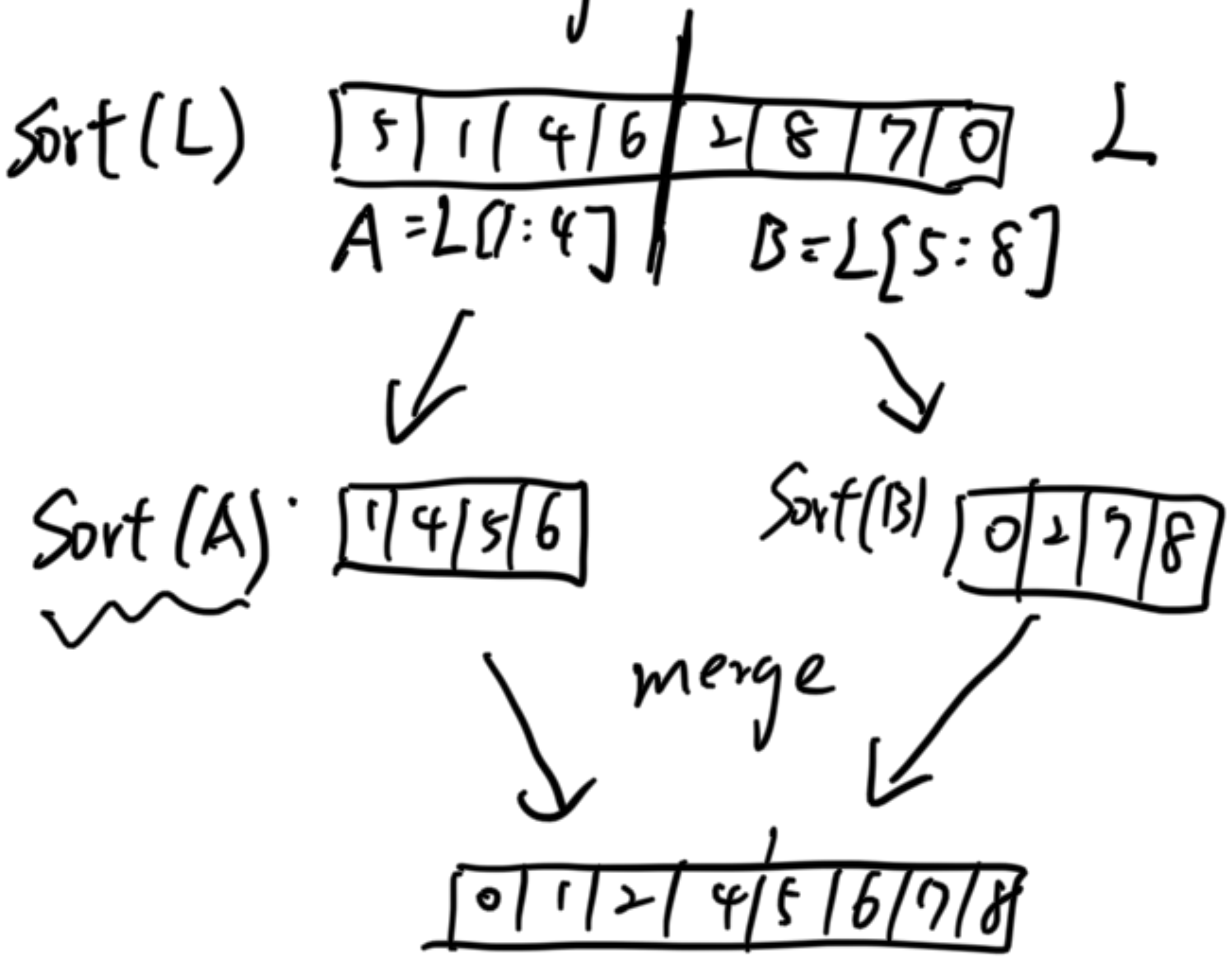
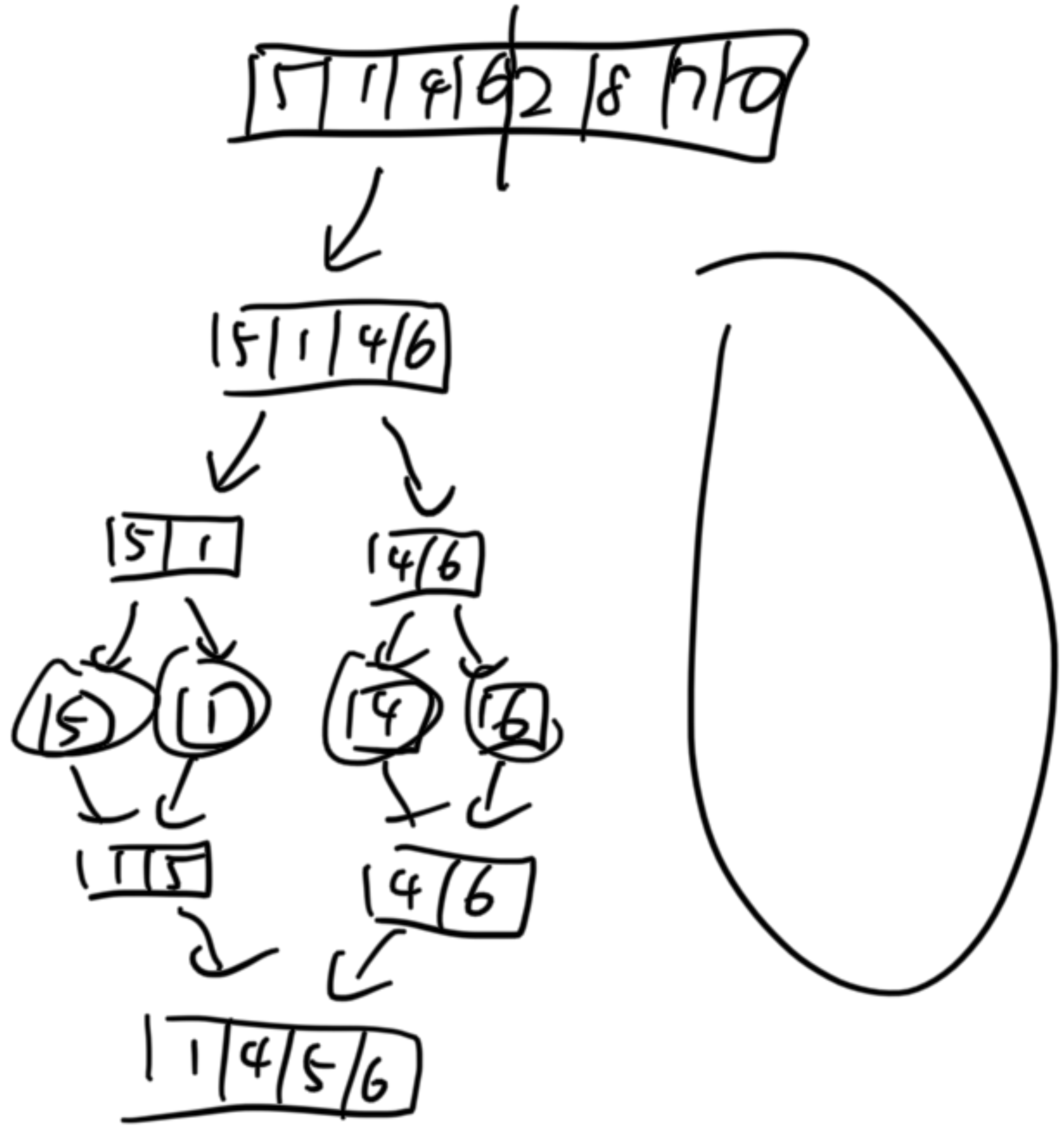
5.1 Divide-and-Conquer.

Merge Sort.





MergeSort(L)

$T(n)$

- If L has ≤ 1 element
  - return L
- Else:
  - $O(1) \leftarrow$ $A = L[1 : n/2]$, $B = L[n/2+1 : n]$
  - $T(n/2) \leftarrow$ $A \leftarrow$ MergeSort(A)
  - $T(n/2) \leftarrow$ $B \leftarrow$ MergeSort(B)
  - $O(n) \leftarrow$ $L \leftarrow$ Merge(A, B)
  - return L

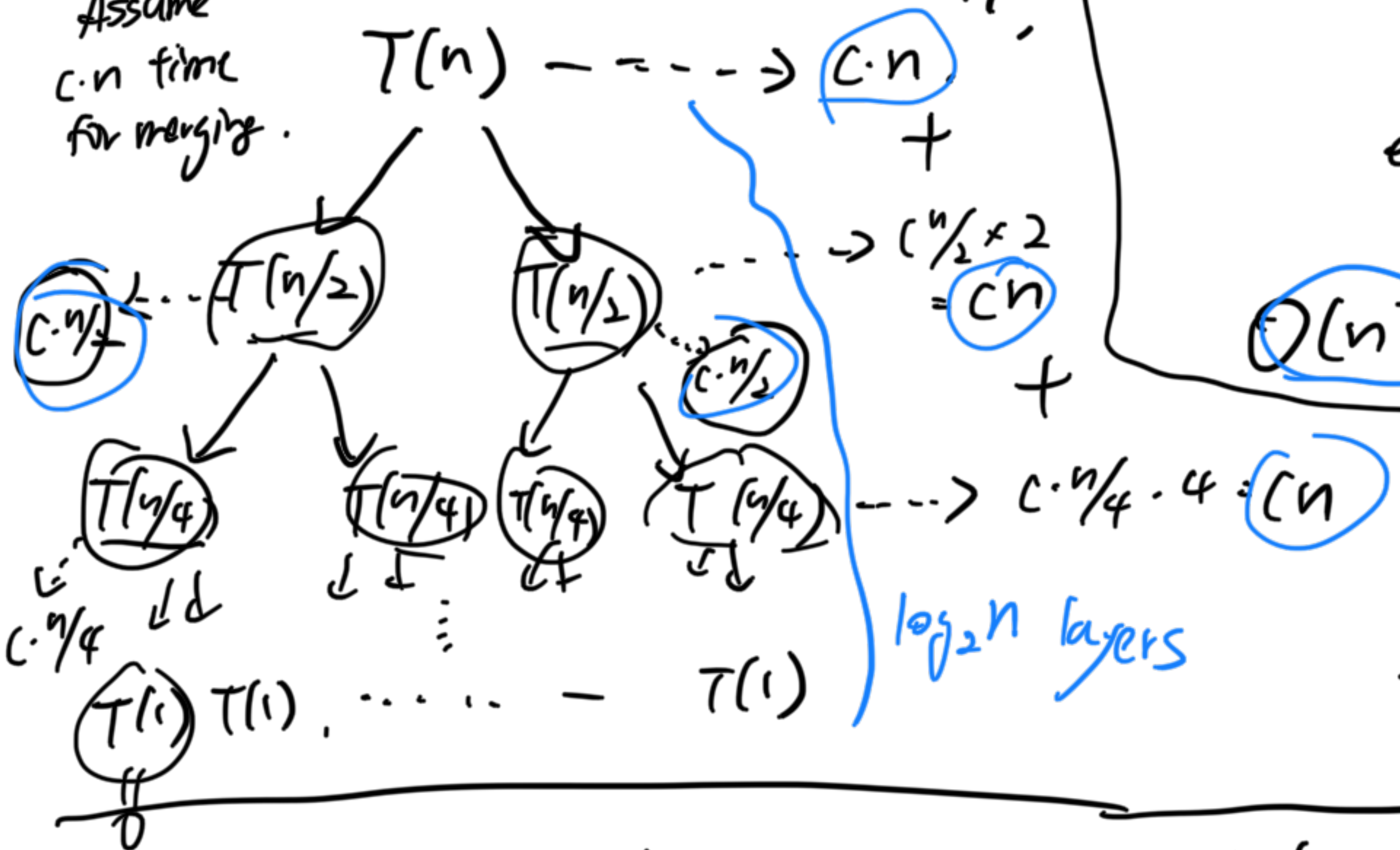
Merge two sorted array:

Initial $i = 1$, $j = 1$

return L.

$T(n)$: time for solving problem of size n.

Assume $c \cdot n$ time for merging.



while ($i \le n$ and $j \le n$)
    if $A[i] \le B[j]$
        add $A[i]$ to L
        i++.
    else
        add $B[j]$ to L
        j++.

$O(n)$ time for merging.

total time: $\underbrace{\log_2 n}_{\text{layers}} \times \underbrace{cn}_{\text{total cost for each layer}} = c n \log_2 n = O(n \log n)$.

Induction to prove $T(n) = O(n \log_2 n)$.

$T(n) \le 2 \cdot T(n/2) + c \cdot n$, $T(1) = 0$. ↗ const.

by induction: Base case $n = 2$ $\quad T(2) = 2 \cdot T(1) + c \cdot 2 = 2c = 2c \cdot \log_2 2 = O(n \log n)$.

Induction Hypothesis: Assume $T(m) \le cm \cdot \log_2 m \quad \forall m < n$

Induction Step: proof this for n

$$T(n) \le 2 \cdot T(n/2) + cn \le \cancel{2}\left(c \cdot \frac{n}{\cancel{2}} \log_2 \frac{n}{2}\right) + cn$$

$$= cn \underbrace{\log_2 \frac{n}{2}}_{\log_2 n - 1} + cn = cn\log_2 n - cn + cn$$

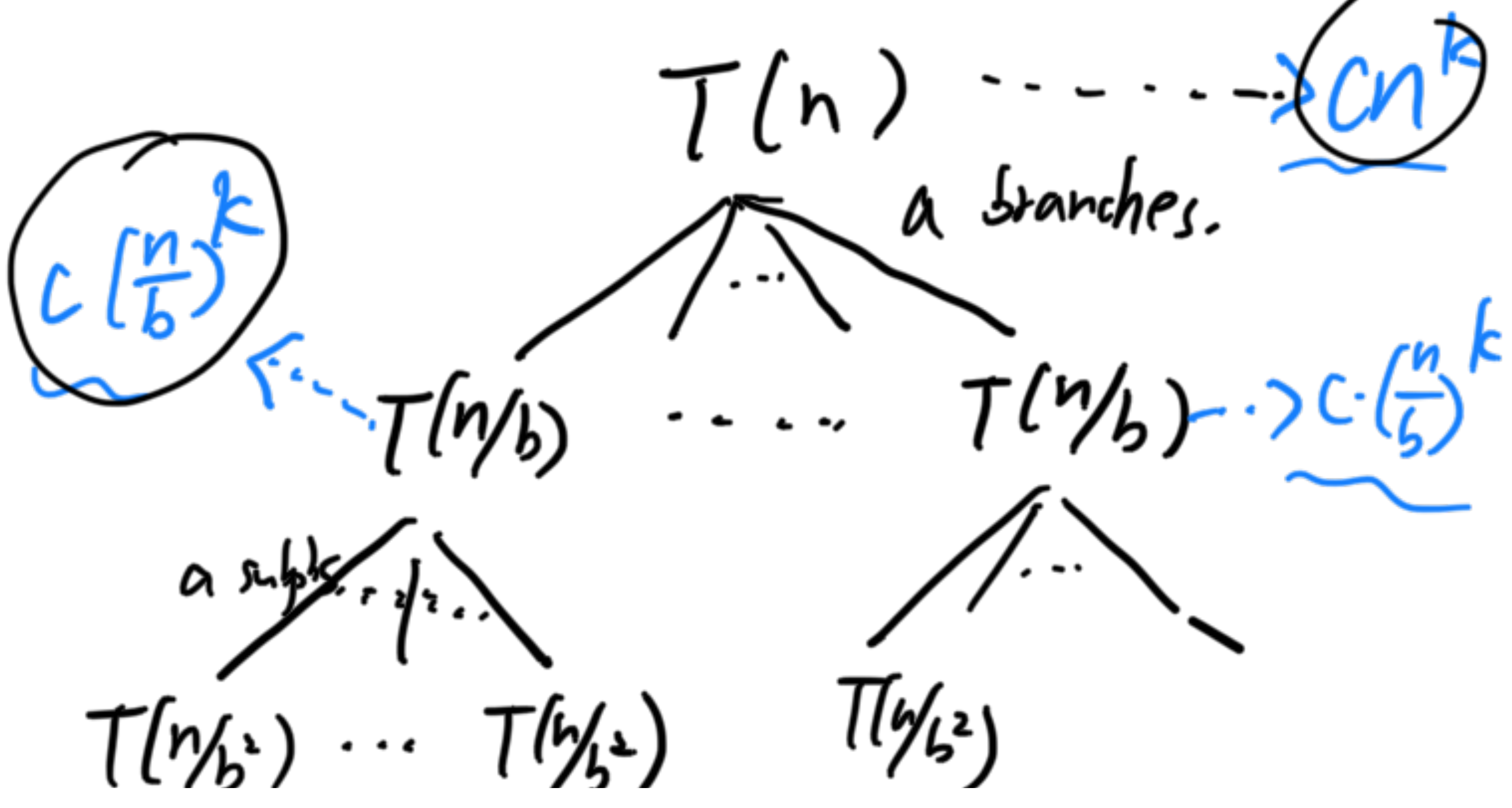
$$= cn \log_2 n \quad \checkmark$$

General Case:

\# subpb: a subpb size: n/b cost for merging cn^k.

$$T(n) \le a \cdot T(n/b) + cn^k \qquad r = \frac{a}{b^k}$$

e.g mergesort $a=2$ $b=2$, $k=1$.



layer	#subpbs	size	cost
1	1	n	$cn^k = cn^k \cdot r^0$
2	a	n/b	$a \cdot c \cdot \left(\frac{n}{b}\right)^k = cn^k \cdot r^1$ $= cn^k \cdot \left(\frac{a}{b^k}\right)$
3	a^2	n/b^2	$a^2 \cdot c \cdot \left(\frac{n}{b^2}\right)^k$

$= cn^k \cdot \left(\frac{a}{b^k}\right)$

$= cn^k \cdot r^2$

$\vdots$

ℓ $\quad cn^k \cdot r^{\ell-1}$ $+$

$\approx \log_b n$ layers.

$$\text{total cost} = \underline{cn^k(1 + r + r^2 + \cdots + r^{\log_b n - 1})}$$

Case I: $r = 1$, $a/b^k = 1$ ← mergesort.

$$\text{total cost} = cn^k(\underbrace{1+1+1+\cdots+1}_{\log_b n}) = c \cdot \log_b n \cdot n^k = O(n^k \log n).$$

Case $\underline{\text{II}}$: $r > 1$, $a/b^k > 1$

$$\text{total cost} = cn^k(1 + r + r^2 + \cdots + r^{\log_b n - 1})$$

$$= cn^k\left(\frac{r^{\log_b n} - 1}{r - 1}\right) \le cn^k \frac{r^{\log_b n}}{r - 1}$$

$$= O(n^k \cdot r^{\log_b n})$$

$$= O\left(\cancel{n^k} \frac{n^{\log_b a}}{\cancel{n^k}}\right)$$

$r^{\log_b n} = \left(\frac{a}{b^k}\right)^{\log_b n}$

$= \frac{a^{\log_b n}}{(b^k)^{\log_b n}}$ $a^{\log_b n} \to n^{\log_b a}$

$$= \underbrace{O(n^{\log_b a})}.$$

$$(b^{\log_b n})^k = n^k$$

Case III: $r < 1$, $a/b^k < 1$

total cost $= cn^k(1 + r + r^2 + \dots + r^{\log_b n - 1})$

$$= cn^k\left(\frac{1 - r^{\log_b n}}{1 - r}\right) \le cn^k \frac{1}{1-r} = O(n^k).$$

Master Theorem: If $T(n) \le \underline{a} \cdot T(n/b) + cn^k$ then

$$T(n) = \begin{cases} O(n^k \log n) & \text{if } a/b^k = 1 \\ O(n^{\log_b a}) & \text{if } a/b^k > 1 \\ O(n^k) & \text{if } a/b^k < 1. \end{cases}$$