6 Mergesort and Solving Recursions [KT 5.1]

The Algorithm: Divide the problem to two subproblems, sort separately, and then merge.

Merge-Sort($A[]$)

IF (A.length $\leq$ 1) THEN RETURN $A[]$ // Boundary case

$X[]$ = Merge-Sort($A[1, A.len/2]$)

$Y[]$ = Merge-Sort($A[A.len/2+1, n]$)

RETURN Merge($X[], Y[]$)

Now let us see how to design the Merge($n/2, X[], Y[]$) process. Here $X[]$ and $Y[]$ are two sorted lists each with $n/2$ number, and the goal is to return a sorted list (namely $Z[]$) of all n numbers. We decide the numbers in Z one by one. Note that the first number in $Z[]$ should be the smallest among $X[]$ and $Y[]$, therefore should be the smallest between $X[1]$ and $Y[1]$. Here we used the order of $X[]$ and $Y[]$ so that we do not have to scan the whole lists. We pick the smaller one in $X[1]$ and $Y[1]$ to be $Z[1]$, and then decide $Z[2], Z[3], \ldots$ accordingly. We describe it in pseudo-code as follows.

Merge($X[], Y[]$)

$i = j = k = 1$

WHILE ($i \leq X.len$ or $j \leq Y.len$)

IF (($j > Y.len$) or ($i \leq X.len$ and $X[i] \leq Y[j]$))

THEN $Z[k++] = X[i++]$

ELSE $Z[k++] = Y[j++]$

RETURN $Z[]$

We see that the runtime of Merge is $\Theta(n)$. Let $T(n)$ be the runtime of Merge-Sort on n numbers, we have the following recurrence:

$$T(n) \leq 2T(n/2) + cn \text{ when } n \geq 2,$$

where c is some constant, and $T(1) \leq c$.

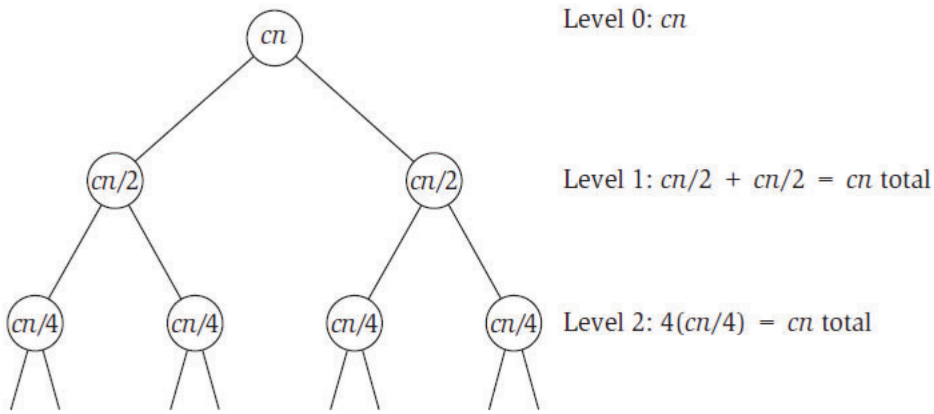

Figure 5.1 Unrolling the recurrence $T(n) \le 2T(n/2) + O(n)$.

Two ways of solving recursion

- Unrolling. See Figure 5.1. After $\log_2 n$ steps, the nodes at level $\log_2 n$ (the root is at level 0) correspond to subproblems of size 1, and thus have running time at most c. The running time of the original problem can thus be bounded by $cn \times \log_2 n = O(n \log n)$.

- Guess + Verify. Guess $T(n) \le cn \log_2 n$ for all $n \le k$, and then use induction to prove that $T(n) \le cn \log_2 n$ is also true for $n = k + 1$.

Homework 5 *Try to solve the following recursions using "Unrolling" and "Guess + Verify"*

1. $T(n) = 2T(\frac{n}{2}) + c$, *with boundary condition* $T(n) = 1$ *for all* $n \le 1$.

2. $T(n) = 2T(\frac{n}{2}) + c\sqrt{n}$, *with boundary condition* $T(n) = 1$ *for all* $n \le 1$.

3. $T(n) = 2T(\frac{n}{2}) + cn^2$, *with boundary condition* $T(n) = 1$ *for all* $n \le 1$.