case: sorted list

$pos_p = 1$
$T(n) = T(n-1) + T(0) + \Theta(n)$

$$\max_{1 \le pos_p \le n} \Big[\underbrace{T(n - pos_p)}_{\text{upper part of list}} + \underbrace{T(pos_p - 1)}_{\text{lower part of list}} \Big] + \underbrace{\Theta(n)}_{\text{partition}}$$

$pos_p = n$
$T(n) = T(0) + T(n-1) + \Theta(n)$

For worst case, we have to assume the least amount of division to happen, so ....

$\downarrow$

Imagine $pos_p = 1$ on each iteration.

$T(n) = T(n-1) + T(0) + \Theta(n)$

$\Rightarrow T(n) \in \Theta(n^2)$

$$O(n^2) \text{ work} \left\{ \begin{array}{l} n \\ n-1 \\ n-2 \\ \vdots \\ 1 \end{array} \right.$$

What about <u>Best Case</u>?

Perfect Pivot

$$T(0) = T(1) = 0$$
$$T(n) = 2 \cdot T(n/2) + \Theta(n)$$

$$T(n) \in \Theta(n \log n)$$

(by Master Theorem
or Recursion Tree)

What about average case?

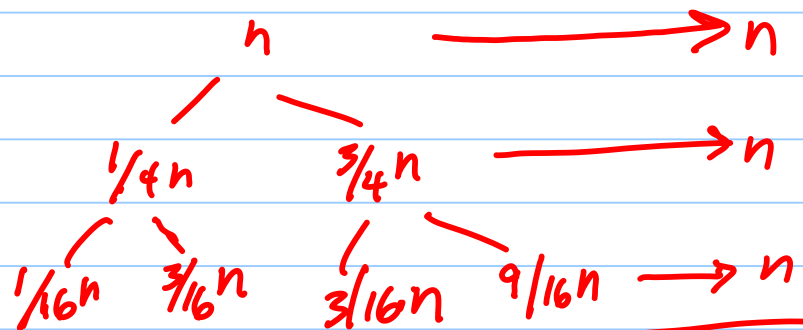
Look at imbalanced cases.

Okay, what about some <u>very</u> unbalanced cases?

Case 1: 75% / 25% split

$$T(n) = T(\overset{\frac{3}{4}}{\cancel{.75}}\, n) + T(\overset{\frac{1}{4}}{\cancel{.25}}\, n) + \Theta(n)$$

$$\Longrightarrow T(n) \in \Theta(n \log n)$$



levels $= \log_{\frac{4}{3}} n \Longrightarrow \boxed{n \log_{4/3} n}$

Case 2: What if it's as bad as 99% / 1%?

$$T(n) = T(.99n) + T(.01n) + \Theta(n)$$

levels $= \log_{\frac{100}{99}} n$

$$T(n) \in \Theta(n \log n)$$

But what about on average?

Average Case Analysis

Let's return to the idea of expected values.

Let's assume that every "division situation" is equally likely.

If we let ***pos$_p$*** represent the position of p, then we could represent the expected runtime as being:

$$T(n) = (n-1) + \frac{\sum_{pos_p=1}^{n}\left[T(pos_p - 1) + T(n - pos_p)\right]}{n}$$

equally likely.

Can we simplify this?

$$T(n) = \frac{\sum_{pos_p=1}^{n} \left[T(pos_p - 1) + T(n - pos_p)\right]}{n} + (n-1)$$

$$= \frac{\sum_{pos_p=1}^{n} T(pos_p - 1) + \sum_{pos_p=1}^{n} T(n - pos_p)}{n} + (n-1)$$

i is the new name for pos_p

Note: $\sum_{i=1}^{5} T(i-1) \Rightarrow T(0) + T(1) + \ldots + T(4)$

$$\boxed{\sum_{pos_p=0}^{n-1} T(pos_p)}$$

Note: $\sum_{i=1}^{5} T(5-i) \Rightarrow T(4) + T(3) + \ldots + T(0)$

$$\boxed{\sum_{pos_p=0}^{n-1} T(pos_p)}$$

$$= \frac{\sum_{pos_p=0}^{n-1} T(pos_p) + \sum_{pos_p=0}^{n-1} T(pos_p)}{n} + (n-1)$$

$$= \frac{2 \sum_{pos_p=0}^{n-1} T(pos_p)}{n} + (n-1)$$

$$= \frac{2}{n} \sum_{pos_p=0}^{n-1} T(pos_p) + (n-1)$$

What if we assume $T(n)$ is less than $c \cdot n \cdot \log n$?

IH $\forall\, 0 \le i < k \quad T(i) \le c \cdot i \cdot \log i$

Let i represent pos_p.

IS show $T(k) \le c \cdot k \cdot \log k$

$$\frac{2}{k} \sum_{i=0}^{k-1} T(i) + (k-1) \overset{?}{\le} c \cdot k \cdot \log k$$

$$\frac{2}{k} \sum_{i=0}^{k-1} T(i) \overset{?}{\le} c \cdot k \log k \underbrace{- (k-1)}_{-k+1}$$

Use IH

New Goal

$$\frac{2}{k} \sum_{i=0}^{k-1} c \cdot i \cdot \log i \overset{?}{\le} c\, k \log k - k + 1$$

How can we simplify?

$\downarrow$

$$\frac{2}{k}\sum_{i=0}^{k-1} c\cdot i\cdot \log i$$

- Summation overestimation
- Integration overestimation

Let's try integration.

We know $\displaystyle\int_a^b x \ln x \, dx = \frac{1}{4}x^2(2\ln x - 1)\Big]_a^b$

Note: $\log 0$ undefined, but $T(0)=0$, so we can make summation start at $\underline{\underline{1}}$

Also, change from generic base to $\log_e$

$$\frac{2c}{k}\sum_{i=1}^{k-1} i \ln i \overset{\checkmark}{\leq} \frac{2c}{k}\int_1^k i \ln i \, di$$

$$\frac{2c}{k}\int_1^k i \ln i \, di$$

$$= \frac{2c}{k}\left(\frac{1}{4}i^2(2\ln i - 1)\right)\Big]_1^k$$

$$= \frac{2c}{k}\left(\frac{1}{4}k^2(2\ln k - 1) - \frac{1}{4}(-1)\right)$$ plugging in upper + lower

$$= \frac{2c}{k}\left(\frac{1}{2}k^2\cdot\ln k - \frac{1}{4}k^2 + \frac{1}{4}\right)$$

$$= ck\ln k - \frac{ck}{2} + \frac{c}{2k}$$

So, what is our goal?

Our goal is:

$$c \cdot k \cdot \ln k - \frac{ck}{2} + \frac{c}{2k} \overset{?}{\leq} c \cdot k \ln k - k + 1$$

these cancel

$$-\frac{ck}{2} + \frac{c}{2k} \overset{?}{\leq} 1 - k$$

Can we get these to cancel?

Let $c = 2$!

$$\frac{2}{2k} - \frac{2k}{2} \overset{?}{\leq} 1 - k$$

$$\frac{1}{k} - k \overset{?}{\leq} 1 - k$$

$$\frac{1}{k} \overset{?}{\leq} 1 \qquad \text{Yes for } k \geq 1 \ \checkmark$$

So QuickSort (avg case) is $T(n) \leq 2n \ln n$