1-1

GETMIN(A,*p*)	cost	times
min ← A[*p*]	c1	1
for *j* ← *p*+1 to *n*	c2	1
if A[*p*] < min	c3	n-p-1
min ← A[*j*]	c4	n-p-1
k ← *j*	c5	n-p-1
F ← 1	c6	n-p-1
if F = 1	c7	1
exchange A[*p*] ↔ A[*k*]	c8	1

6

T(*p*) = c1+c2+c7+c8 +(c3+c4+c5+c6)(n-p-1)
= c1+c2+c7+c8 -(c3+c4+c5+c6)(p+1)
+ (c3+c4+c5+c6) n
= c9+c10 n
T(*p*) = $O(n)=\Theta(n)$

3

GETMIN(A,p)	cost	times
for *j* ← *p*+1 to *n*	c1	1
if A[*p*] < A[*j*]	c2	n-p-1
exchange A[*p*] ↔ A[*j*]	c3	<n-p-1

4

T(*p*) = c1+(c2+c3)(n-p-1)
= c1-(c2+c3)(p+1)+(c2+c3) n
= c4 +c5 n
T(*p*) = $O(n)$

2

1-2
$T(n)=T(n-1)+\Theta(n)$ 2

T(n-1) ⇒ solving the problem of n-1 elements 2

$\Theta(n)$ ⇒ running time of GETMIN 2

$T(n)= O(n^2) = \Theta(n^2)$ ⇒ using substitution method or recursion tree method 3

1-3
No ⇒ b=1 3

1-4
Using min-priority queue
Initially: Build-Min-Heapify should be performed. 9
Use Extract-Min to get the min for each A[*p*]

2-1
Merge-sort ⇒ $O(n \lg n)$
Extraction ⇒ $O(1)$ 3
$T(n)=O(n \lg n + m) = O(n \lg n)$

2-2
Build a min-heap ⇒ $O(n)$
Extract-Min ⇒ $O(\lg n)$ 3
$T(n)=O(n + m \lg n)$

2-3

YES.

1- for i← 1 to m

2- . GETMIN(i)

2

GETMIN $\Rightarrow O(n)$

T(n)= $O(mn)$

2

3-1

$$\frac{n^2}{2} - 3n \stackrel{?}{=} \Theta(n^2)$$

$$c_1 n^2 \le \frac{n^2}{2} - 3n \le c_2 n^2$$

5

$$c_1 \le \frac{1}{2} - \frac{3}{n} \le c_2$$

$$c_1 = 1/14 \qquad c_2 = 1/2 \qquad n_\circ = 7$$

$$\frac{n^2}{2} - 3n = \Theta(n^2)$$

3-2

$$(\sqrt{2})^{\lg n} \stackrel{?}{=} \Theta(n)$$

$$(\sqrt{2})^{\lg n} = 2^{\frac{1}{2}\lg n} = n^{\frac{1}{2}} = \sqrt{n}$$

$$c_1 n \le \sqrt{n} \le c_2 n$$

5

$$c_1 \le \frac{1}{\sqrt{n}} \le c_2$$

$$\frac{1}{\sqrt{\infty}} = 0 \qquad c_1 = !!!$$

$$(\sqrt{2})^{\lg n} \ne \Theta(n)$$

3-3

$$f(n) + g(n) \stackrel{?}{=} \Theta(\max(f(n), g(n)))$$

$$c_1(\max(f(n), g(n))) \le f(n) + g(n) \le c_2(\max(f(n), g(n)))$$

$$(\max(f(n), g(n))) \le f(n) + g(n) \Rightarrow c_1 = 1$$

5

$$f(n) + g(n) \le 2(\max(f(n), g(n))) \Rightarrow c_2 = 2$$

$$n_0 = 1$$

$$f(n) + g(n) = \Theta(\max(f(n), g(n)))$$

4-1

$T(n)=4T(n/2)+n^3$

a=4 b=2 f(n)=n^3

$n^{\log_b a} = n^{\log_2 4} = n^2$

3^{rd} case of master method ε=1 **5**

2^{nd} condition of 3^{rd} case: $af(n/b) < cf(n)$

$4(n/2)^3 = 1/2\ n^3$ c=1/2<1

$T(n)=\Theta(n^3)$

4-2

$T(n) = 3T(n^{\frac{1}{3}}) + \log_3 n$

$n = 3^m$

$T(3^m) = 3T(3^{\frac{m}{3}}) + m$

$S(m) = 3S(m/3) + m$ **5**

$a = 3 \quad b = 3 \quad f(m) = m$

$m^{\log_b a} = m^{\log_3 3} = m = f(m)$

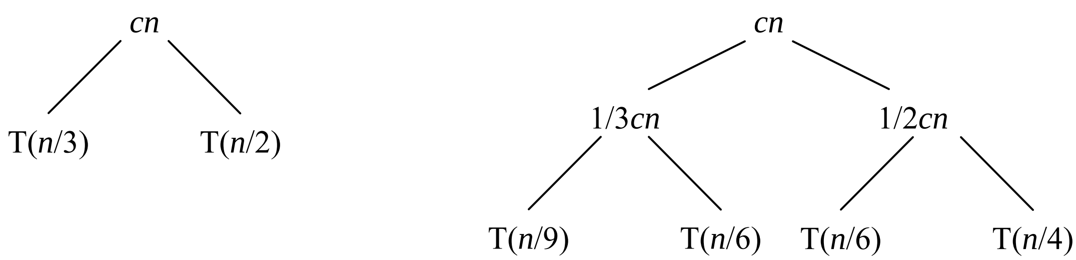
2^{nd} case of master method

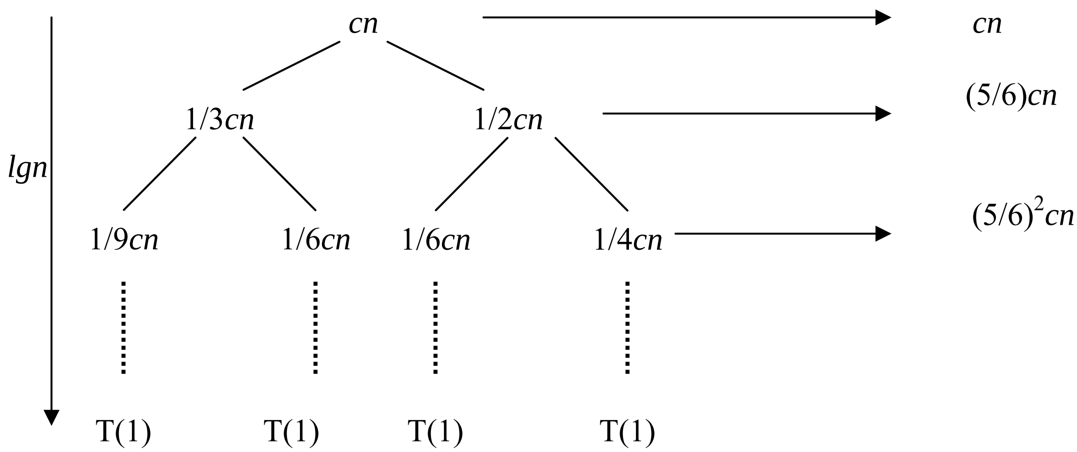
$S(m)=\Theta(m\lg m)$

$T(n)=T(3^m)=S(m)= \Theta(m\lg m)= \Theta(\log_3 n \lg \log_3 n)= \Theta(\lg n \lg \lg n)$

4-3

T(n)=T(n/2)+T(n/3)+n





$$T(n) = \sum_{i=0}^{\lg n} \left(\frac{5}{6}\right)^i cn$$

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \qquad \Rightarrow \sum_{i=0}^{\infty} \left(\frac{5}{6}\right)^i = \frac{1}{1-\frac{5}{6}} = 6$$

$$T(n) \le 6cn$$

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

Prove that this guess is correct

$T(n) \le T(n/2)+T(n/3)+c\,n$
$= d/2\,n + d/3\,n + c\,n$
$= 5/6\,d\,n + c\,n$
$T(n) \le d\,n$ $\quad c \le 1/6\,d$ **5**

5-1
B-Sort takes $O(k\sqrt{k})$ times for each k-element list. **6**

Sorting n/k of k-element lists $\Rightarrow$ $O(\frac{n}{k}\cdot k\sqrt{k}) = O(n\sqrt{k})$

5-2
Recursion tree $\Rightarrow$ last level problem size = n/k $\Rightarrow$ tree height lg (n/k) **6**
Running time = $O(n \lg(n/k))$

5-3
T(n) = $O(n\sqrt{k} + n\lg n - n\lg k)$
$\sqrt{k} = \lg n \Rightarrow k = O(\lg^2 n)$ **9**

5-4
T(n) = $cn\sqrt{k} + dn\lg n - dn\lg k$ **9**
$\frac{\partial T(n)}{\partial k} = \frac{cn}{2\sqrt{k}} - \frac{dn}{k} = 0$
$k = 2d/c$