Algorithms & Data Structures
Part 2 (weeks 6–10)
Set 7: Randomised QuickSort

Konrad Dabrowski
konrad.dabrowski@durham.ac.uk

Online Office Hour:
Mondays 13:30–14:30 from Week 7
See Duo for the Zoom link

QuickSort

- If balanced then $\Theta(n \log n)$ (precisely as MergeSort)
- If extremely unbalanced then $\Theta(n^2)$

What about "kinda balanced"?

What would that even mean?

Suppose I promised that every pivot split its problem into 2-to-13 proportion, that is, n spawns subproblems of size $\frac{2}{15}n$ and $\frac{13}{15}n$

Recurrence would be

$$T(n) = T\left(\frac{2}{15}n\right) + T\left(\frac{13}{15}n\right) + cn$$

for some constant $c \geq 1$ (cn is for partitioning)

Questions: how deep does recursion tree go?

Note: $2/15 < 13/15$, thus, starting with n

- there's a **short** branch

$$n \to \frac{2}{15}n \to \left(\frac{2}{15}\right)^2 n \to \left(\frac{2}{15}\right)^3 n \to \cdots \to 1$$

- there's a **long** branch

$$n \to \frac{13}{15}n \to \left(\frac{13}{15}\right)^2 n \to \left(\frac{13}{15}\right)^3 n \to \cdots \to 1$$

- all other branches mix $2/15$ and $13/15$ and therefore are somewhere in-between

Thus, the question appears to be: for what x have we got

$$\left(\frac{13}{15}\right)^x \cdot n \leq 1?$$

Well,

$$\left(\frac{13}{15}\right)^x \cdot n \leq 1 \quad \Leftrightarrow \quad \left(\frac{13}{15}\right)^x \leq \frac{1}{n} \quad \Leftrightarrow \quad \left(\frac{15}{13}\right)^x \geq n$$

Take logarithms on both sides:

$$x \cdot \log_2(15/13) \geq \log_2 n \quad \Leftrightarrow \quad x \geq \frac{\log_2 n}{\log_2(15/13)} = \Theta(\log_2 n)$$

In each level work (partitioning) of $\leq cn$, overall work of $O(n \log n)$ **whenever all splits are proportional with constant ratios**

One idea of using randomisation in algorithms design is to "eliminate" bad input instances

Algorithms often simple, analyses often not so much

Our version of randomised QuickSort (RQS) almost exactly like the one we've seen, with one difference:

- In each call to the partition function, pick pivot uniformly at random from all candidates

Why? Hope is that it then produces reasonably good splits, much of the time

How does one analyse something like this?

Very many ways

Result will be: if X is random variable that denotes number of comparisons during a run of RQS then

$$\mathbb{E}[X] = O(n \log n)$$

Proof

Selection of random pivot spits input of size n into two sub-problems, SMALL and LARGE, of sizes 0 and $n-1$, or 1 and $n-2$, or…or $n-1$ and 0

Each of those is equally likely. With $T(n)$ being expected number of comparisons:

$$
\begin{aligned}
T(n) &= (n-1) + \underbrace{\frac{1}{n} \cdot \left[\sum_{i=0}^{n-1} (T(i) + T(n-i-1))\right]}_{\text{average over splits}} \\
&= (n-1) + \frac{1}{n} \cdot \left[\sum_{i=0}^{n-1} T(i) + \sum_{i=0}^{n-1} T(n-i-1)\right] \\
&= (n-1) + \frac{1}{n} \cdot \left[\sum_{i=0}^{n-1} T(i) + \sum_{i=0}^{n-1} T(i)\right] \\
&= (n-1) + \frac{2}{n} \cdot \sum_{i=0}^{n-1} T(i)
\end{aligned}
$$

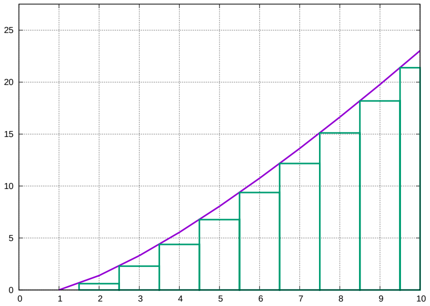
Now use "guess-and-verify" method

What guess? Intuition: most splits fairly close to middle/median, hence $O(n \log n)$, like MergeSort

Will upper-bound sum by integral

If f is increasing function then

$$\sum_{i=1}^{n-1} f(i) \le \int_1^n f(x)dx$$



For us:

$$\int (cx \ln x)dx = (c/2)x^2 \ln x - cx^2/4$$

Guess: $T(i) \le ci \ln i$ for $i \le n-1$

Base case works

Recursive case:

$$\begin{aligned} T(n) &\le (n-1) + \frac{2}{n}\sum_{i=1}^{n-1}(ci \ln i) \\ &\le (n-1) + \frac{2}{n}\int_1^n (cx \ln x)dx \\ &\le (n-1) + \frac{2}{n}\left((c/2)n^2 \ln n - cn^2/4 + c/4\right) \\ &\le cn \ln n \qquad \text{for } c = 2 \end{aligned}$$

Also:

- This is same as basic QS's on randomly shuffled input
  $\Rightarrow$ Basic QS's *average* running time $O(n \log n)$
- Can show *high probability*: if X is actual running time and $T(n)$ expected then $p(X > (1+\delta)T(n)) \le 1/n^\alpha$ for some constants $\delta, \alpha \ge 1$