CS 344: The Master Theorem

We would like to solve recurrence relation of the type that often arise in Divide-and-Conquer algorithms.

$$T(n) = aT(\lceil \frac{n}{b} \rceil) + O(n^d),$$

where $a > 0$, $b > 1$, $d > 0$. We assume $T(1) = c$, a constant.

We can think of a as the number of subproblems, n/b as the size of each subproblem. The quantity $O(n^d)$ can be viewed as the cost of dividing and combining subproblems.

Theorem (Master Theorem).

$$T(n) = O(n^d), \quad if \quad d > \log_b a,$$

$$T(n) = O(n^d \log n), \quad if \quad d = \log_b a,$$

$$T(n) = O(n^{\log_b a}), \quad if \quad d < \log_b a.$$

Proof. For simplicity we will assume $n = b^k$ and $O(n^d) = n^d$. We may write

$$T(n) = T(b^k) = aT(b^{k-1}) + b^{kd} =$$

$$a(aT(b^{k-2}) + b^{d(k-1)}) + b^{kd} = a^2T(b^{k-2}) + ab^{d(k-1)} + b^{kd} =$$

$$a^3T(b^{k-3}) + a^2b^{d(k-2)} + ab^{d(k-1)} + b^{kd} = \cdots$$

$$a^kT(1) + \sum_{j=0}^{k-1} a^j b^{d(k-j)}.$$

Let

$$r = b^d, \quad \rho = \frac{a}{r}.$$

Note that $a^k = a^{\log_b n} = n^{\log_b a}$ and $r^k = b^{dk} = (b^k)^d = n^d$. Thus we have

$$T(n) = cn^{\log_b a} + n^d \sum_{j=0}^{k-1} \rho^j.$$

It is now a matter analyzing $S = n^d \sum_{j=0}^{k-1} \rho^j$ for different values of ρ.

$$If \quad d > \log_b a, \quad then \quad r > a. \quad Thus \quad \rho < 1 \quad and \quad S = n^d \frac{\rho^k - 1}{\rho - 1} = O(n^d).$$

$$If \quad d = \log_b a, \quad then \quad r = a. \quad Thus \quad \rho = 1 \quad and \quad since \quad \sum_{j=0}^{k-1} \rho^j = k, \quad S = O(n^d \log n).$$

$$If \quad d < \log_b a, \quad then \quad r < a. \quad Thus \quad \rho > 1 \quad and \quad S = n^d \frac{\rho^k - 1}{\rho - 1} = O(r^k \rho^k) = O(a^k) = O(n^{\log_b a}).$$