Chapter 7
Quicksort – Average case

Algorithm Analysis

School of CSEE

Exercise 1

What is running time of quicksort?

Worst case	
Best case	
Average case	

Exercise 2

For each of the following cases

(a) the **worst case** of quicksort

(b) the **best case** of quicksort

(c) the **average case** of quicksort

1) Express the time complexity $T(n)$ as a recurrence equation.
2) Find the running time of $T(n)$.

Answer 2-a

The **worst case** of quicksort

1) When the input array is sorted/reversely sorted

$\rightarrow$ One side of the partition has no elements.

$T(n) = T(0) + T(n-1) + \Theta(n)$

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

2) $T(n) = \Theta(n^2)$

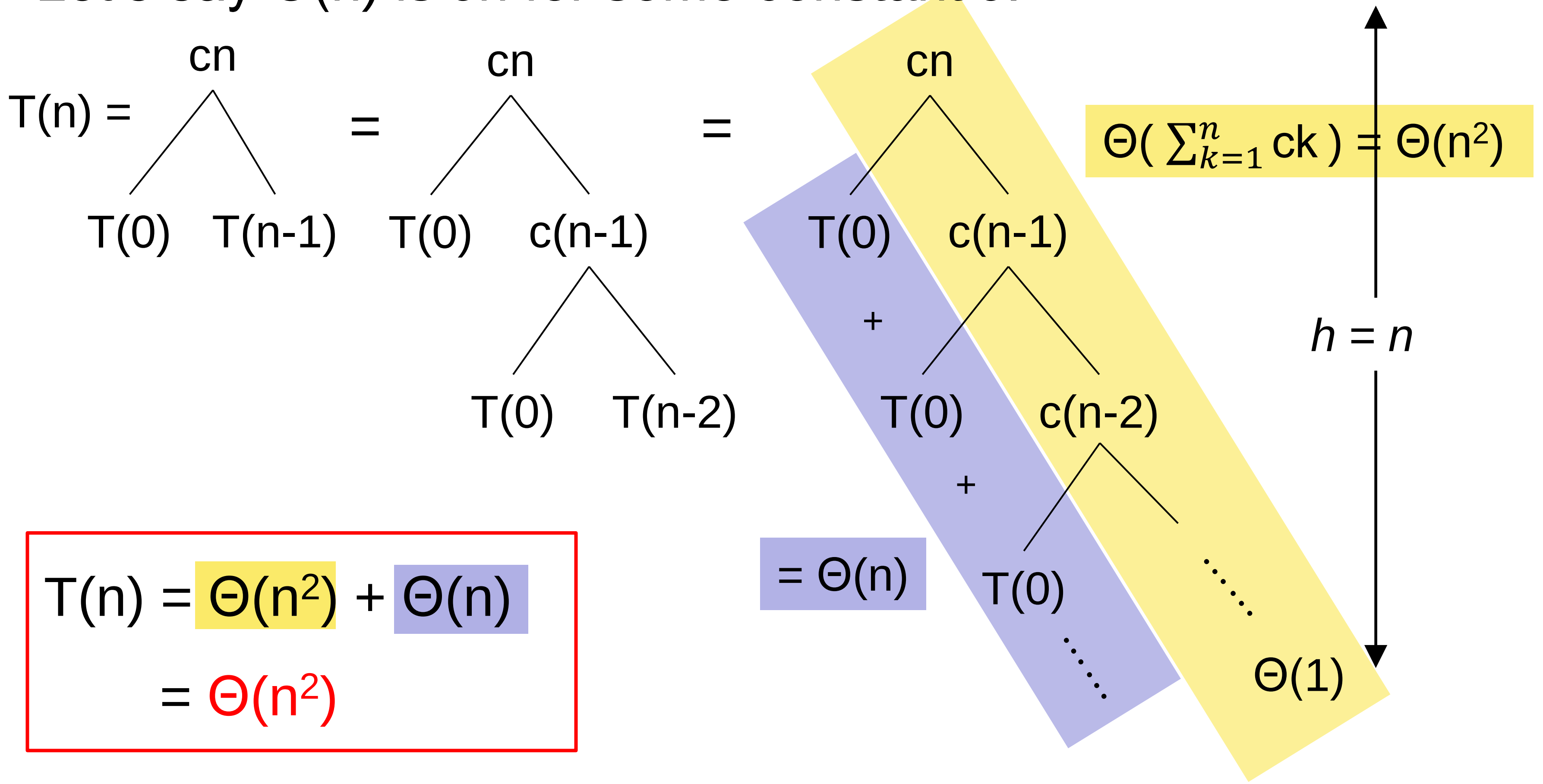
$T(n) = T(n-1) + \Theta(n) = \Theta(n^2)$ (arithmetic series, like selection sort)

Or this can be proven by recursion tree method.

Answer 2-a

$T(n) = T(0) + T(n-1) + \Theta(n)$.

Let's say $\Theta(n)$ is *cn* for some constant *c*.



$\Theta(\sum_{k=1}^{n} ck) = \Theta(n^2)$

$T(n) = \Theta(n^2) + \Theta(n)$

$= \Theta(n^2)$

Answer 2-b

The **best case** of quicksort

1) When the array is partitioned into half, n/2 : n/2

$$T(n) = 2T(\frac{n}{2}) + \Theta(n)$$

2) $T(n) = \Theta(n\lg n)$

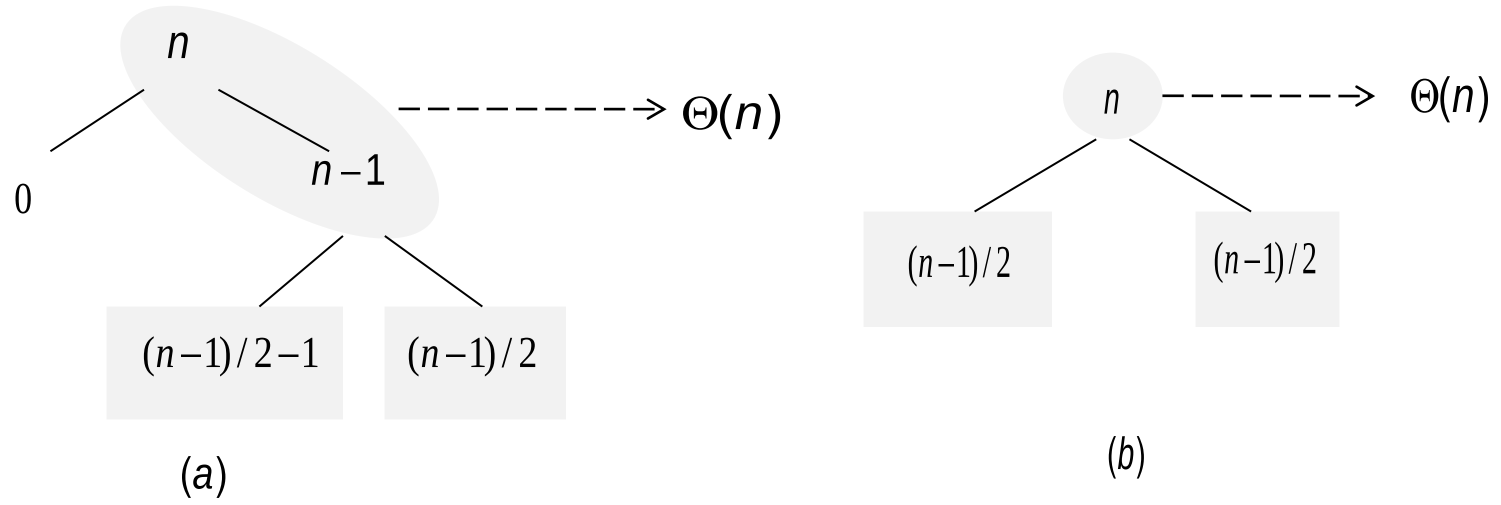
a=2, b=2, so $\boldsymbol{n^{\log_b a} = n^1}$. Since f(n) = Θ(n), case 2 applies.

Thus, $f(n)=\Theta(n\lg n)$.

Answer 2-c

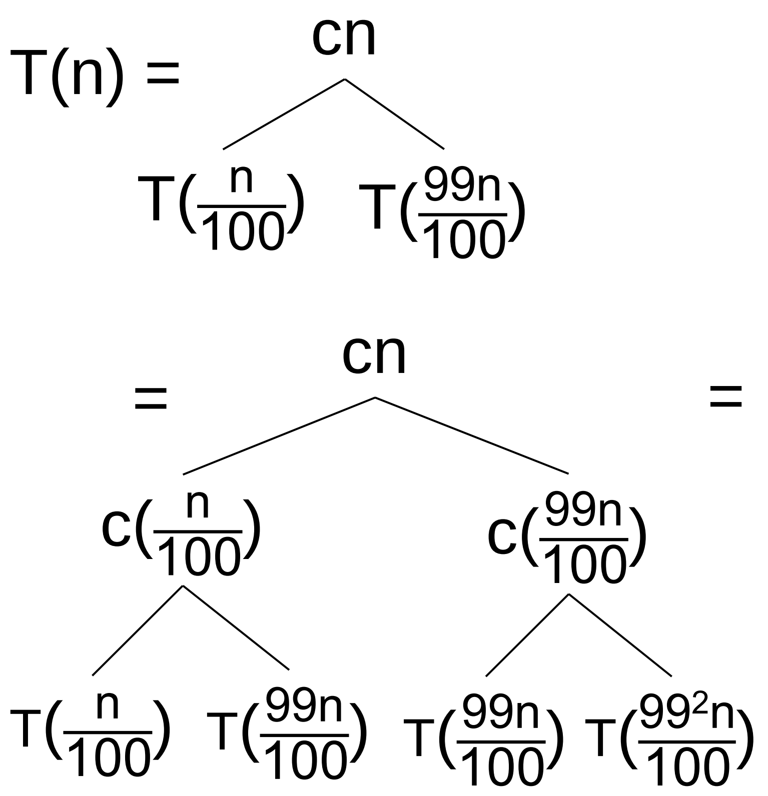
– *What happens if we bad-split root node, then good-split the resulting size (n-1) node?*

- We end up with three subarrays, size 0, (n-1)/2 -1, and (n-1)/2
- Combined cost of splits = $\Theta(n) + \Theta(n - 1) = \Theta(n)$
- No worse than if we had good-split alone!

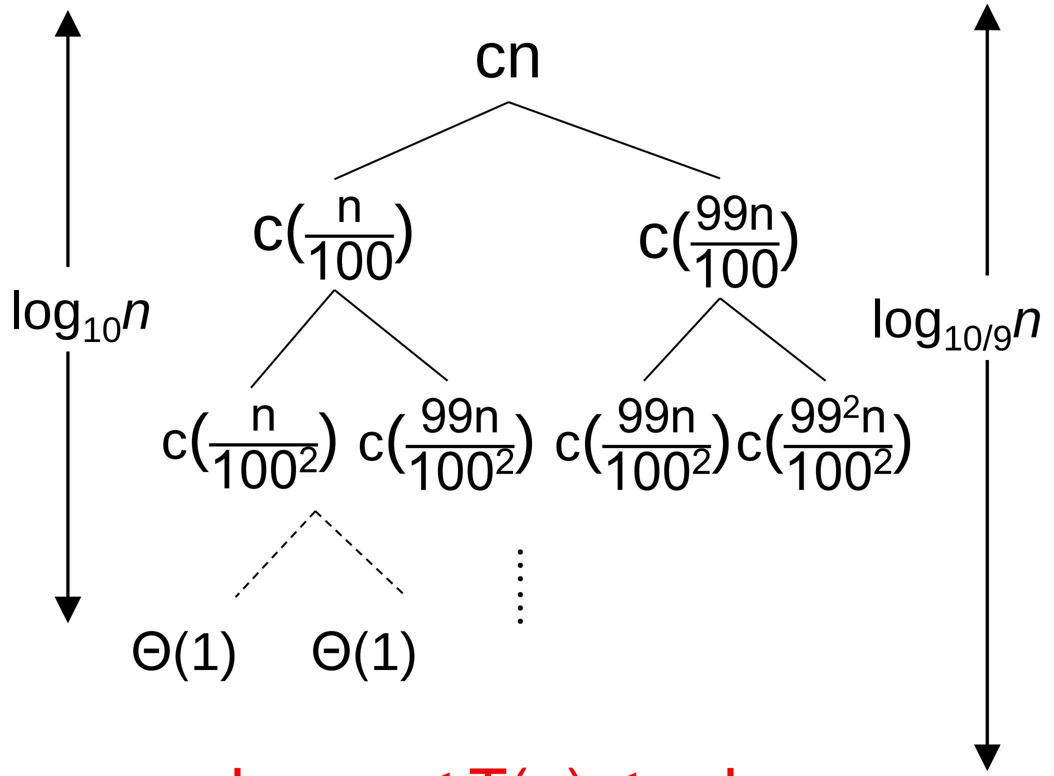


Answer 2-c

$$T(n) = T(\tfrac{n}{100}) + T(\tfrac{99n}{100}) + \Theta(n). \quad \text{Sup. } \Theta(n) = cn \text{ for some } c$$



$T(n) =$
cn
$T(\frac{n}{100})$ $T(\frac{99n}{100})$
$=$
cn
$c(\frac{n}{100})$ $c(\frac{99n}{100})$
$T(\frac{n}{100})$ $T(\frac{99n}{100})$ $T(\frac{99n}{100})$ $T(\frac{99^2n}{100})$
$=$



$\log_{10}n$
cn
$c(\frac{n}{100})$ $c(\frac{99n}{100})$
$c(\frac{n}{100^2})$ $c(\frac{99n}{100^2})$ $c(\frac{99n}{100^2})$ $c(\frac{99^2n}{100^2})$
$\Theta(1)$ $\Theta(1)$
$\log_{10/9}n$

$$cn\log_{10}n \le T(n) \le cn\log_{10/9}n$$

$$T(n) = \Theta(n\lg n)$$

Answer 3-c

All the cases of T(n):

$T(n) = T(0) + T(n-1) + \Theta(n)$ if) 0 : n-1 split

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

$= \vdots$

$= T(n-1) + (0) + \Theta(n)$ if) n-1 : 0 split

Probability of each case: 1/n

$$\Rightarrow T(n) = \frac{1}{n} \times \sum_{k=0}^{n-1} \left[T(k) + T(n-k-1) + \Theta(n) \right]$$

$$= \frac{1}{n} \times \sum_{k=0}^{n-1} \left[T(k) + T(n-k-1) \right] + \Theta(n)$$

Analyzing Quicksort: Average Case

- For simplicity, assume:
  - All inputs are distinct (no repeats)
  - `Randomized-partition()` procedure
    - partition around a random element from the subarray.
    - all splits (0:*n*-1, 1:*n*-2, 2:*n*-3, … , *n*-1:0) are equally likely to happen.
- *What is the probability of a particular split happening?*
- Answer: 1/*n*

Chapter 8
Sorting in linear time

Algorithm Analysis

School of CSEE

Exercise 1

Explain the following terms and give 1 algorithm that is relevant.

(a) sort in place

(b) stable sort

Answer 1

(a) sort in place

If only a constant number of elements of the input are ever stored outside the array

ex) insertion sort, heap sort, quick sort

(b) stable sort

Numbers with the same value appear in the output array in the same order as they do in the input array.

ex) insertion sort, merge sort, counting sort

Exercise 2

What is the result of **C** array after A has been counting sorted?

A =

2	**7**	**0**	**2**	**3**	**5**	**3**	**6**	**5**	**5**

```
1     let C[0,..,k] be a new array
2     for i = 0 to k
3         C[ i ] = 0
4     for j = 1 to A.length
5         C[A[ j ]] = C[A[ j ]] + 1
6     for i = 1 to k
7         C[i] = C[i] + C[i – 1]
8     for j = A.length downto 1
9         B[C[A[ j ]]] = A[ j ]
10        C[A[ j ]] = C[A[ j ]] – 1
```

Answer 2

	1	2	3	4	5	6	7	8	9	10
A =	**2**	**7**	**0**	**2**	**3**	**5**	**3**	**6**	**5**	**5**

	0	1	2	3	4	5	6	7	
C =	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	After line 3

	0	1	2	3	4	5	6	7	
C =	**1**	**0**	**2**	**2**	**0**	**3**	**1**	**1**	After line 5

	0	1	2	3	4	5	6	7	
C =	**1**	**1**	**3**	**5**	**5**	**8**	**9**	**10**	After line 7

	1	2	3	4	5	6	7	8	9	10
B =	**0**	**2**	**2**	**3**	**3**	**5**	**5**	**5**	**6**	**7**

	0	1	2	3	4	5	6	7	
Result C =	**0**	**1**	**1**	**3**	**5**	**5**	**8**	**9**	After line 10

Exercise 3

In what condition is the time complexity of radix sort, $\Theta(d(n+k))$, for n d-digit numbers where each digit can take up to k possible values?

```
Radix-Sort(A, d)
1  for i = 1 to d
2        Sort array A on digit i
```

Answer 3

Radix sort correctly sorts numbers $\Theta(d(n+k))$ time if the stable sort it uses takes $\Theta(n+k)$ time.

Also, d should be constant, and $k=O(n)$

```
Radix-Sort($A$, $d$)
1  for $i$ = 1 to $d$
2        Stable sort array $A$ on digit $i$
```

d passes, each pass over n numbers: $\Theta(n+k)$

Then, the total time is $\Theta(d(n+k))$.

Chapter 9
Order Statistics

Algorithm Analysis

School of CSEE

Exercise 1

Trace following code and write result.

```
main(){
        int A[max] = {22, 15, 6, 78, 39, 13, 99, 30, 54, 27};
        int result;

        result = RandomizedSelect(A, 0, max-1, 4);
        printf("In main: result is %d\n",result);

}
```

```
int RandomizedSelect(int A[], int p, int r, int i){
    int q, k, j;
    if (p == r) return A[p];
    q = Partition(A, p, r);
    k = q - p + 1;
   printf("p=%d, r=%d, i=%d, q=%d, k=%d\n", p, r, i, q, k);
    if (i == k)   return A[q];
    if (i < k)    return RandomizedSelect(A, p, q-1, i);
    else          return RandomizedSelect(A, q+1, r, i-k);
}
```

Answer 1

RandomizedSelect(A, 0, 9, 4)

0	1	2	3	4	5	6	7	8	9
22	**15**	**6**	**78**	**39**	**13**	**99**	**30**	**54**	**27**

pivot

0	1	2	3	4	5	6	7	8	9
22	**15**	**6**	**13**	**27**	**78**	**99**	**30**	**54**	**39**

p = **0**, r = **9,** i = **4**, q = **4**, k =4-0+1= **5**

0	1	2	3
22	**15**	**6**	**13**

pivot

0	1	2	3
6	**13**	**22**	**15**

p = **0**, r = **3,** i = **4**, q = **1**, k =1-0+1= **2**

2	3
22	**15**

pivot

2	3
15	**22**

p	r	i	q	k
0	9	4	4	5
0	3	4	1	2
2	3	2	2	1

p = **2**, r = **3,** i = (i-k=) **2**, q = **2**, k =2-2+1= **1**

Then, call Rand…(A, 3(q+1), 3(r), 1), and since p==r, A[3](=22) is answer.

Exercise 2

What is the minimum number of comparisons that is needed to find median of 5 elements?

Answer 2

CEX *i,j* compares the keys with indexes *i* and *j* and interchanges them if necessary so that the smaller key is in the position with the smaller index.

```
CEX 1,2
CEX 3,4
CEX 1,3
If an interchange occurs here, also interchange E[2] and E[4].
// E[1] is smaller than three other keys; it can be rejected.
// We know E[3] < E[4].
CEX 2,5
CEX 2,3
If an interchange occurs here, also interchange E[4] and E[5].
// E[2] is smaller than three other keys; it can be rejected.
// We know E[3] < E[4].
CEX 3,5
// Now E[3] is the smallest of the three remaining keys; it is the median.
```