CS170–Spring 2022 — Homework 1

CurMack

Master Theorem

If $T(n) = aT(\lceil n/b \rceil) + \mathcal{O}(n^d)$ for some constants $a > 0, b > 1$, and $d \geq 0$, then:

$$T(n) = \begin{cases} \mathcal{O}(n^d) & \text{if } d > \log_b a \\ \mathcal{O}(n^d \log n) & \text{if } d = \log_b a \\ \mathcal{O}(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

2. Recurrence Relations

(a) $T(n) = 4T(n/2) + 42n$
Use Mater Theorem: $a = 4, b = 2, d = 1$, since $d < \log_b a$, so:

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$$

(b) $T(n) = 4T(n/3) + n^2$
Use Mater Theorem: $a = 4, b = 3, d = 2$, since $d > \log_b a$, so:

$$T(n) = \Theta(n^d) = \Theta(n^2)$$

(c) $T(n) = T(3n/5) + T(4n/5)$(We have $T(1) = 1$)
Notice that $5^2 = 3^2 + 4^2$, so suppose $T(n) = n^2$, then:

$$T(n) = T(3n/5) + T(4n/5) = (\frac{3}{5}n)^2 + (\frac{4}{5}n)^2 = n^2$$

Therefore the above recurrence is $T(n) = n^2$, which means $T(n) = \Theta(n^2)$

3. Computing Factorials

(a) (Note: You may not use Stirling's formula)
Since: $\log(N!) \leq \log(N^N) = N \log N$, so it's $\mathcal{O}(N \log N)$.
Then: $\log(N!) \geq \log((\frac{N}{2})^{N/2}) = \frac{N}{2} \cdot (\log(N) - 1)$, so it's also $\Omega(N \log N)$.
Therefore, the number has $\Theta(N \log N)$ bits.

(b) we compute $N!$ naively.

Algorithm 1: factorial(N)

$f \leftarrow 1$
for $i = 2 : N$ **do**
  $f \leftarrow f \cdot i$
return f

And the runtime will be $\Theta(N^2 \log^2 N)$.

4. Decimal to Binary

Like Karatsuba's algorithm , we can take an n-digit number x as $10^{n/2} \cdot a + b$ where $a, b, 10^{n/2}$ are 3 $n/2$-digit numbers. So the algorithm will recursively compute the binary representation of a, b and $10^{n/2}$, then the algorithm will take one multiplication and one addition.Since the multiplication takes $\mathcal{O}(n^{\log_2 3})$ by the Karatsuba's algorithm, therefore:

$$T(n) = 3T(n/2) + \mathcal{O}(n^{\log_2 3})$$

According to the Master Theorem, it has solution $T(n) = \mathcal{O}(n^{\log_2 3} \log n)$, which doesn't take much more time than Karatsuba's algorithm.

5. Pareto Optimality

(a) Firstly sort the point array by x-coordinate. Then split the array into 2 subset by x-coordinate. Let L be the left half and R be the right half. And L', R' be the sets of Pareto-optimal points returned. Since every points in R' is Pareto-optimal, and for each point in L', if its y-coordinate is lager then the largest y-coordinate y_{max} in R', it is Pareto optimal. So remove the points with lower y-coordinate in L' and return the union of L' and R'.
Since the scan takes linear time, so:

$$T(n) = 2T(n/2) + \mathcal{O}(n)$$

According to the Master theorem , the runtime is $\mathcal{O}(n \log n)$.

(b) Pseudocode:

Algorithm 2: Pareto Optimality(P)

Input: P is a piont array preprocessed by sorting in x-coordinate.

$P' \leftarrow \phi$

split P into L and R by x-coordinate.

$L' \leftarrow \mathbf{PO}(L), R' \leftarrow \mathbf{PO}(R)$

$P' \leftarrow P' \cup R'$

$y_{max} \leftarrow$ the maximum y-value in R'

for $(x_i, y_i) \in L'$ **do**

    if $y_i > y_{max}$ **then**

        $P' \leftarrow P' \cup (x_i, y_i)$

return P'