② $\lim_{n\to\infty} \sum_{i=1}^{d-1} a_i n^{i-k} = 0$, find c by the definition of the limit.

$\underline{\underline{c}}$ n_0 ① $c = \sum |a_i|$

Hw9-Hw11

- Q3: Prove $\sum_{i=0}^{d} a_i n^i = O(n^k)$ if $k \geq d$ by ~~definition~~.
- Q4: Show that the majority element problem can be reduced to the sorting problem, following the three steps of reduction.

step 1:

step 2: sorting algorithm

step 3: output of sorting
$\Theta(n)$
$\downarrow$
output of majority element.

① find the median from the sorted array $O(1)$, count # of appearances of it to see whether it is the majority element. $O(n)$

② since identical elements are grouped together in the sorted array, scan the sorted array and count # of appearances for each element to see whether there is a majority element; $O(n)$.

9.3-1 In the algorithm **SELECT**, the input elements are divided into groups of **5**. Will the algorithm work in linear time if they are divided into groups of **7**? Argue that **SELECT** does not run in linear time if groups of **3** are used.

$$T(n) = T\left(\frac{n}{7}\right) + T\left(\frac{10}{14}n\right) + \Theta(n)$$

Suppose that $T(k) \leq kn$ and $\Theta(n) = an$

$$T(n) \leq \frac{c}{7}n + \frac{5}{7}cn + an$$
$$\leq \left(\frac{6}{7}c + a\right)n$$

Inequality holds when $c \geq 7a$.

$T(n) = O(n)$.

9.3-1 In the algorithm **SELECT**, the input elements are divided into groups of **5**. Will the algorithm work in linear time if they are divided into groups of **7**? Argue that **SELECT** does not run in linear time if groups of **3** are used.

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{4}{6}n\right) + \Theta(n)$$

an an

an $\frac{n}{3}an$ $\frac{2}{3}an$

Wrong: Supposing $T(k) \le ck$ and $\Theta(n) = an$ to get

$$T(n) \le cn + an > cn$$

We CANNOT reach the conclusion that T(n) is not O(n)!

Two possible solutions:

1. Suppose $T(n) \ge cn \lg n$. $\qquad T(n) = \Omega(n \lg n)$
2. For any $c > 0$,

   Prove $T(n) > cn$. $\qquad T(n) = \omega(n)$

9.3-7 Describe an $O(n)$ time algorithm that, given a set S of n distinct numbers and a positive integer $k \leq n$, determines the k numbers in S that are closest to the median of S.

1. Find the median using SELECT. $O(n)$

2. Compute all $|A[i] - median|$. $O(n)$

3. Find the k th smallest absolute difference X. $O(n)$

4. Scan through the absolute differences and pick out those smaller than X. Restore the original elements from the differences. $O(n)$

Numbers between the $\left(\frac{n-k}{2}\right)^{th}$ and the $\left(\frac{n+k}{2}\right)^{th}$ smallest elements?

1 2 3 4 4.1 4.2 5

4-1 Solve recurrences by the master method.

b) $T(n) = T\left(\frac{7}{10}n\right) + n$

d) $T(n) = 7T\left(\frac{n}{3}\right) + n^2$

Case 3 of the master method. Don't forget to verify the **regularity condition**!

$af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$.

4-2

1. An array is passed by a pointer. Time = $\Theta(1)$. ✓

2. An array is passed by a copying. Time = $\Theta(N)$, where N is the size of the array.

3. An array is passed by copying only the subrange that might be accessed by the called procedure. Time = $\Theta(q-p+1)$ if the subarray $A[p\,..\,q]$ is passed. ✓

Consider the MERGE-SORT algorithm. Give recurrences for the worst-case running times when arrays are passed using each of the three methods above, and give good upper bounds on the solutions of the recurrences.

Let N be the size of the original problem and n be the size of a subproblem.

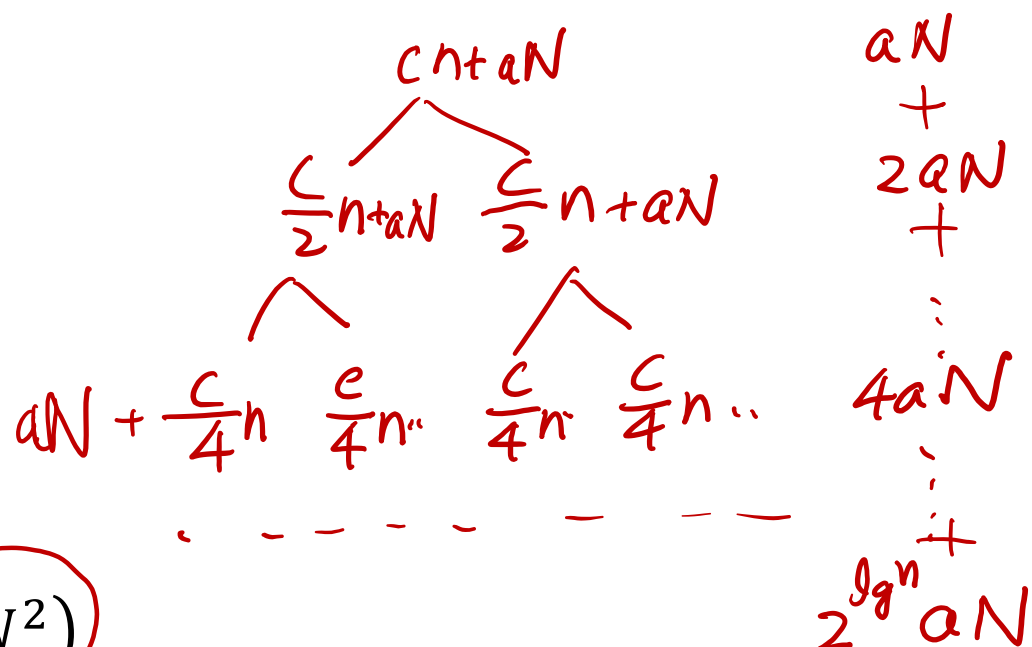
$T(n) = 2T(n/2) + \Theta(n) + \Theta(N)$

2. An array is passed by a copying. Time = $\Theta(N)$, where N is the size of the array.

$$T(n) = 2T\left(\frac{n}{2}\right) + \underbrace{\Theta(n)}_{cn} + \underbrace{\Theta(N)}_{aN}$$

N is constant in terms of n.

Add extra running time to each layer of the recursion tree.



$$a \cdot N\left(1 + 2 + \cdots + 2^{\lg n}\right)$$

$$T(N) = \Theta(N \lg N) + \Theta(N^2)$$
$$= \Theta(N^2)$$