2-11 Heapsort

Hengfeng Wei

hfwei@nju.edu.cn

June 11, 2018

ALGORITHM 245
TREESORT 3 [M1]
ROBERT W. FLOYD (Recd. 22 June 1964 and 17 Aug. 1964)
Computer Associates, Inc., Wakefield, Mass.

procedure *TREESORT* 3 (M, n);
value n; **array** M; **integer** n;
comment *TREESORT* 3 is a major revision of *TREESORT* [R. W. Floyd, Alg. 113, *Comm. ACM 5* (Aug. 1962), 434] suggested by *HEAPSORT* [J. W. J. Williams, Alg. 232, *Comm. ACM 7* (June 1964), 347] from which it differs in being an in-place sort. It is shorter and probably faster, requiring fewer comparisons and only one division. It sorts the array $M[1:n]$, requiring no more than $2 \times (2 \uparrow p-2) \times (p-1)$, or approximately $2 \times n \times (\log_2(n)-1)$ comparisons and half as many exchanges in the worst case to sort $n = 2 \uparrow p - 1$ items. The algorithm is most easily followed if M is thought of as a tree, with $M[j \div 2]$ the father of $M[j]$ for $1 < j \leq n$;

Worst-case of MAX-HEAPIFY (TC $6.2 - 6$)

Show that the worst-case running time of MAX-HEAPIFY on an n-element heap is $\Omega(\log n)$.

Worst-case of MAX-HEAPIFY (TC $6.2-6$)

Show that the worst-case running time of MAX-HEAPIFY on an n-element heap is $\Omega(\log n)$.

EXCHANGE *vs.* COMPARE

反馈: O, Θ, Ω 傻傻分不清。

什么时候用哪个?

这道题为什么问的是 Ω, 而不问 O 或 Θ?

Algorithm $\mathcal{A}$

Inputs $\mathcal{I}$ of size n

	O	Ω	Θ
Best-case			
Worst-case			
Average-case			

Algorithm $\mathcal{A}$

Inputs $\mathcal{I}$ of size n

	O	Ω	Θ
Best-case			$O = \Omega$
Worst-case			$O = \Omega$
Average-case			$O = \Omega$

Algorithm $\mathcal{A}$

Inputs $\mathcal{I}$ of size n

	O	Ω	Θ
Best-case	by example	"weakness" of $\mathcal{A}$	$O = \Omega$
Worst-case			$O = \Omega$
Average-case			$O = \Omega$

Algorithm $\mathcal{A}$

Inputs $\mathcal{I}$ of size n

	O	Ω	Θ
Best-case	by example	"weakness" of $\mathcal{A}$	$O = \Omega$
Worst-case	"power" of $\mathcal{A}$	by example	$O = \Omega$
Average-case			$O = \Omega$

Algorithm $\mathcal{A}$

Inputs $\mathcal{I}$ of size n

	O	Ω	Θ
Best-case	by example	"weakness" of $\mathcal{A}$	$O = \Omega$
Worst-case	"power" of $\mathcal{A}$	by example	$O = \Omega$
Average-case	$\leq$	$\geq$	$O = \Omega$

Worst-case of MAX-HEAPIFY (TC $6.2-6$)

Show that the worst-case running time of MAX-HEAPIFY on an n-element heap is $\Omega(\log n)$.

Worst-case of MAX-HEAPIFY (TC $6.2-6$)

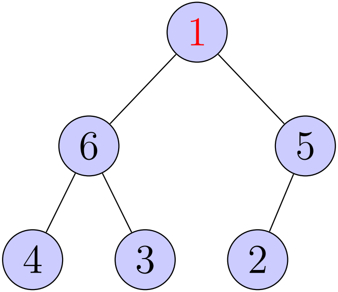
Show that the worst-case running time of MAX-HEAPIFY on an n-element heap is $\Omega(\log n)$.

By Example.

Worst-case of MAX-HEAPIFY (TC $6.2 - 6$)

Show that the worst-case running time of MAX-HEAPIFY on an n-element heap is $\Omega(\log n)$.

By Example.

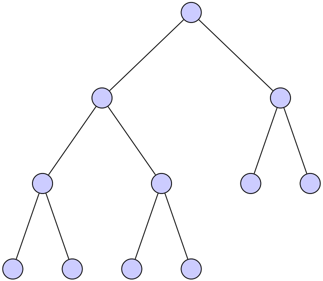


Worst-case of MAX-HEAPIFY (TC $6.2 - 6$)

Show that the worst-case running time of MAX-HEAPIFY on an n-element heap is $O(\log n)$.

Worst-case of Max-Heapify (TC $6.2 - 6$)

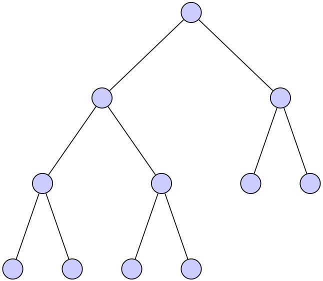
Show that the worst-case running time of Max-Heapify on an n-element heap is $O(\log n)$.



$$W(n) \leq H(n)$$

Worst-case of MAX-HEAPIFY (TC $6.2 - 6$)

Show that the worst-case running time of MAX-HEAPIFY on an n-element heap is $O(\log n)$.



$$W(n) \leq H(n)$$

No Examples Here!

Therefore . . .

Worst-case of MAX-HEAPIFY (TC $6.2-6$)

Show that the worst-case running time of MAX-HEAPIFY on an n-element heap is $\Theta(\log n)$.

	O	Ω	Θ
Worst-case	"power" of $\mathcal{A}$	by example	$O = \Omega$

Worst-case of HEAPSORT (TC $6.4 - 4$)

Show that the worst-case running time of HEAPSORT is $\Omega(n \log n)$.

Worst-case of HEAPSORT (TC $6.4 - 4$)

Show that the worst-case running time of HEAPSORT is $\Omega(n \log n)$.

By Example.

Worst-case of HEAPSORT (TC $6.4-4$)

Show that the worst-case running time of HEAPSORT is $\Omega(n \log n)$.

By Example.

Non-proof.

$$\underbrace{\Theta(n)}_{\text{EXTRACT-MAX}} \times \underbrace{\Omega(\log n)}_{\text{MAX-HEAPIFY}} = \Omega(n \log n)$$

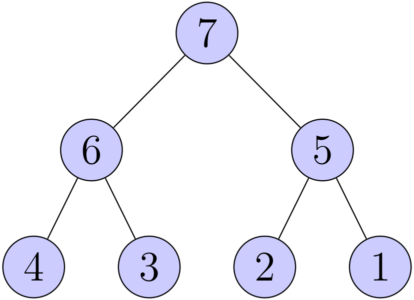
□

Worst-case of HEAPSORT (TC $6.4-4$)

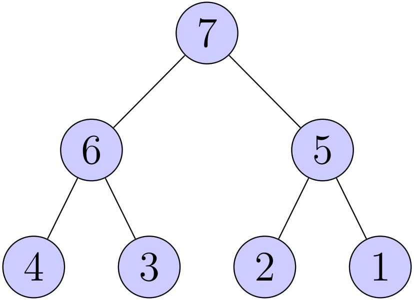
Show that the worst-case running time of HEAPSORT is $\Omega(n \log n)$.

Heap in decreasing order?

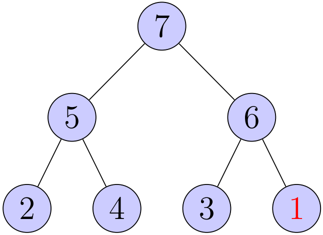
Heap in decreasing order?

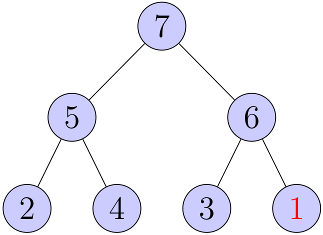


Heap in decreasing order?

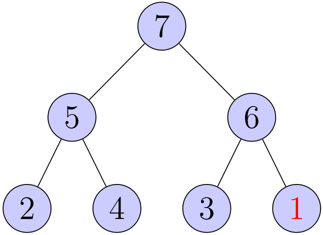


$$T(7) = 2 + 1 + 1 + 1 + 0 + 0 = 5$$


7
5
6
2
4
3
1

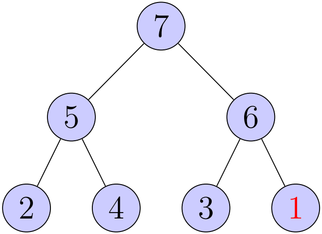


$T(7) = 2 + 2 + 2 + 1 + 1 + 0 = 8$



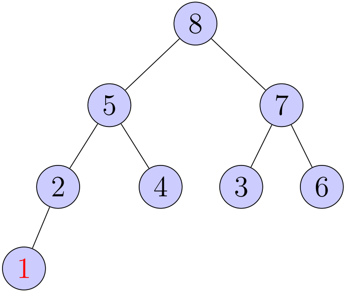
$T(7) = 2 + 2 + 2 + 1 + 1 + 0 = 8$

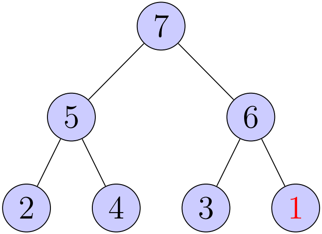
(Ex. 23, Section 5.2.3, TAOCP Vol 3)



$$T(7) = 2 + 2 + 2 + 1 + 1 + 0 = 8$$

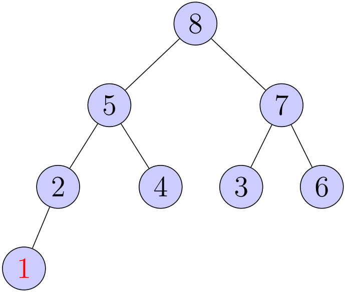
(Ex. 23, Section 5.2.3, TAOCP Vol 3)



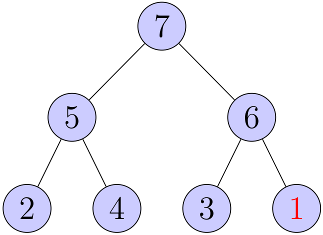


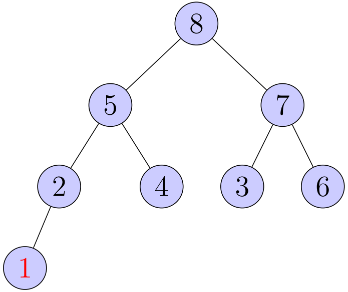
$$T(7) = 2 + 2 + 2 + 1 + 1 + 0 = 8$$

(Ex. 23, Section 5.2.3, TAOCP Vol 3)



$$\sum_{r=1}^{n-1} \lfloor \log r \rfloor$$

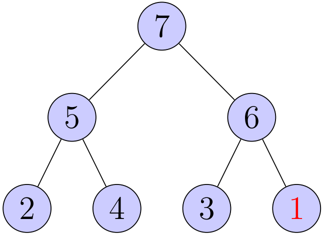


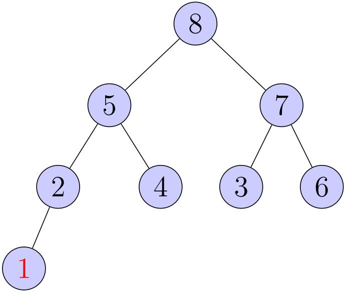


$$T(7) = 2 + 2 + 2 + 1 + 1 + 0 = 8$$

(Ex. 23, Section 5.2.3, TAOCP Vol 3)

$$\sum_{r=1}^{n-1} \lfloor \log r \rfloor = n \lfloor \log n \rfloor - 2^{\lfloor \log n \rfloor + 1} + 2$$





$$T(7) = 2 + 2 + 2 + 1 + 1 + 0 = 8$$

(Ex. 23, Section 5.2.3, TAOCP Vol 3)

$$\sum_{r=1}^{n-1} \lfloor \log r \rfloor = n \lfloor \log n \rfloor - 2^{\lfloor \log n \rfloor + 1} + 2 = \Omega(n \log n)$$

Worst-case of HEAPSORT (TC $6.4 - 4$)

Show that the worst-case running time of HEAPSORT is $O(n \log n)$.

Worst-case of $\textsc{Heapsort}$ (TC $6.4 - 4$)

Show that the worst-case running time of $\textsc{Heapsort}$ is $O(n \log n)$.

$$\sum_{r=1}^{n-1} \lfloor \log r \rfloor = n \lfloor \log n \rfloor - 2^{\lfloor \log n \rfloor + 1} + 2 = O(n \log n)$$

Worst-case of HEAPSORT (TC $6.4 - 4$)

Show that the worst-case running time of HEAPSORT is $O(n \log n)$.

$$\sum_{r=1}^{n-1} \lfloor \log r \rfloor = n \lfloor \log n \rfloor - 2^{\lfloor \log n \rfloor + 1} + 2 = O(n \log n)$$

No Examples Here!

Worst-case of HEAPSORT (TC $6.4-4$)

Show that the worst-case running time of HEAPSORT is $O(n\log n)$.

$$\sum_{r=1}^{n-1} \lfloor \log r \rfloor = n\lfloor \log n \rfloor - 2^{\lfloor \log n \rfloor + 1} + 2 = O(n\log n)$$

No Examples Here!

$$\underbrace{\Theta(n)}_{\text{EXTRACT-MAX}} \times \underbrace{O(\log n)}_{\text{MAX-HEAPIFY}} = O(n\log n)$$

Therefore ...

Worst-case of HEAPSORT (TC $6.4-4$)

Show that the worst-case running time of HEAPSORT is $\Theta(n \log n)$.

	O	Ω	Θ
Worst-case	"power" of $\mathcal{A}$	by example	$O = \Omega$

Best-case of HEAPSORT (TC $6.4-5$)

Show that when all elements are distinct, the best-case running time of HEAPSORT is $\Omega(n \log n)$.

Best-case of HEAPSORT (TC $6.4 - 5$)

Show that when all elements are distinct, the best-case running time of HEAPSORT is $\Omega(n \log n)$.

Consider the largest $m = \lceil n/2 \rceil$ elements.

The largest m elements form a subtree.

Best-case of HEAPSORT (TC $6.4-5$)

Show that when all elements are distinct, the best-case running time of HEAPSORT is $\Omega(n \log n)$.

Consider the largest $m = \lceil n/2 \rceil$ elements.

The largest m elements form a subtree.

$\geq \lfloor m/2 \rfloor$ of m must be nonleaves of that subtree.

Best-case of HEAPSORT (TC $6.4-5$)

Show that when all elements are distinct, the best-case running time of HEAPSORT is $\Omega(n \log n)$.

Consider the largest $m = \lceil n/2 \rceil$ elements.

The largest m elements form a subtree.

$\geq \lfloor m/2 \rfloor$ of m must be nonleaves of that subtree.

$\geq \lfloor m/2 \rfloor$ of m appear in the first $\lfloor n/2 \rfloor$ positions.

Best-case of HEAPSORT (TC $6.4-5$)

Show that when all elements are distinct, the best-case running time of HEAPSORT is $\Omega(n \log n)$.

Consider the largest $m = \lceil n/2 \rceil$ elements.

The largest m elements form a subtree.

$\geq \lfloor m/2 \rfloor$ of m must be nonleaves of that subtree.

$\geq \lfloor m/2 \rfloor$ of m appear in the first $\lfloor n/2 \rfloor$ positions.

They must be promoted to the root before being EXTRACT-MAX.

Best-case of HEAPSORT (TC $6.4 - 5$)

Show that when all elements are distinct, the best-case running time of HEAPSORT is $\Omega(n \log n)$.

Consider the largest $m = \lceil n/2 \rceil$ elements.

The largest m elements form a subtree.

$\geq \lfloor m/2 \rfloor$ of m must be nonleaves of that subtree.

$\geq \lfloor m/2 \rfloor$ of m appear in the first $\lfloor n/2 \rfloor$ positions.

They must be promoted to the root before being EXTRACT-MAX.

$$\sum_{k=1}^{\lfloor m/2 \rfloor} \lfloor \log k \rfloor = \frac{1}{2} m \log m + O(m)$$

Best-case of HEAPSORT (TC $6.4 - 5$)

Show that when all elements are distinct, the best-case running time of HEAPSORT is $\Omega(n \log n)$.

Consider the largest $m = \lceil n/2 \rceil$ elements.

The largest m elements form a subtree.

$\geq \lfloor m/2 \rfloor$ of m must be nonleaves of that subtree.

$\geq \lfloor m/2 \rfloor$ of m appear in the first $\lfloor n/2 \rfloor$ positions.

They must be promoted to the root before being EXTRACT-MAX.

$$\sum_{k=1}^{\lfloor m/2 \rfloor} \lfloor \log k \rfloor = \frac{1}{2} m \log m + O(m)$$

$$B(n) \geq \frac{1}{4} n \log n + O(n) + B(\lfloor n/2 \rfloor)$$

Best-case of HEAPSORT (TC $6.4 - 5$)

Show that when all elements are distinct, the best-case running time of HEAPSORT is $\Omega(n \log n)$.

Consider the largest $m = \lceil n/2 \rceil$ elements.

The largest m elements form a subtree.

$\geq \lfloor m/2 \rfloor$ of m must be nonleaves of that subtree.

$\geq \lfloor m/2 \rfloor$ of m appear in the first $\lfloor n/2 \rfloor$ positions.

They must be promoted to the root before being EXTRACT-MAX.

$$\sum_{k=1}^{\lfloor m/2 \rfloor} \lfloor \log k \rfloor = \frac{1}{2} m \log m + O(m)$$

$$B(n) \geq \frac{1}{4} n \log n + O(n) + B(\lfloor n/2 \rfloor) \implies B(n) \geq \frac{1}{2} n \log n + O(n)$$

Best-case of HEAPSORT (TC $6.4-5$)

Show that when all elements are distinct, the best-case running time of HEAPSORT is $O(n \log n)$.

Best-case of HEAPSORT (TC $6.4-5$)

Show that when all elements are distinct, the best-case running time of HEAPSORT is $O(n \log n)$.

By Example.

Best-case of HEAPSORT (TC $6.4-5$)

Show that when all elements are distinct, the best-case running time of HEAPSORT is $O(n \log n)$.

By Example.

“On the Best Case of Heapsort” (1994)

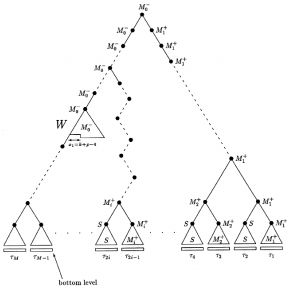


FIG. 2. Initial heap (more detailed).

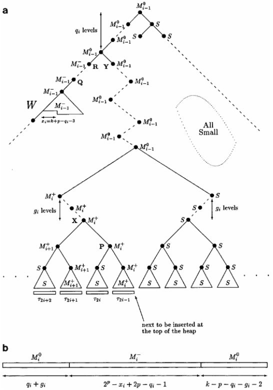


FIG. 3. (a) Odd i; (b) contents of the bottom level of τ_{2i-1}, i odd.

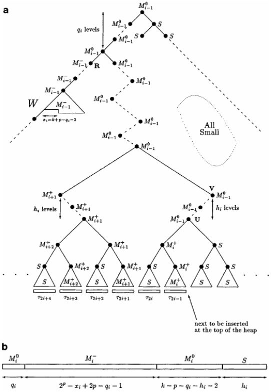


FIG. 4. (a) Even i; (b) contents of the bottom level of τ_{2i-1}, i even.

“On the Best Case of Heapsort” (1994)

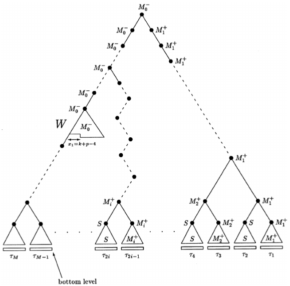


FIG. 2. Initial heap (more detailed).

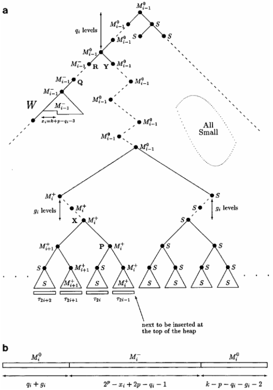


FIG. 3. (a) Odd i; (b) contents of the bottom level of τ_{2i-1}, i odd.

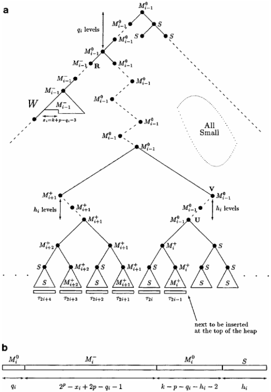


FIG. 4. (a) Even i; (b) contents of the bottom level of τ_{2i-1}, i even.

$$B(n) \leq \frac{1}{2} n \log n + O(n \log \log n)$$

Therefore . . .

Best-case of HEAPSORT (TC $6.4 - 5$)

Show that when all elements are distinct, the best-case running time of HEAPSORT is $\Theta(n \log n)$.

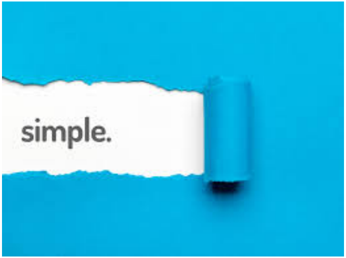
	O	Ω	Θ
Best-case	by example	"weakness" of $\mathcal{A}$	$O = \Omega$

Average-case of HEAPSORT

Assume that all elements are distinct. Show that the average-case running time of HEAPSORT is $\Theta(n \log n)$.

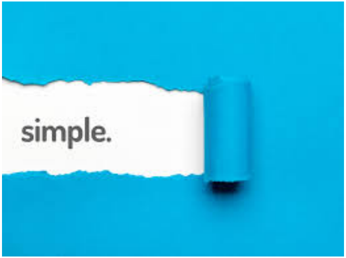
Average-case of HEAPSORT

Assume that all elements are distinct. Show that the average-case running time of HEAPSORT is $\Theta(n \log n)$.



Average-case of HEAPSORT

Assume that all elements are distinct. Show that the average-case running time of HEAPSORT is $\Theta(n \log n)$.



I said simple,
not easy.

"By a surprisingly short counting argument."

Robert Sedgewick

“By a surprisingly short counting argument.”

Robert Sedgewick

D. E. Knuth

“It is elegant.

“By a surprisingly short counting argument.”

Robert Sedgewick

D. E. Knuth

“It is elegant. see exercise 30.”

$$\boxed{f(n) \triangleq \# \text{ of heaps of } n \text{ distinct keys } [1 \cdots n]}$$

$$\boxed{f(n) \triangleq \#\text{ of heaps of } n \text{ distinct keys } [1 \cdots n]}$$

$$f(n) = \binom{n-1}{m} f(m) f(n-1-m)$$

$$\boxed{f(n) \triangleq \#\text{ of heaps of } n \text{ distinct keys } [1 \cdots n]}$$

$$f(n) = \binom{n-1}{m} f(m) f(n-1-m)$$

$$\frac{f(n)}{n!} = \frac{1}{n} \frac{f(m)}{m!} \frac{f(n-1-m)}{(n-1-m)!}$$

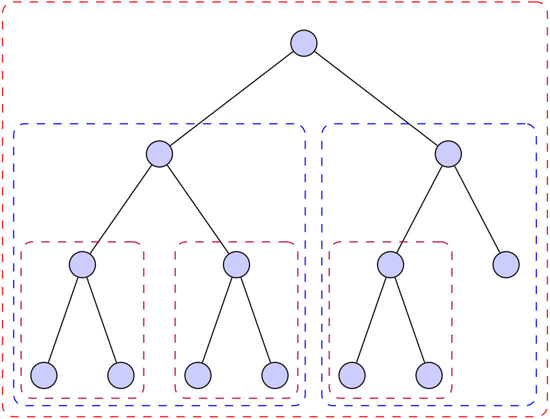
$$\boxed{f(n) \triangleq \# \text{ of heaps of } n \text{ distinct keys } [1 \cdots n]}$$

$$f(n) = \binom{n-1}{m} f(m) f(n-1-m)$$

$$\frac{f(n)}{n!} = \frac{1}{n} \frac{f(m)}{m!} \frac{f(n-1-m)}{(n-1-m)!}$$

$$\boxed{f(n) = \frac{n!}{\prod_{1 \le i \le n} s_i}}$$

$s_i \triangleq$ size of the subtree rooted at i

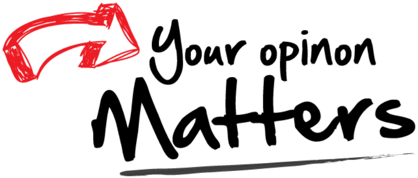


$$f(13) = \frac{13!}{13 \cdot 7 \cdot 5 \cdot 3 \cdot 3 \cdot 3} = 506880$$


TREASURE
HUNT
on foot


TIPS


Thank
You!



Office 302

Mailbox: H016

hfwei@nju.edu.cn