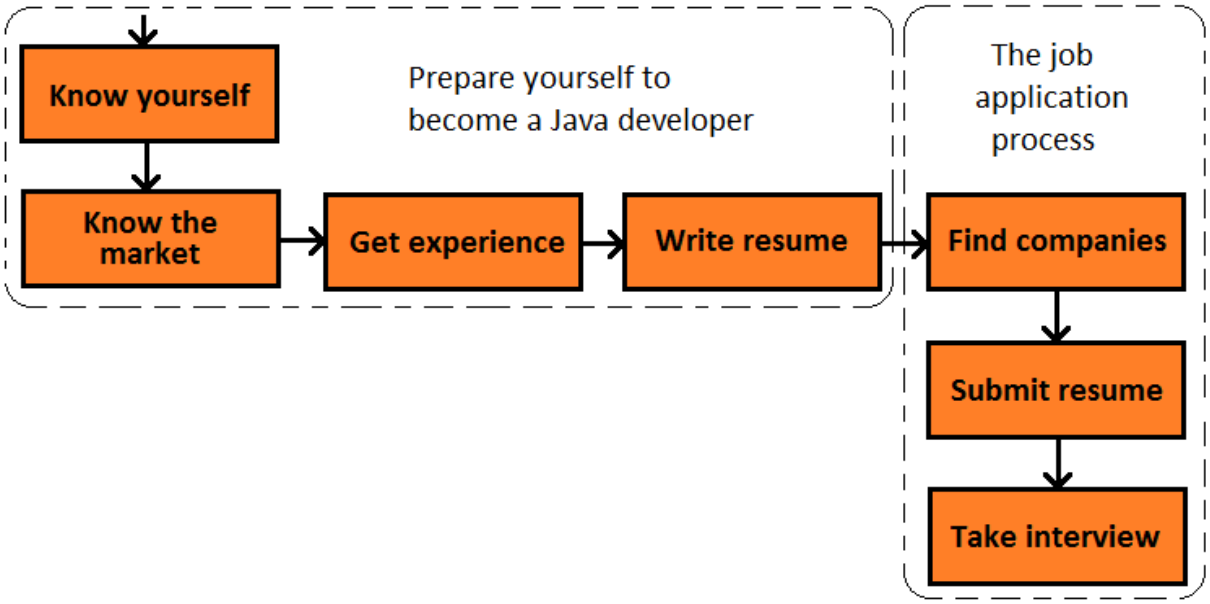
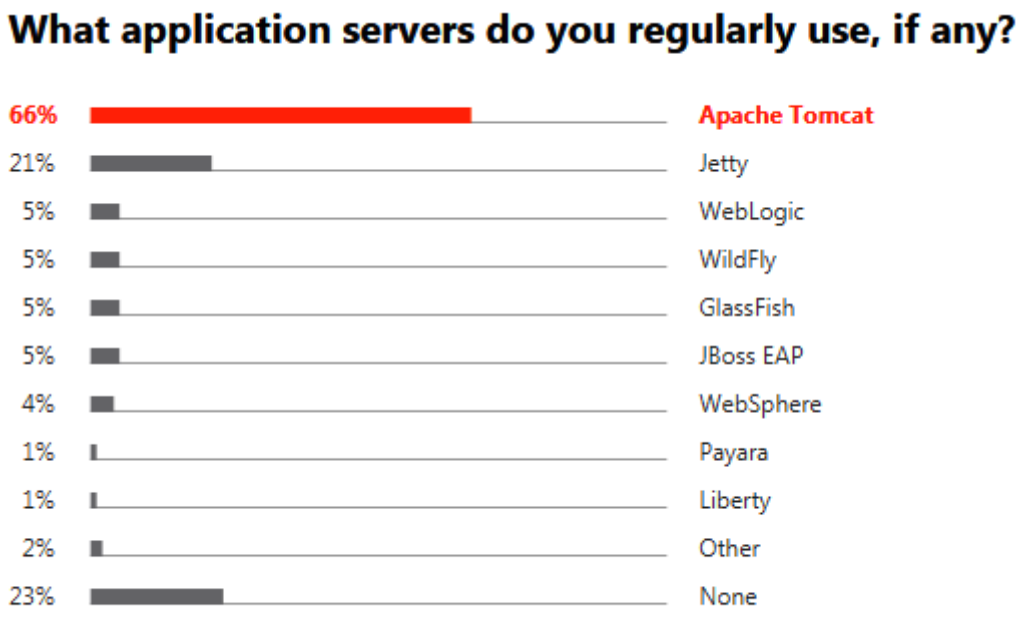
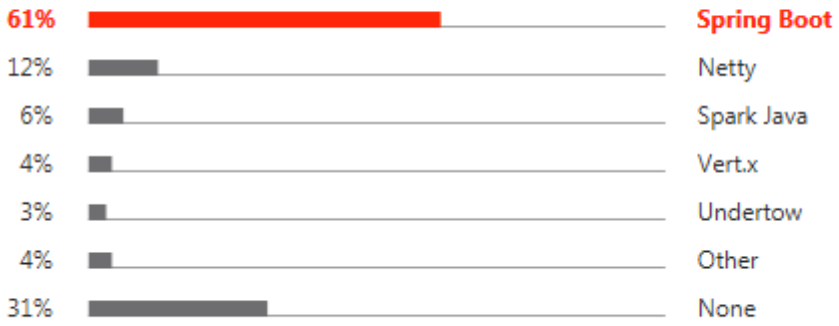
Chapter 1: Where to Start and How to Prepare for the Interview





https://www.jetbrains.com/lp/devecosystem-2019/java/

Which frameworks do you use as an alternative to an application server, if any?



https://www.jetbrains.com/lp/devecosystem-2019/java/

Popular

Technology *Foo*
Technology *Buzz*
...

Skip for now

Technology *Bizz*
...

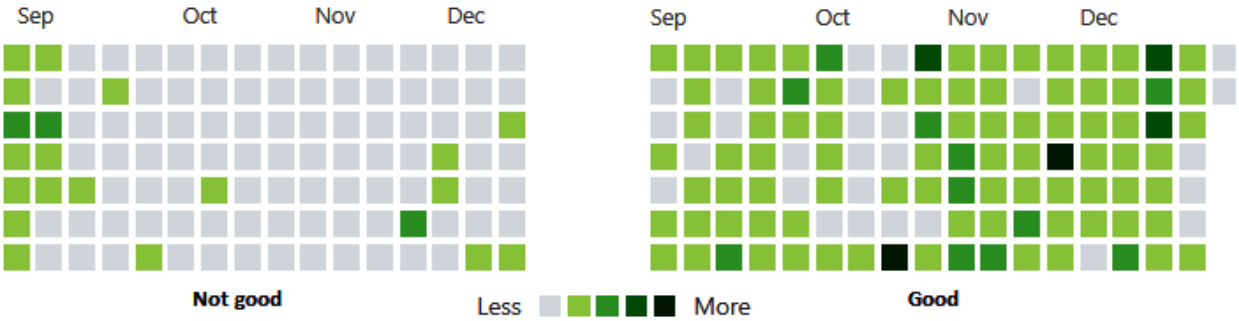
github.com/hibernate/hibernate-orm/pulls

hibernate / hibernate-orm

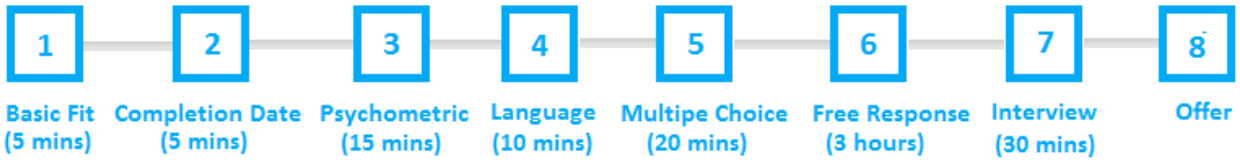
Code | Pull requests 161 | Actions | Wiki | Security | Insights

First time contributing to hibernate/hibernate-orm?

If you would like to submit code to this repository, consider opening a pull request. You can read this repository's contributing guidelines to learn how to open a good pull request.



Chapter 2: What Interviews at Big Companies Look Like



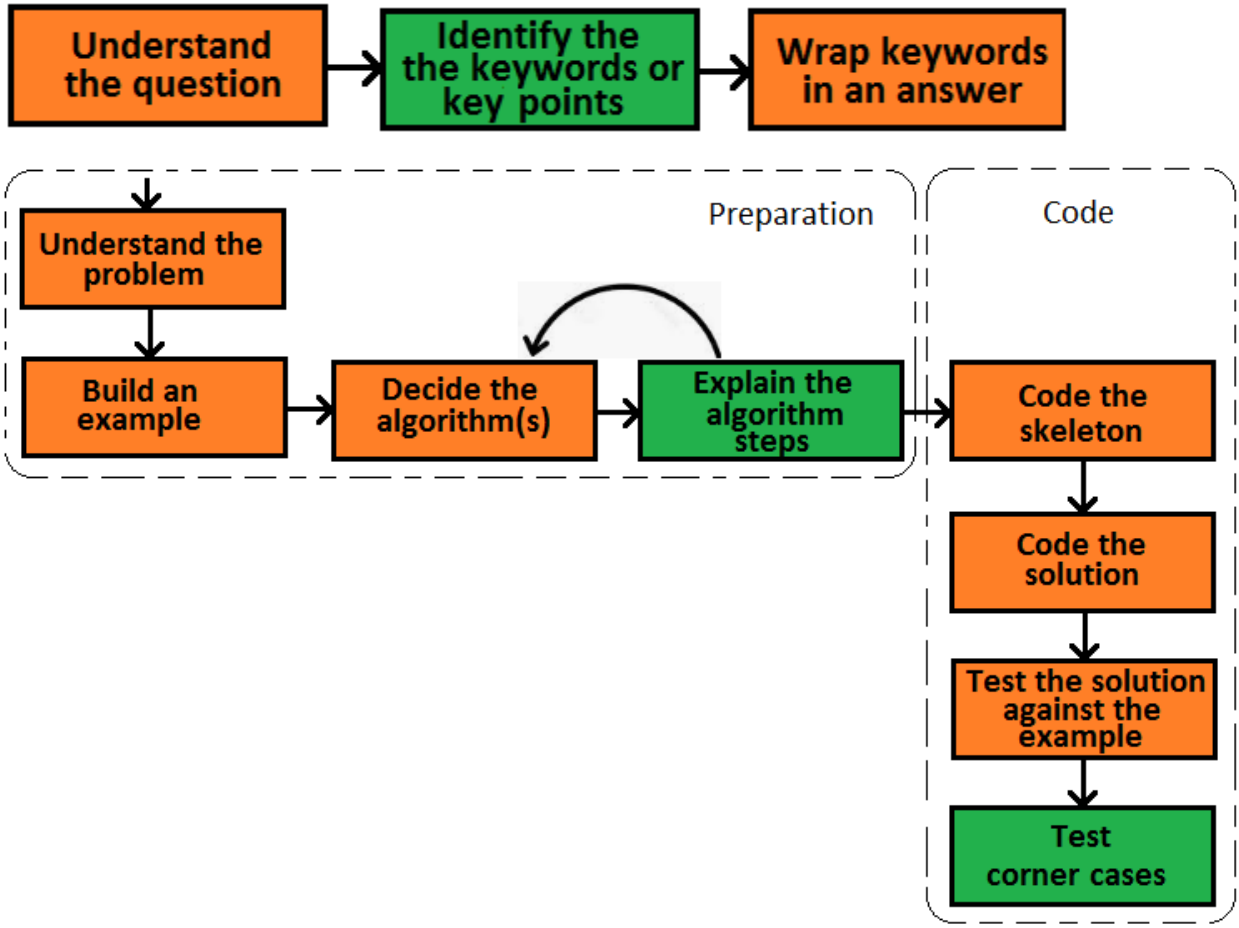
Chapter 3: Common Non-Technical Questions and How To Answer Them

No images…

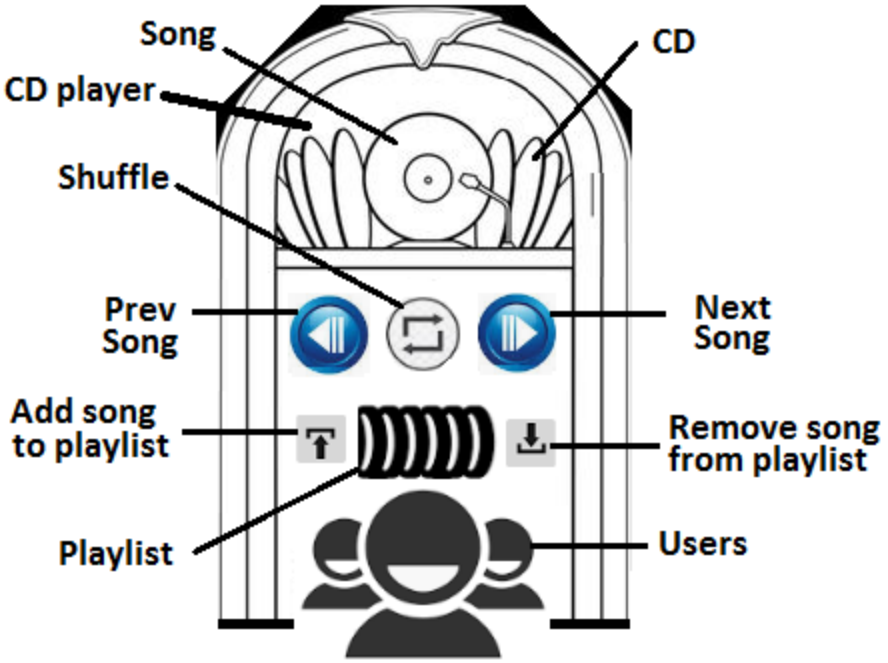
Chapter 4: How to Handle Failures

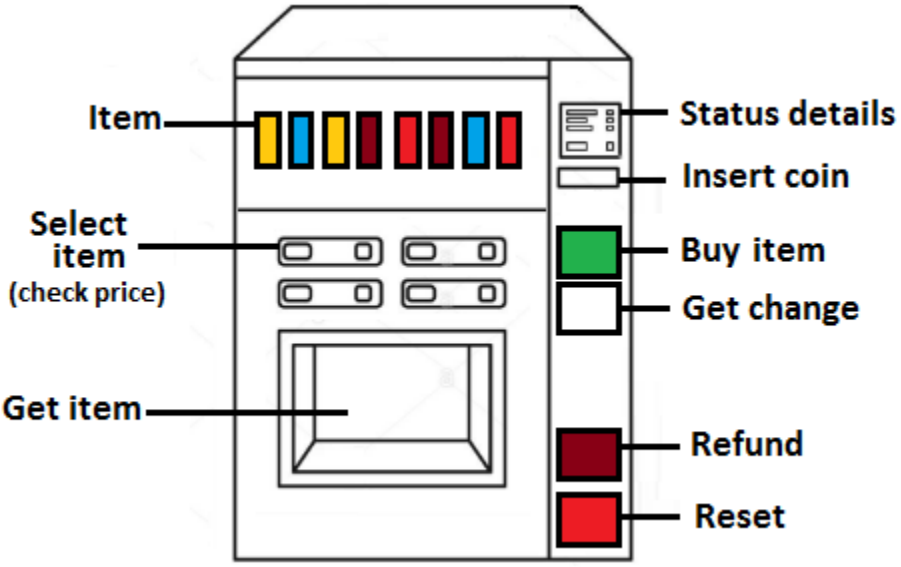
No images…

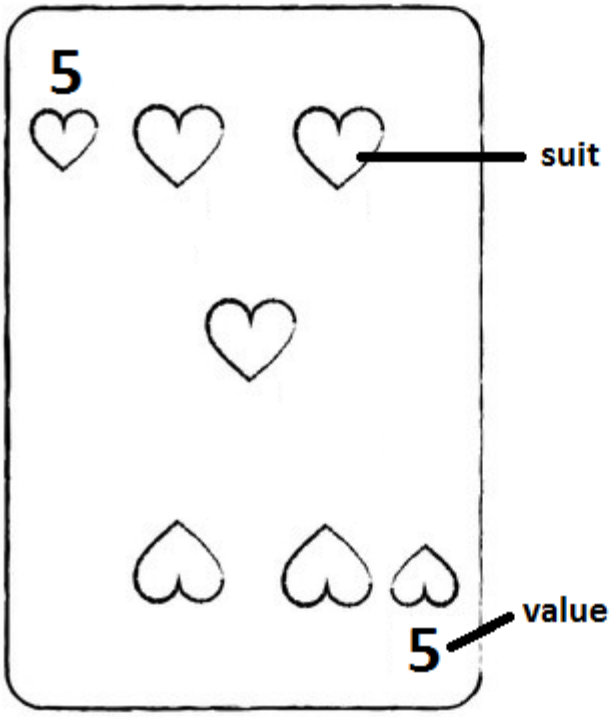
Chapter 5: How to Approach a Coding Challenge

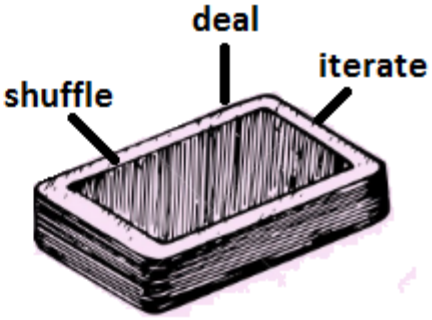


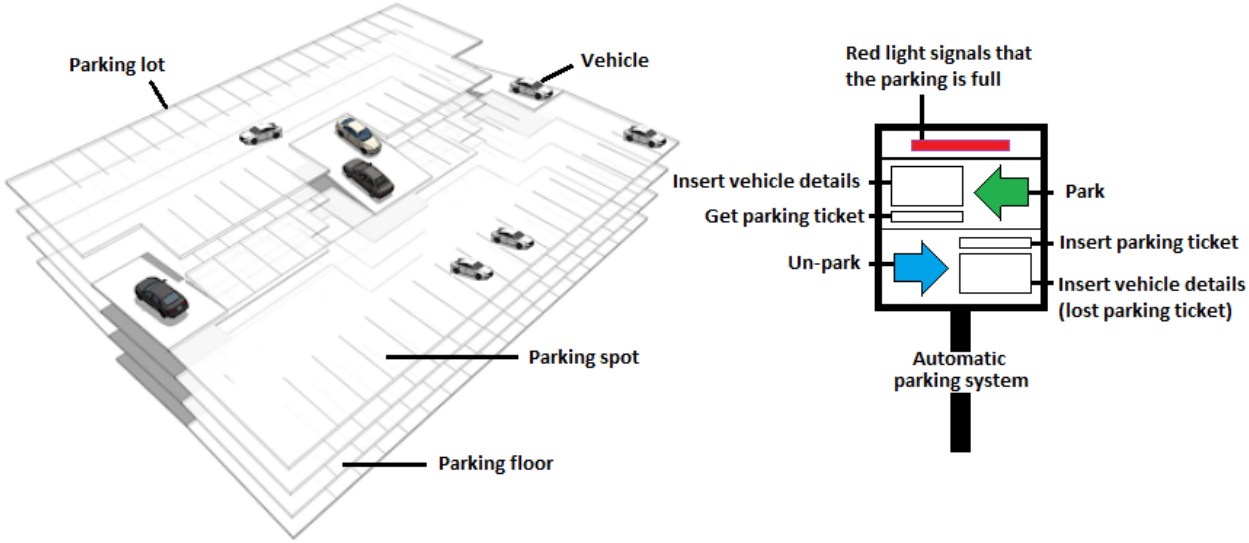
Chapter 6: Object-Oriented Programming

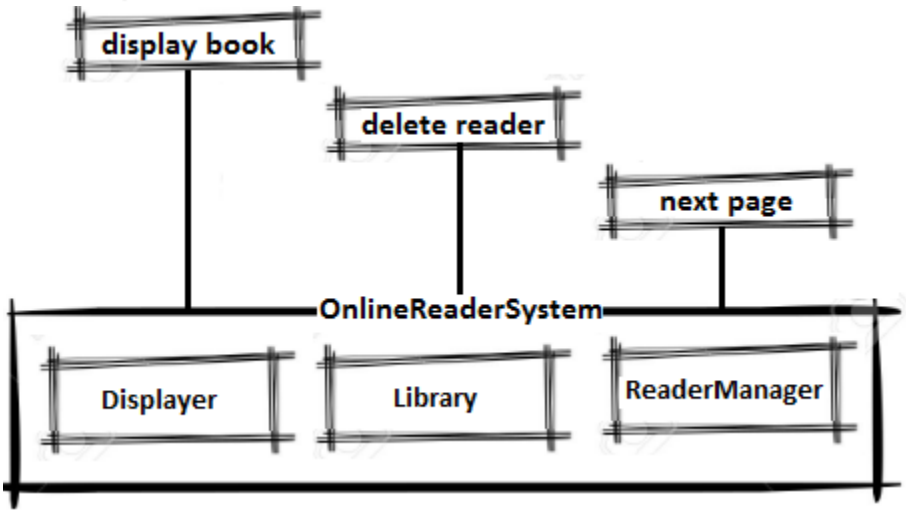


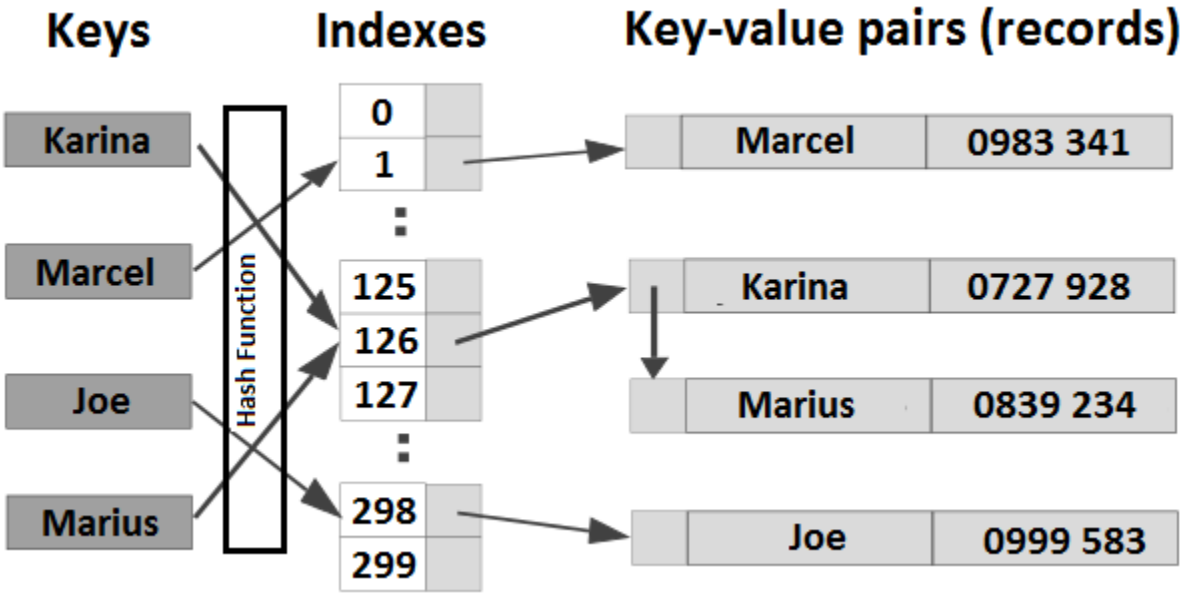



5
suit
value
5

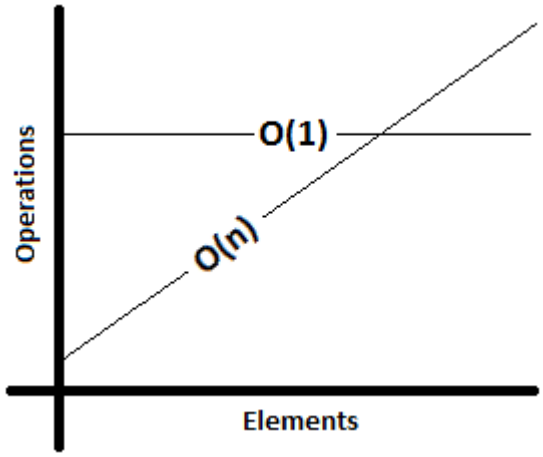

deal
iterate
shuffle

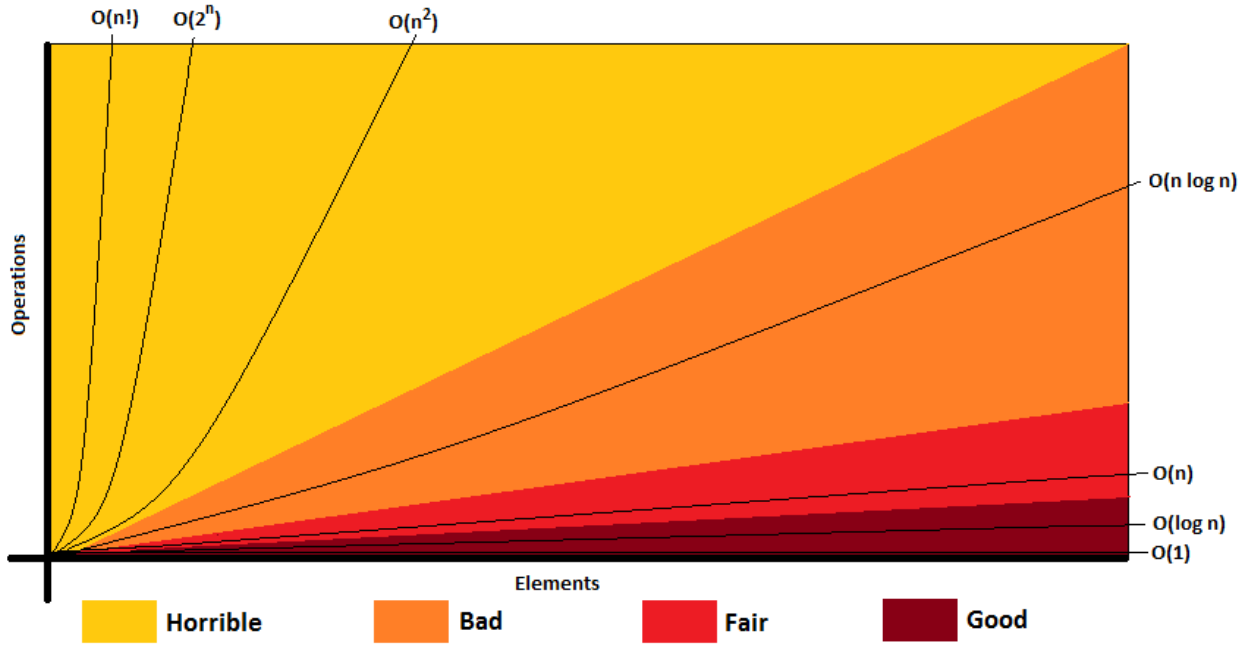

Parking lot
Vehicle
Parking spot
Parking floor
Red light signals that the parking is full
Insert vehicle details
Get parking ticket
Un-park
Park
Insert parking ticket
Insert vehicle details (lost parking ticket)
Automatic parking system


display book
delete reader
next page
OnlineReaderSystem
Displayer
Library
ReaderManager


Keys
Indexes
Key-value pairs (records)
Karina
Marcel
Joe
Marius
Hash Function
0
1
125
126
127
298
299
Marcel
0983 341
Karina
0727 928
Marius
0839 234
Joe
0999 583

Chapter 7: Big O Analysis of Algorithms





```java
// snippet 1
int min = Integer.MAX_VALUE;
int max = Integer.MIN_VALUE;
for (int i = 0; i < a.length; i++) {
     if (a[i] < min) {
         min = a[i];
     }
     if (a[i] > max) {
         max = a[i];
     }
}
```

```java
// snippet 2
int min = Integer.MAX_VALUE;
int max = Integer.MIN_VALUE;
for (int i = 0; i < a.length; i++) {
     if (a[i] < min) {
         min = a[i];
     }
}

for (int i = 0; i < a.length; i++) {
     if (a[i] > max) {
         max = a[i];
     }
}
```

```java
for (int i = 0; i<a.length; i++) {
   System.out.println(a[i]);
}
```

O(n)

```java
for (int i = 0; i<a.length; i++) {
    for (int j = 0; j <a.length; j++) {
        System.out.println(a[i] + a[j]);
    }
}
```

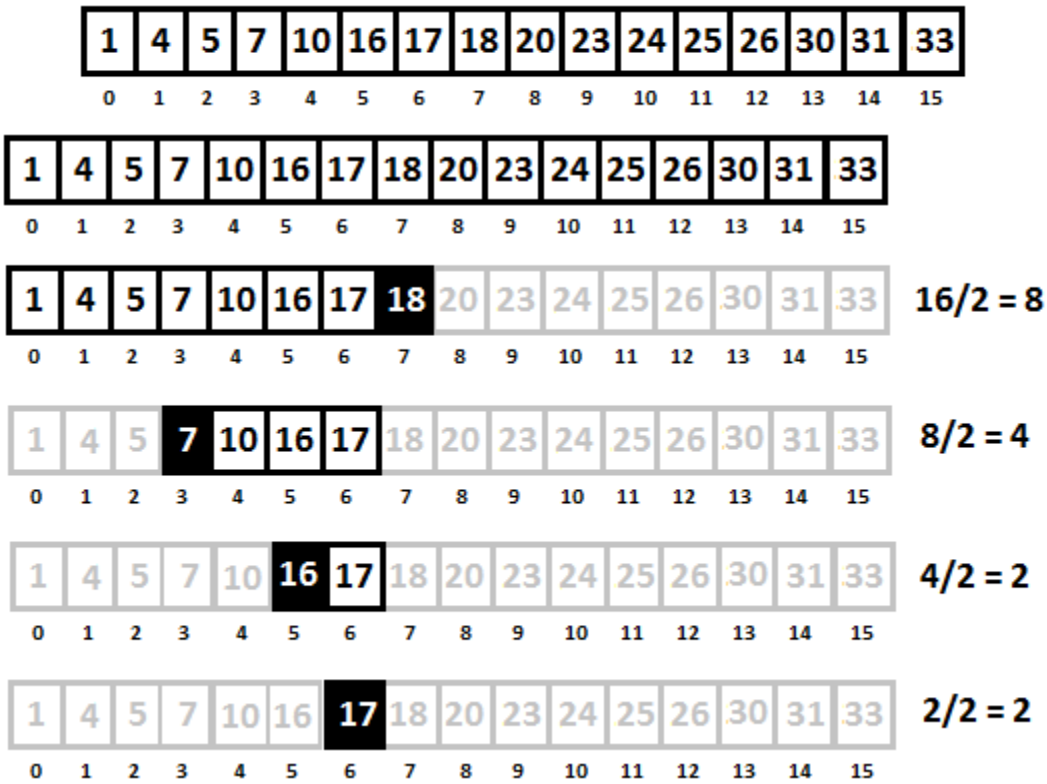
$O(n^2)$

```java
// snippet 1
for(int i=0; i<a.length; i++){
}

for(int i=0; i<a.length; i++){
}
```

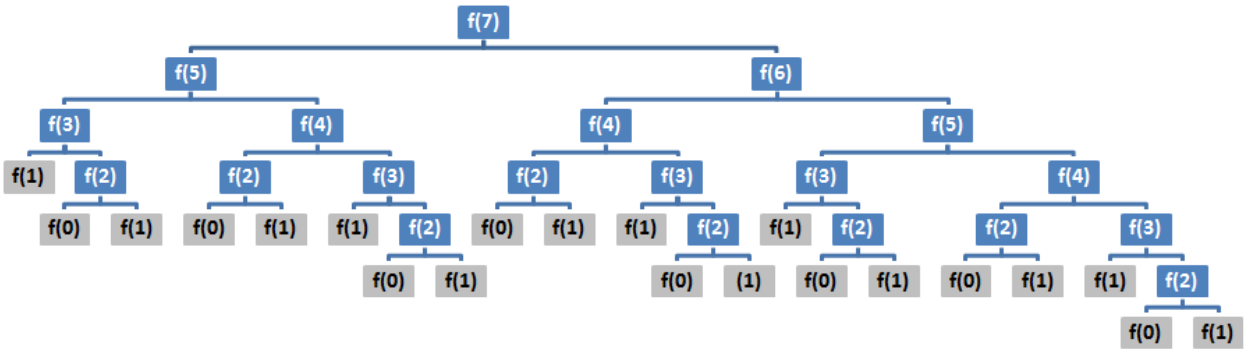
```java
// snippet 2
for(int i=0; i<a.length; i++){
}

for(int i=0; i<b.length; i++){
}
```

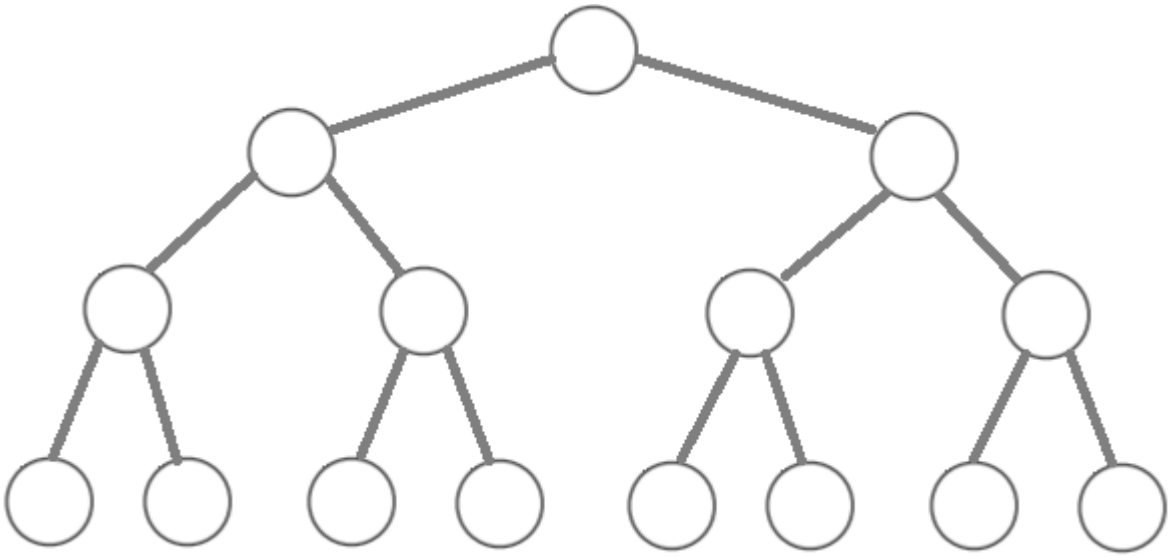
```java
// snippet 1
for (int i=0;i<a.length;i++){
  System.out.println(a[i]);
}

for (int j=0;j<b.length;j++){
  System.out.println(b[j]);
}
```

```java
// snippet 2
for (int i=0;i<a.length;i++) {
  for (int j=0;j<b.length;j++){
    System.out.println(a[i]+b[j]);
  }
}
```



$$n * (1/2)^k = 1 \equiv n*1/2^k = 1 \equiv 2^k*n/2^k = 2^k \equiv 2^k = n$$


f(7)
f(5)
f(6)
f(3)
f(4)
f(4)
f(5)
f(1)
f(2)
f(2)
f(3)
f(2)
f(3)
f(3)
f(4)
f(0)
f(1)
f(0)
f(1)
f(1)
f(2)
f(0)
f(1)
f(1)
f(2)
f(1)
f(2)
f(2)
f(3)
f(0)
f(1)
f(0)
(1)
f(0)
f(1)
f(0)
f(1)
f(1)
f(2)
f(0)
f(1)



```java
// snippet 1
for (int i=0;i<a.length;i++){
 for (int j=0;j<a.length;j++){
  System.out.println(a[i]+a[j]);
 }
}
```

```java
// snippet 2
for (int i=0;i<a.length;i++){
 for (int j=i+1;j<a.length;j++){
  System.out.println(a[i]+a[j]);
 }
}
```

a.length = 5

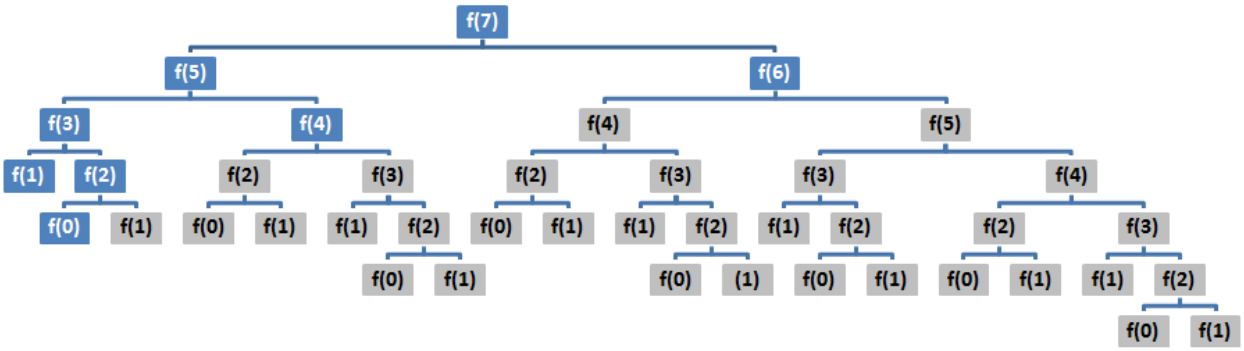
(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)

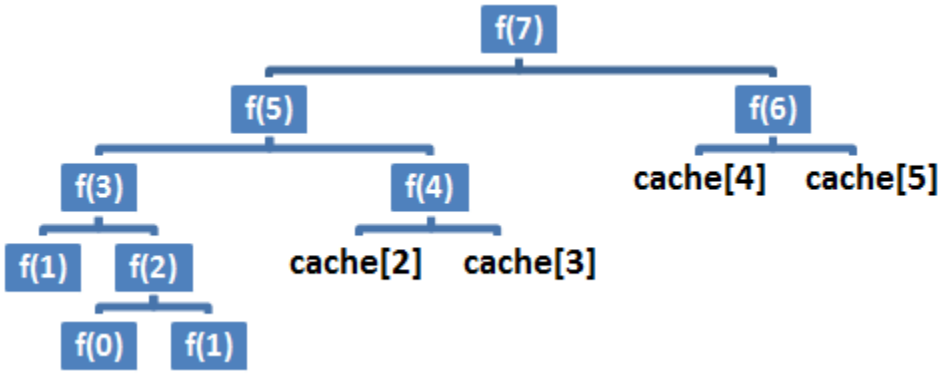
(0, 1)	(0, 2)	(0, 3)	(0, 4)
	(1, 2)	(1, 3)	(1, 4)
		(2, 3)	(2, 4)
			(3, 4)

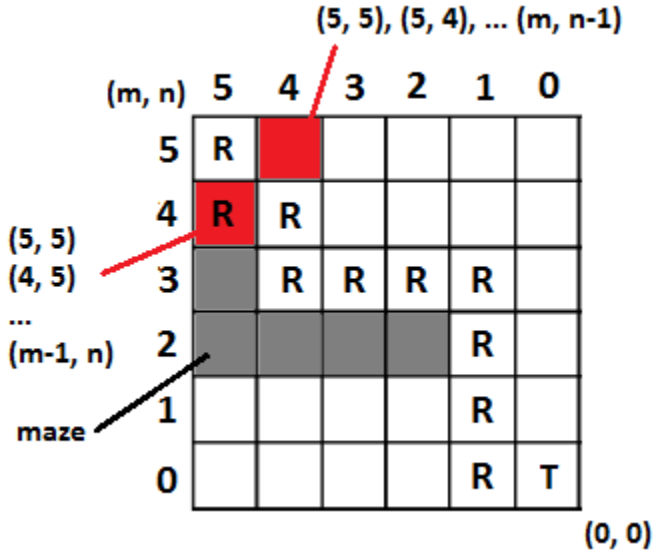
```java
int multiply(int x, int y) {
  int result = 1;
  for (int i=1; i<=y; i++) {
      result *= x;
  }

  return result;
}
```

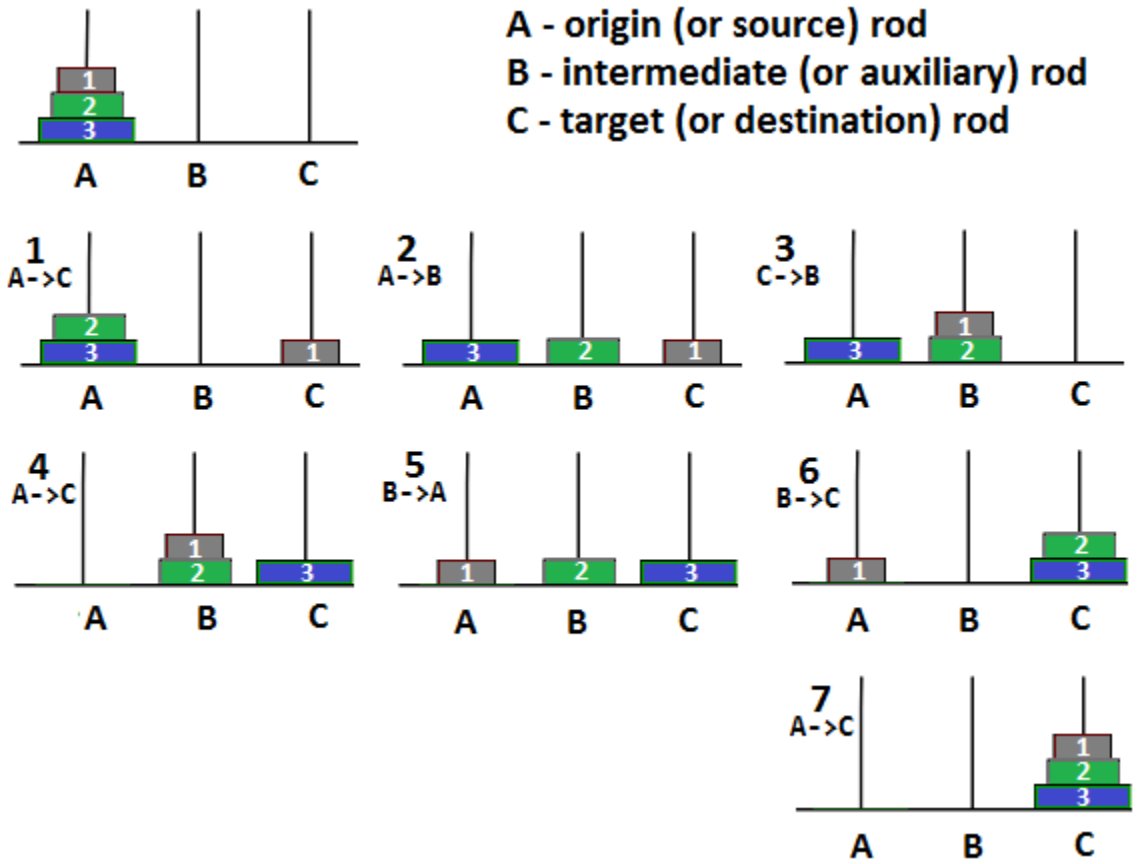
```java
int powerxy(int x, int y) {
    if (y < 0) {
      return 0;
    } else if (y == 0) {
      return 1;
    } else {
      return x*powerxy(x, y-1);
    }
}
```

Chapter 8: Recursion and Dynamic Programming

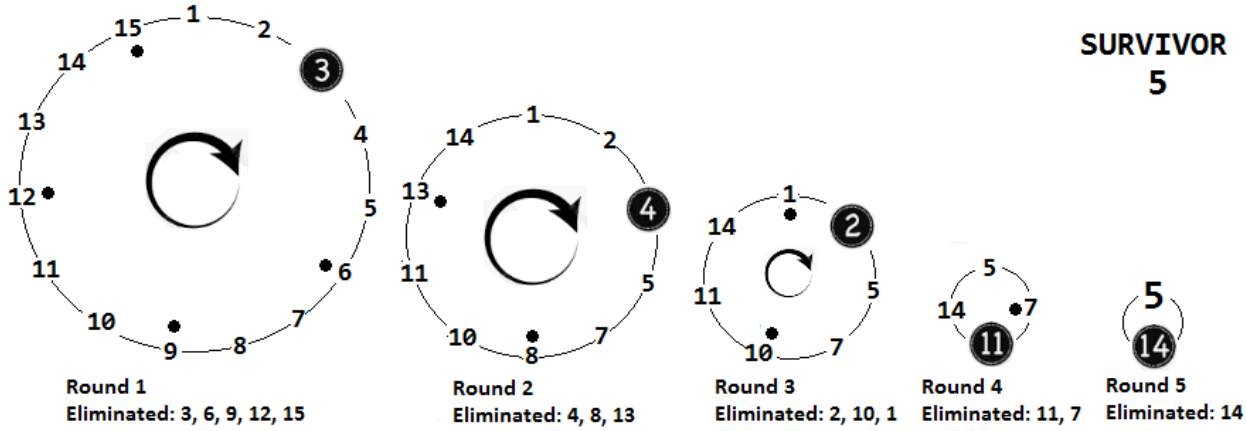








SURVIVOR
5



(0, 0)	0	1	2	3	4
0	1	2	3	3	2
1	2	1	2	3	3
2	2	3	3	3	3
3	3	2	1	1	3
4	1	3	2	3	3

(r, c)

(r-1, c)
(r+1, c)
(r, c-1)
(r, c+1)

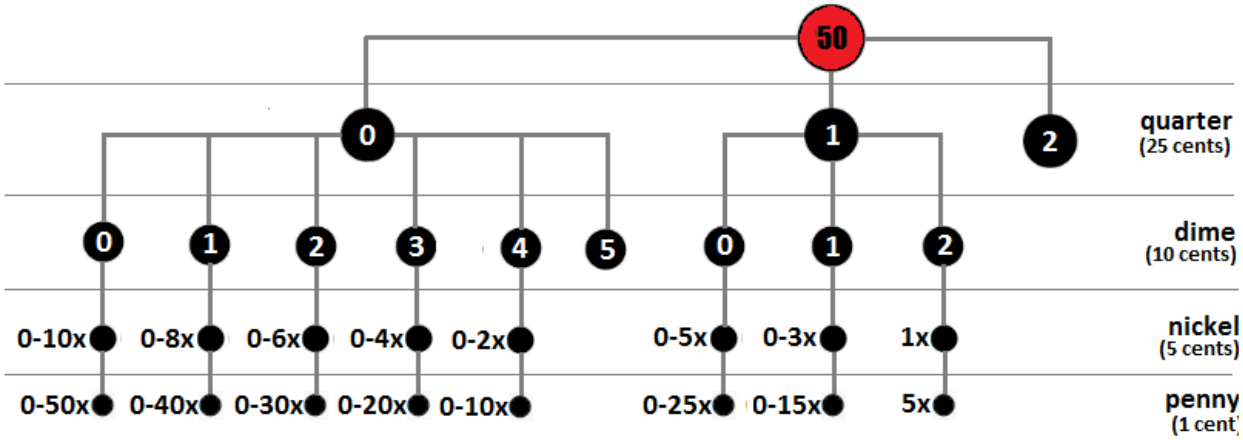
(a)

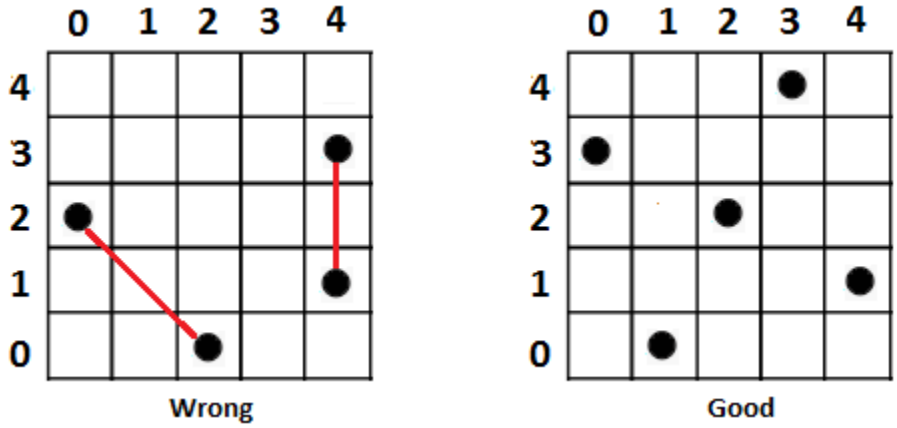
(0, 0)	0	1	2	3	4
0	-1	-2	-3	-3	-2
1	-2	-1	-2	-3	-3
2	-2	-3	-3	-3	-3
3	-3	-2	-1	-1	-3
4	-1	-3	-2	-3	-3

(r, c)

marked as visited by multiplying it with -1

(b)





if we half the array,
the magic index must be on right side

element	-5	-4	-2	0	1	2	3	5	6	7	9	11	13	20	22	24	25	27
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Ordered array, no duplicates

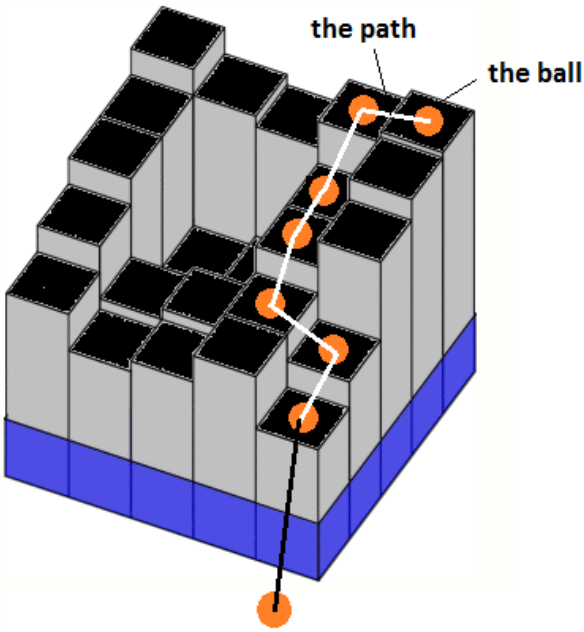
if we half the array,
the magic index can be on both sides

element	1	4	4	4	5	5	6	6	6	11	12	12	12	15	17	20	21	21
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Ordered array with duplicates

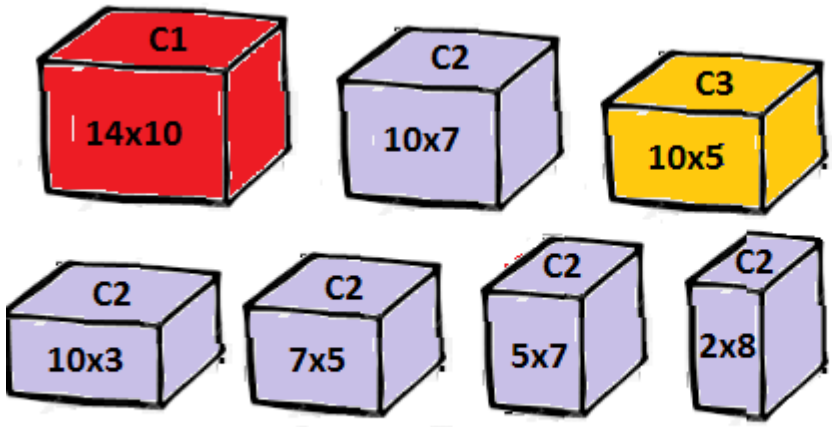
	0	1	2	3	4
4	5	4	3	4	5
3	4	1	1	3	4
2	4	1	1	3	3
1	3	2	2	3	2
0	2	1	1	2	1

initial grid

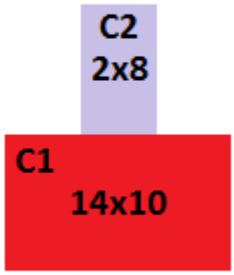


	0	1	2	3	4
4	5	4	0	0	0
3	4	0	0	0	0
2	4	0	0	0	0
1	3	0	0	0	0
0	2	0	0	0	0

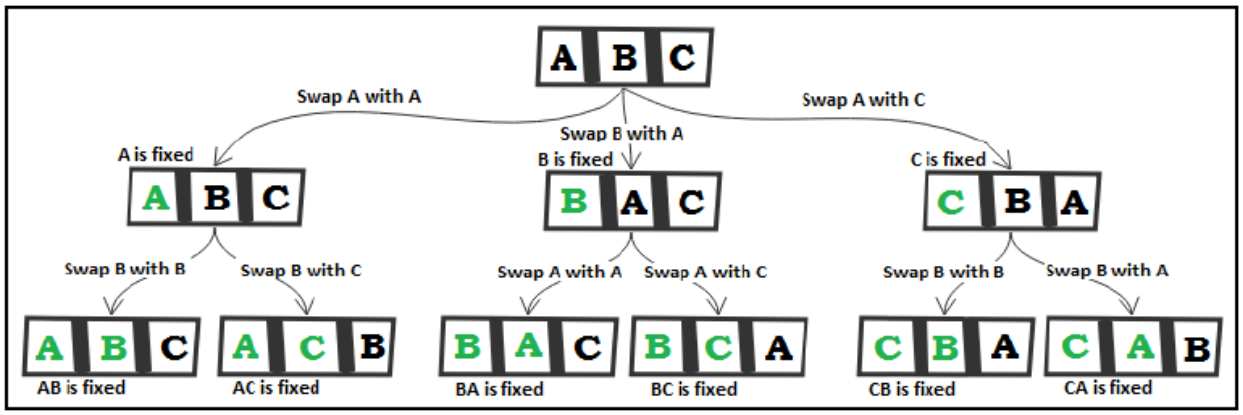
result grid

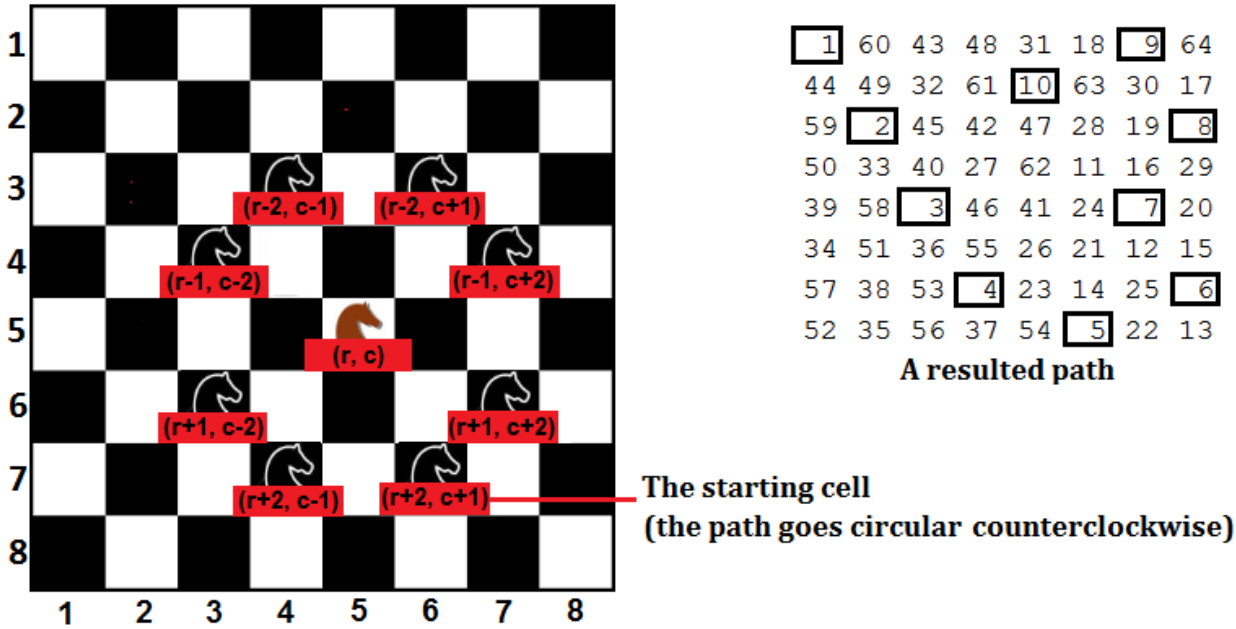


7 boxes of wxh



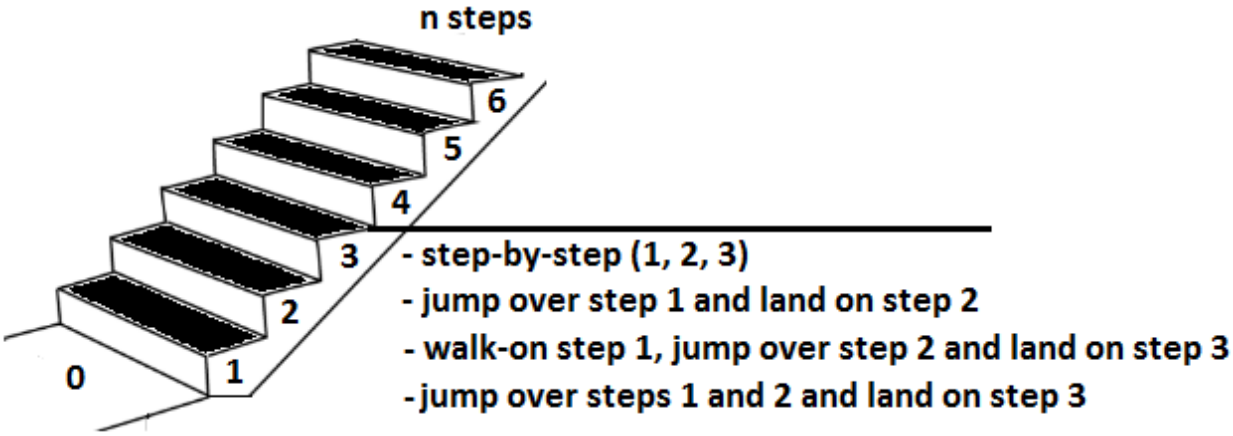
highest tower





1	60	43	48	31	18	9	64
44	49	32	61	10	63	30	17
59	2	45	42	47	28	19	8
50	33	40	27	62	11	16	29
39	58	3	46	41	24	7	20
34	51	36	55	26	21	12	15
57	38	53	4	23	14	25	6
52	35	56	37	54	5	22	13

A resulted path



0	1	2	3	4	5	6	7	8
3	2	7	4	5	1	6	7	9

Sum: 7 Subset: 2 4 1

		0	1	2	3	4	5	6	7	8	9
		0	1	2	3	4	5	6	7	8	9
0	0	T	F	F	F	F	F	F	F	F	F
1	5	T									
2	1	T									
3	6	T									
4	10	T									
5	7	T									
6	11	T									
7	2	T									

		0	1	2	3	4	5	6	7	8	9
		0	1	2	3	4	5	6	7	8	9
0	0	T	F	F	F	F	F	F	F	F	F
1	5	T	F	F	F	F	T	F	F	F	F
2	1	T									
3	6	T									
4	10	T									
5	7	T									
6	11	T									
7	2	T									

		0	1	2	3	4	5	6	7	8	9
		0	1	2	3	4	5	6	7	8	9
0	0	T	F	F	F	F	F	F	F	F	F
1	5	T	F	F	F	F	T	F	F	F	F
2	1	T									
3	6	T									
4	10	T									
5	7	T									
6	11	T									
7	2	T									

		0	1	2	3	4	5	6	7	8	9
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	**0**	T	F	F	F	F	F	F	F	F	F
1	**5**	T	F	F	F	F	T	F	F	F	F
2	**1**	T	T	F	F	F	T	T	F	F	F
3	**6**	T	T	F	F	F	T	T	T	F	F
4	**10**	T	T	F	F	F	T	T	T	F	F
5	**7**	T	T	F	F	F	T	T	T	T	F
6	**11**	T	T	F	F	F	T	T	T	T	F
7	**2**	T	T	T	T	F	T	T	T	T	**T**

		0	1	2	3	4	5	6	7	8	9
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	**0**	T	F	F	F	F	F	F	F	F	F
1	**5**	T	**F**	F	F	F	T	F	F	F	F
2	**1**	T	T	F	F	F	T	T	**F**	F	F
3	**6**	T	T	F	F	F	T	T	T	F	F
4	**10**	T	T	F	F	F	T	T	T	F	F
5	**7**	T	T	F	F	F	T	T	T	T	F
6	**11**	T	T	F	F	F	T	T	T	T	**F**
7	**2**	T	T	T	T	F	T	T	T	T	**T**

9 = 2 + 6 + 1, so a subset is {2, 6, 1}

Chapter 9: Bit Manipulation

00000000 00000000 00000000 00110011

... 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

51

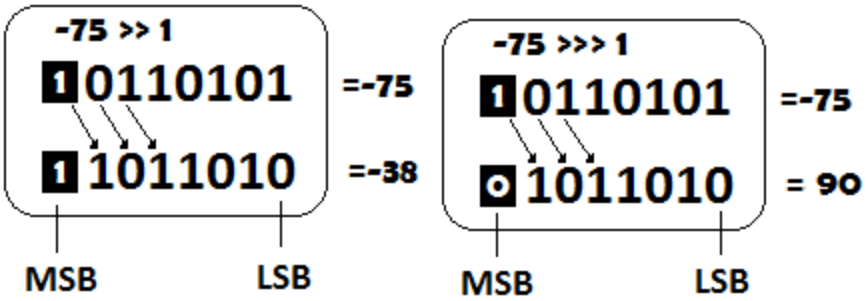
X	Y	X&Y	X|Y	X^Y	~(X)
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

X | 0_s = X	X ^ 0_s = X	X & 0_s = 0
X | 1_s = X	X ^ 1_s = ~X	X & 1_s = X
X | X = X	X ^ X = 0	X & X = X

23 << 3

00000000 00000000 00000000 00010111 = 23

00000000 00000000 00000000 10111000 = 184



Expression	Bit-mask	Example
1 << k	0000...10000000...	k=5, 000...100000
~(1 << k)	111111...01111111...	k=5, 111...011111
(1 << k) - 1	000000...111111...	k=5, 000...11111
-1 << (k + 1)	11111...000000...	k=5, 111...000000

		1						
1	1	1	0	1		1		+
							1	= 1
				1	0	0	1	= 9
		0	1	1	1	0	1	= 29
	1	1	1	1	1	0	0	= 124
1	0	1	0	0	0	1	1	= 163
7	6	5	4	3	2	1	0	

```
                1 1 1 1 1 0 0   *
                  0 1 1 1 0 1
---------------------------------
                1 1 1 1 1 0 0   +
              0 0 0 0 0 0 0 0
            1 1 1 1 1 0 0 0 0
          1 1 1 1 1 0 0 0 0 0
        1 1 1 1 1 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0 0 0
---------------------------------
      1 1 1 0 0 0 0 1 1 0 0 0
```

−

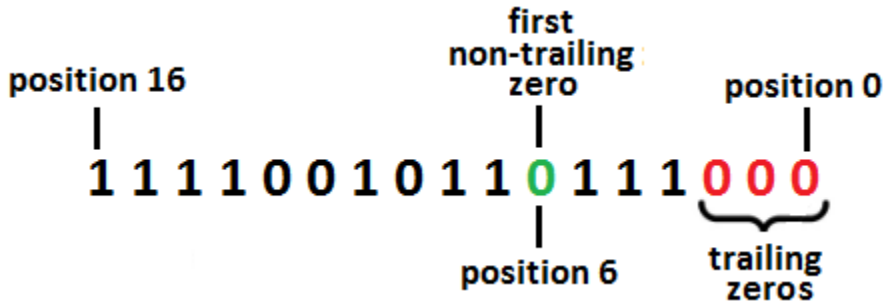
```
                    1                 x 1          1 1 1 x 1
                    1                 1 1          1 1 1 1 1
124 =  1 1 1 1 1 0 0    1 1 1 1 X 0 0    1 1 1 1 X 0 0        1 X X X X 0 0
29 =     0 1 1 1 0 1      0 1 1 1 0 1      0 1 1 1 0 1          0 1 1 1 0 1
-------------------    -------------    -------------   ...  -----------------
                                                 1 1          1 0 1 1 1 1 1   = 95
                                                              6 5 4 3 2 1 0
```

11 ÷ 2 = 5 remainder 1

```
quotient                                                          quotient
                0101                  quotient                    0101.1
          10 | 1011              divisor | dividend          10 | 1011
             -10                          ...                     -10
               011                            ...                   011
             -  10                          remainder             -  10
                 1                                                    10
remainder                                                            -10
                                                                      0
```

```
                j=9        i=4
q       1001100110010
                 |          |
p                111111 <----
                 |          |
result  1001111110010
```

67534	10000011111001110	5
67	1000011	2
339809	10100101111011 00001	9



Given integer:

a) 0 0 0 1 1 0 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1

b) Shift left with 10 positions:

padded zeros

1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 0 0 1 0 0

fallen bits

c) Expected result:

rotated bits

1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 1 1 0 0 1 0 0

1. The given array

[51, 14, 14, 51,98, 7, 14, 98, 51, 98]

2. Sum bits:

	1	1	0	0	1	1	(51)
			1	1	1	0	(14)
			1	1	1	0	(14)
	1	1	0	0	1	1	(51)
1	1	0	0	0	1	0	(98)
				1	1	1	(7)
			1	1	1	0	(14)
1	1	0	0	0	1	0	(98)
	1	1	0	0	1	1	(51)
1	1	0	0	0	1	0	(98)
3	6	3	3	4	10	4	

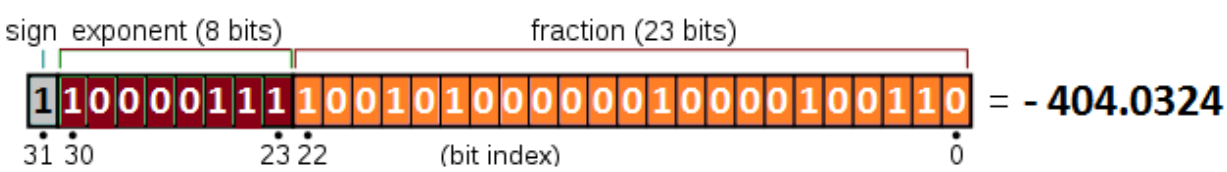
3. Compute % 3

3 % 3 = 0
6 % 3 = 0
3 % 3 = 0
3 % 3 = 0
4 % 3 = 1
10 % 3 = 1
4 % 3 = 1

4. Result

0000111 = 7

n	n	n-1	n&(n-1)
0	0000	0000	0000
1	0001	0000	0000
2	0010	0001	0000
3	0011	0010	0010
4	0100	0011	0000
5	0101	0100	0100
...			
8	1000	0111	0000
9	1001	1000	1000
...			
15	1111	1110	1110
16	10000	1111	0000
...			



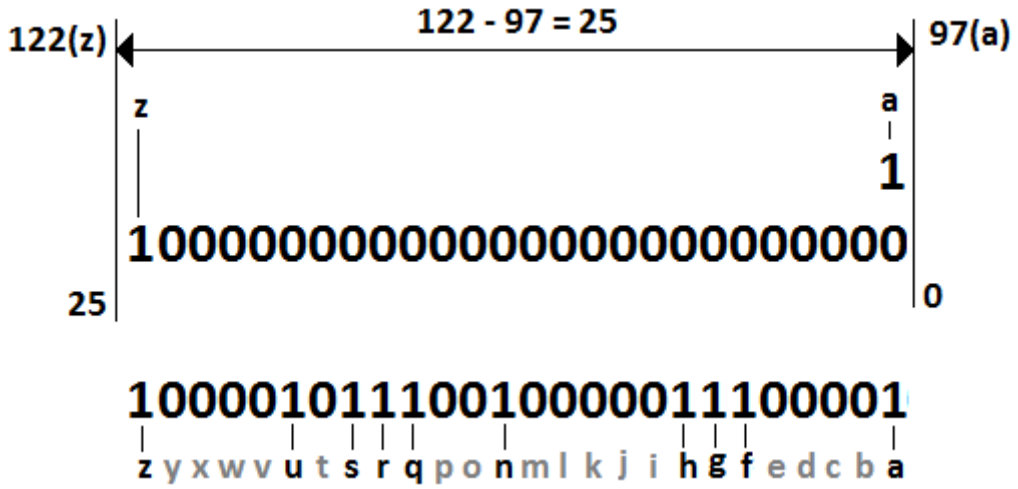
= - 404.0324

Chapter 10: Arrays and Strings

a b c d

豈更車賈滑更

a 豈 b 更₩



Two Hearts 

Java Representation: \uD83D\uDC95
Code point: 128149 (56469)

Musical Score 

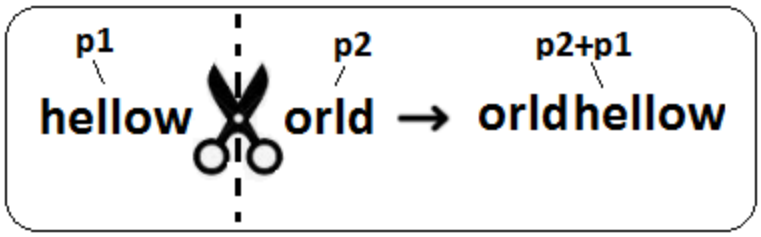
Java Representation: \uD83C\uDFBC
Code Point: 127932 (57276)

Smiley Face 

Java Representation: \uD83D\uDE0D
Code Point: 128525 (56845)

Cyrillic Capital Letter ZHE with Diaeresis 

Java Representation: \u04DC
Code Point: 1244



$$[M^T]_{ij} = [M]_{ji}$$

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

M

$$[M]_{ji} = [M^T]_{ij}$$

1	6	11	16	21
2	7	12	17	22
3	8	13	18	23
4	9	14	19	24
5	10	15	20	25

M^T

5	10	15	20	25
4	9	14	19	24
3	8	13	18	23
2	7	12	17	22
1	6	11	16	21

Reversing columns of the transpose

a)

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

ring 1 **ring 2**

b)

5	10	15	20	25
4	7	8	9	24
3	12	13	14	23
2	17	18	19	22
1	6	11	16	21

After rotating ring 1

c)

5	10	15	20	25
4	9	14	19	24
3	8	13	18	23
2	7	12	17	22
1	6	11	16	21

After rotating ring 2

Initial matrix

1	2	3	4	0	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	0	22
23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38

Solved matrix

0	0	0	0	0	0	0	0
8	9	10	11	0	13	0	15
0	0	0	0	0	0	0	0
23	24	25	26	0	28	0	30
31	32	33	34	0	36	0	38

Stack 1	Stack 2	Stack 3
5		
3	1	4
11	12	21
23	2	91

the array

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	12	23	91	21	11	3	1	4	5	0	0	0	0	0	0

Stack 1	Stack 2	Stack 3
5		
3	1	4
11	12	21
23	2	91

the array

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	3	11	23	0	0	12	2	1	0	0	0	0	91	21	4

L→ L<R ←R

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-5	-4	-2	-2	-1	1	2	3	3	4	5	6	7	12	15

K=10

Result: (-5, 15); (-2, 12); (3, 7); (4, 6)

(n + 1) / 2 n=7

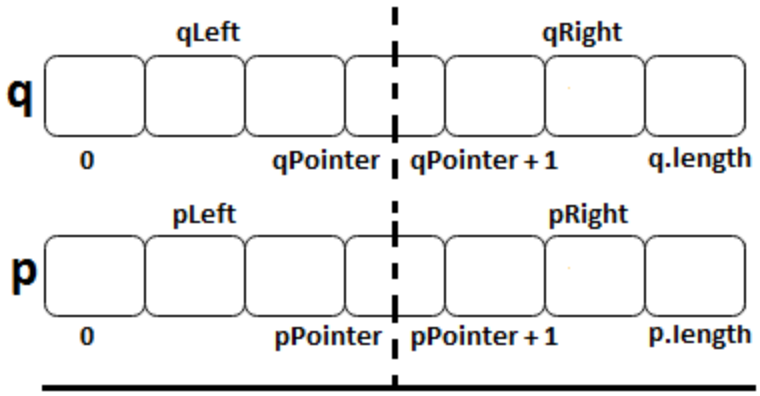
0	1	2	3	4	5	6
2	6	9	10	11	65	67

n is odd

(n/2 + (n/2 + 1)) / 2 n=6

0	1	2	3	4	5
2	6	9	10	11	65

n is even



qLeft.length + pLeft.length = qRight.length + pRight.length

↓

qPointer + pPointer = (q.length - qPointer) + (p.length - pPointer)

qPointerMin = 0 qPointer = (0 + 6) / 2 = 3 qPointerMax = 6

Q	2	6	9	10	11	65	67		
	0	1	2	3	4	5	6	7	8
P	1	5	17	18	25	28	39	77	88

pPointer = (7 + 9 + 1) / 2 - qPointer = 5

25>10

p[pPointer - 1] > q[qPointer] **qPointerMin = qPointer + 1**

qPointerMin = 4 qPointer = (4 + 6) / 2 = 5 qPointerMax = 6

Q	2	6	9	10	11	65	67		
	0	1	2	3	4	5	6	7	8
P	1	5	17	18	25	28	39	77	88

pPointer = (7 + 9 + 1) / 2 - qPointer = 3

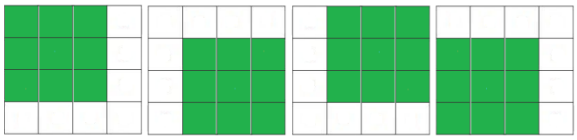
Median value

17.5

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	5	6	9	10	11	17	18	25	28	39	65	67	77	88

0	0	1	1	0	0	0
0	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	0	1	1	1
0	1	1	0	0	0	0

3x3



given matrix

1	1	0	1
0	1	1	1
1	1	1	1
0	1	1	1

subMatrix

		0	
	1	1	1
	1	2	2
	1	2	3

(i-1, j)

(i, j)

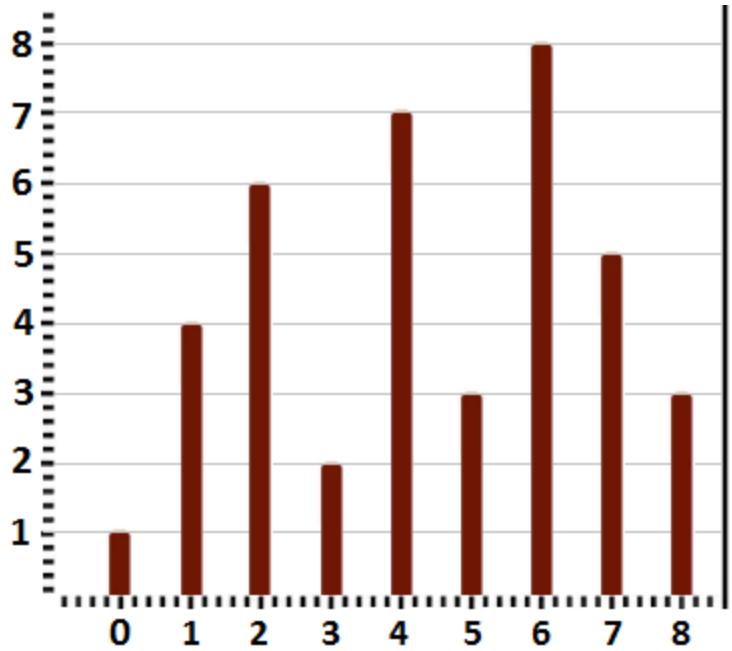
(i, j-1) (i-1, j-1)

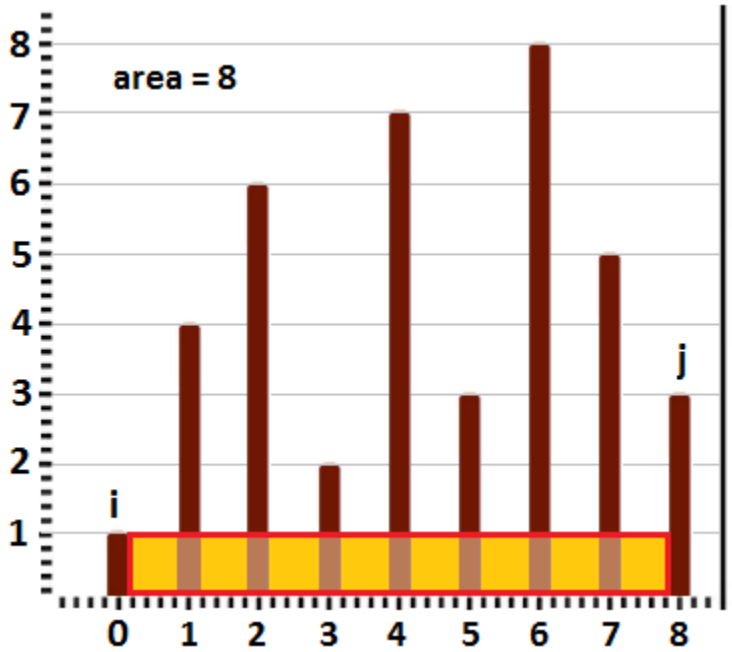
0	0	1	1	0	0	0
0	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	0	1	1	1
0	1	1	0	0	0	0

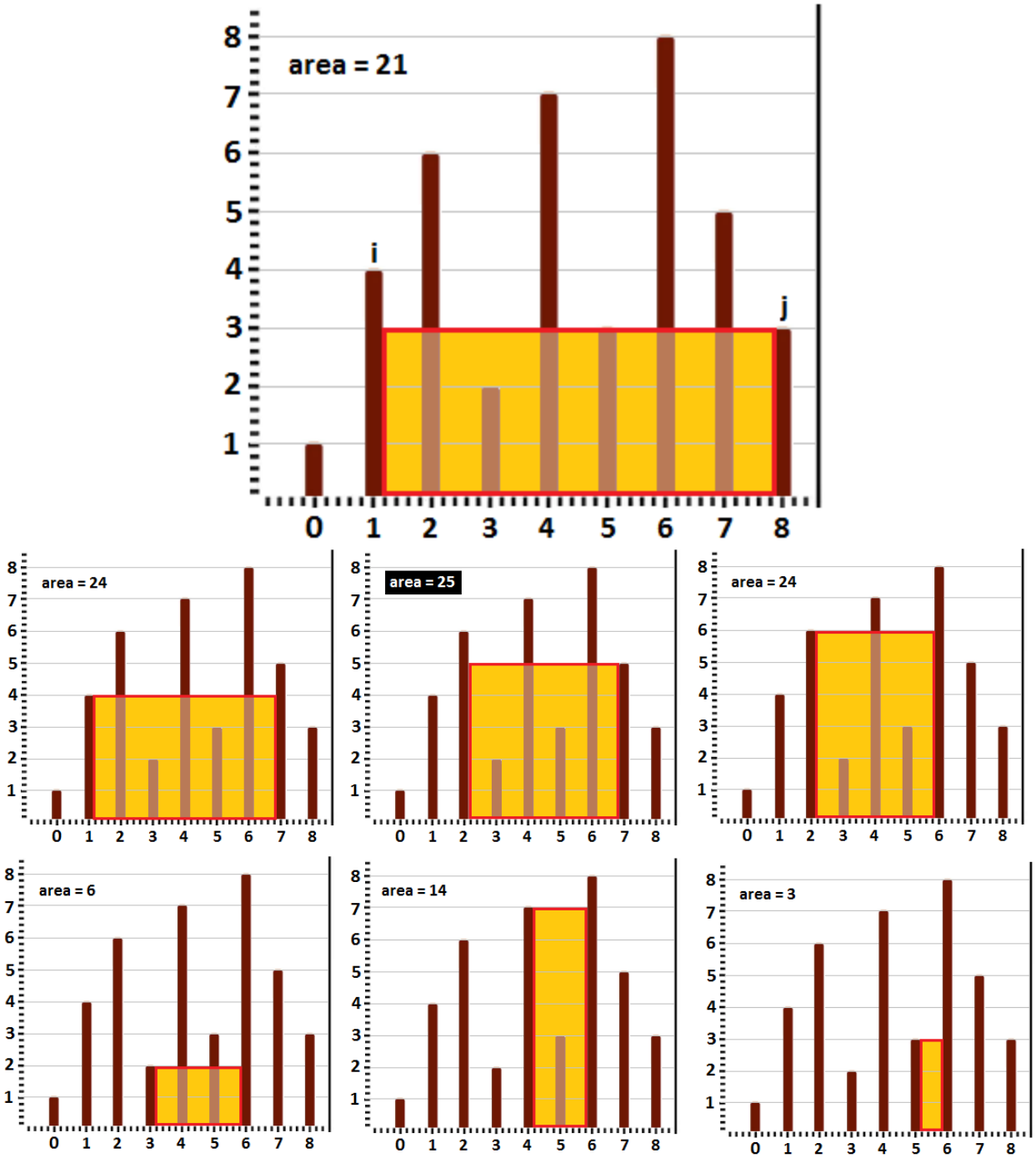
given matrix

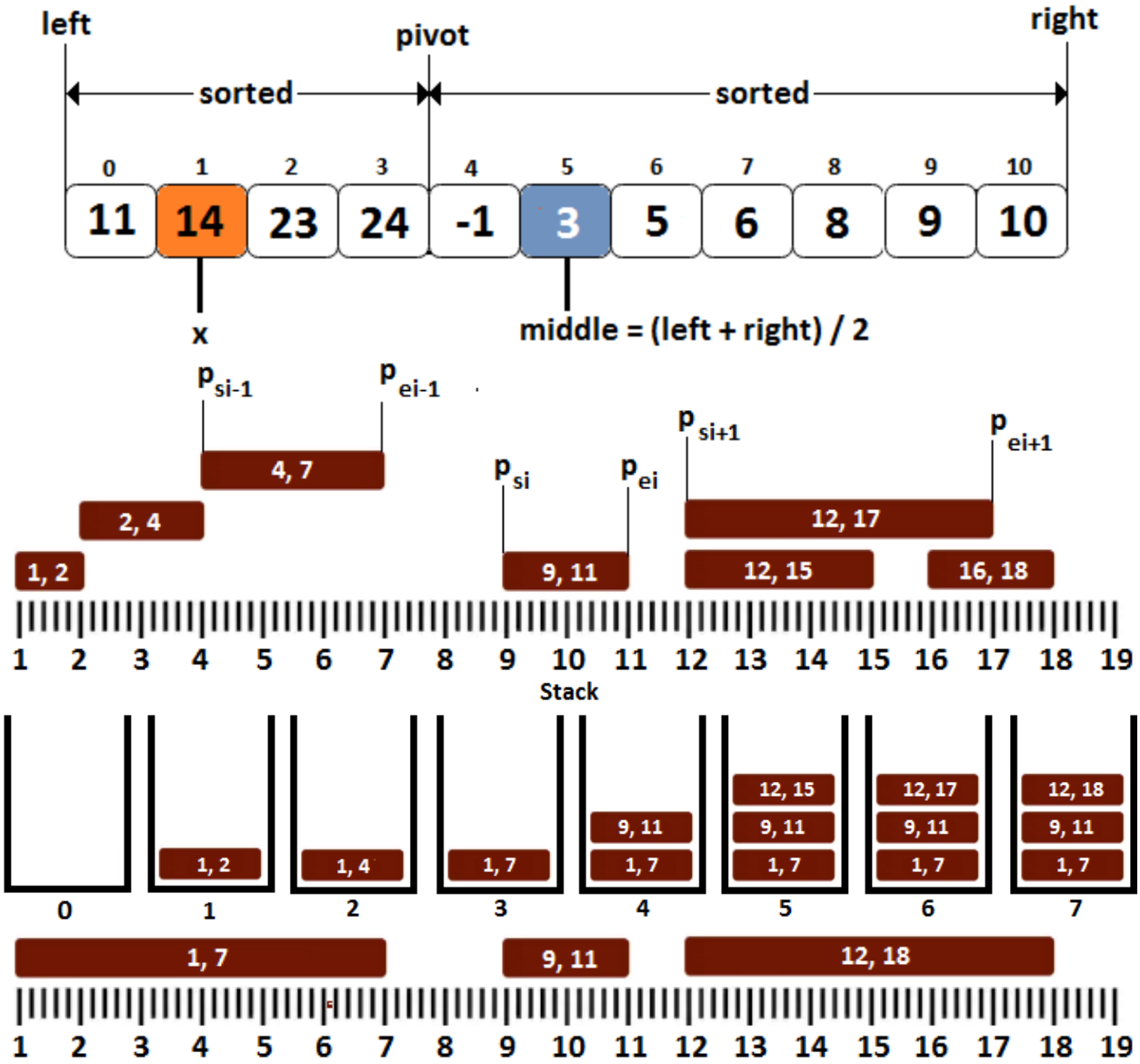
0	0	1	1	0	0	0
0	0	1	2	1	1	1
1	1	0	0	1	2	2
1	2	0	0	1	2	3
0	1	1	0	0	0	0

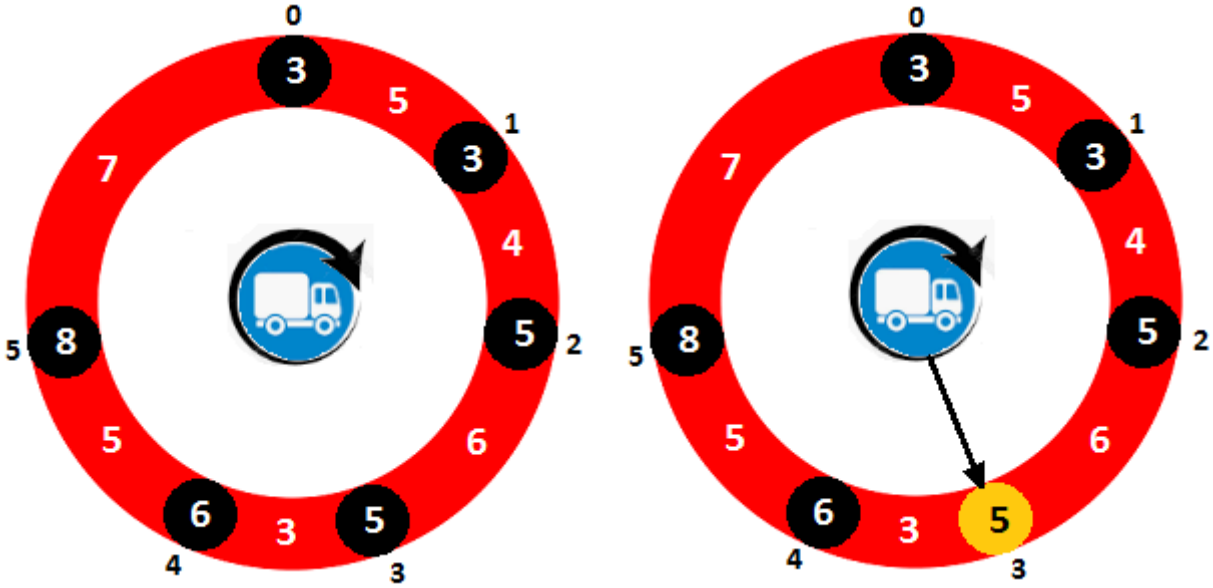
subMatrix

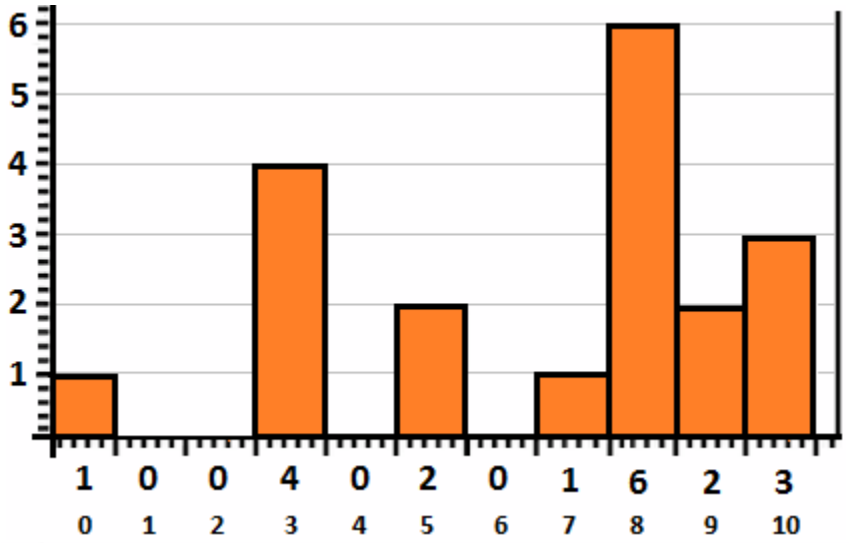

8
7
6
5
4
3
2
1
0 1 2 3 4 5 6 7 8

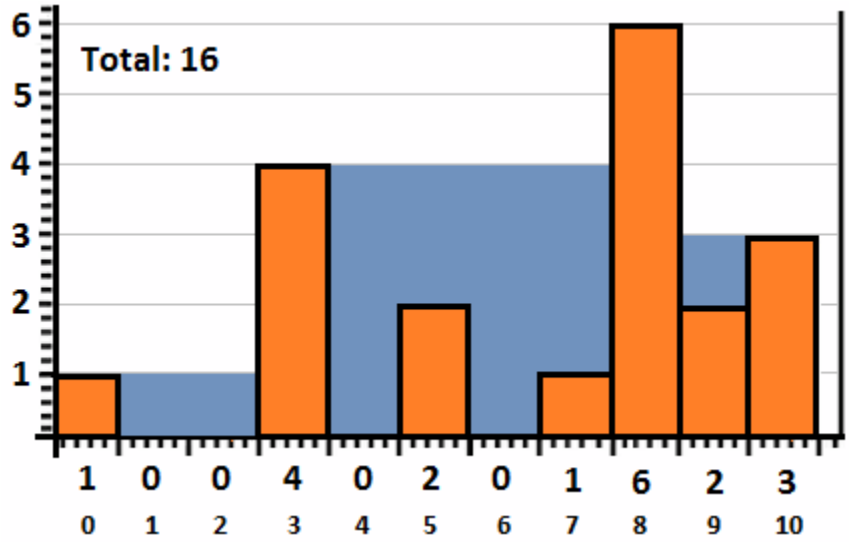

area = 8
i
j
8
7
6
5
4
3
2
1
0 1 2 3 4 5 6 7 8

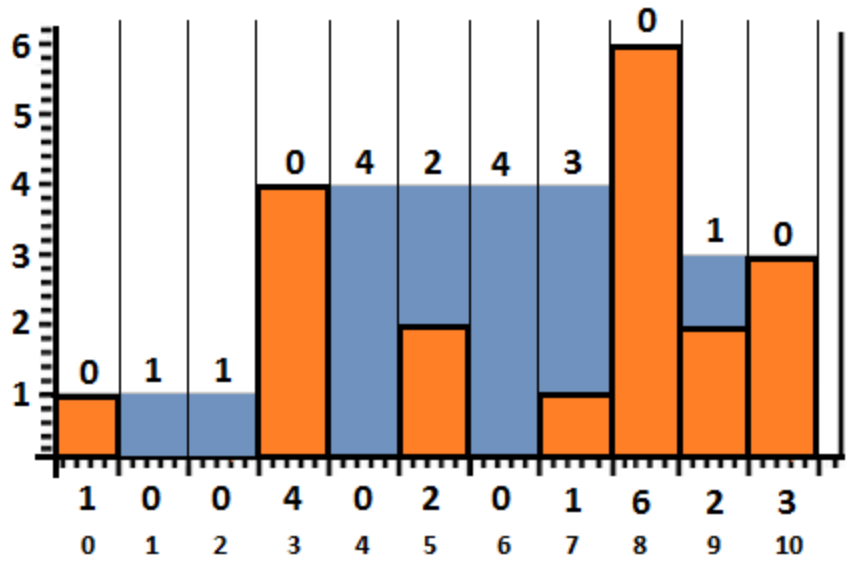

area = 21
i
j
area = 24
area = 25
area = 24
area = 6
area = 14
area = 3

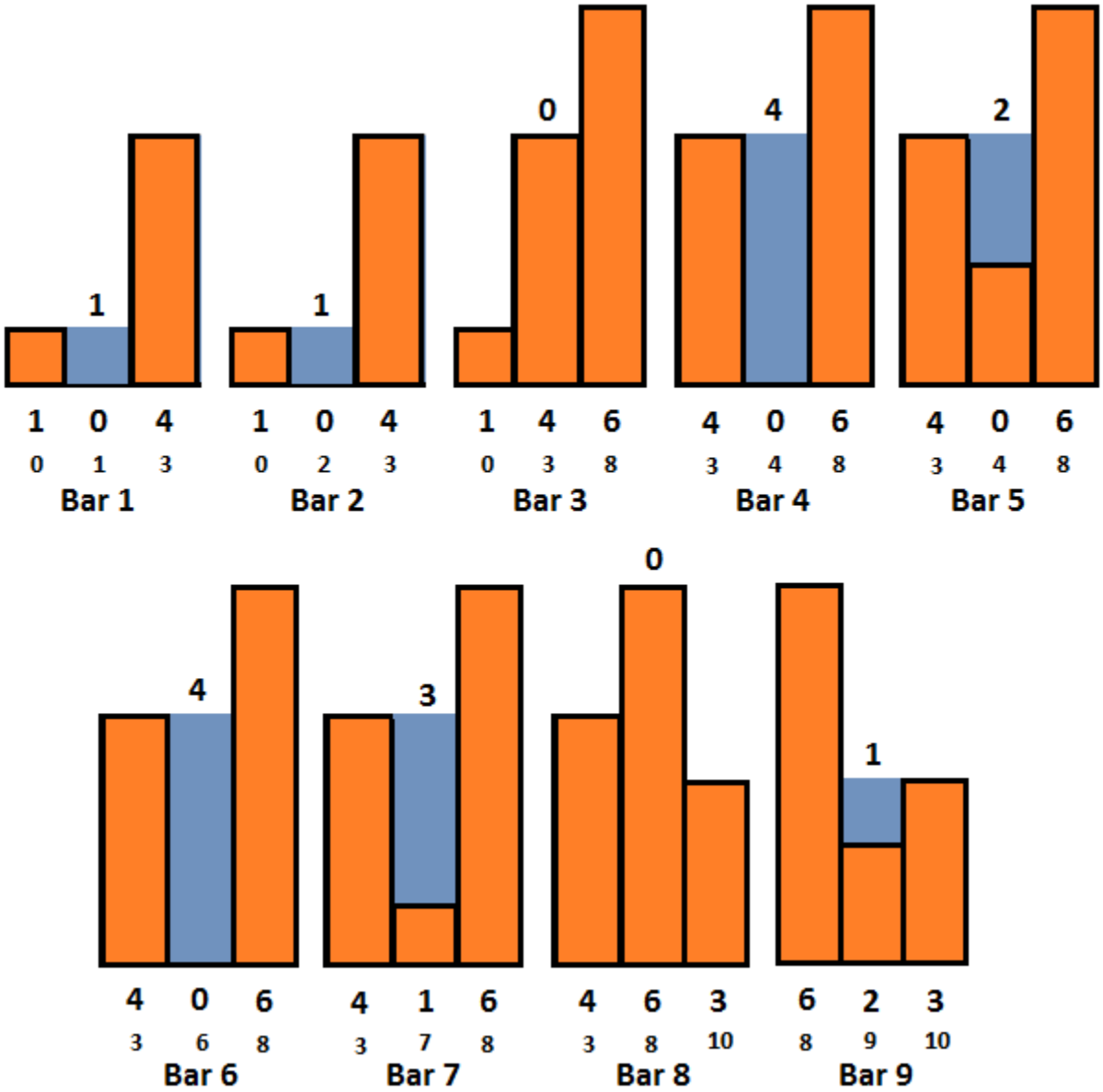

left
pivot
right
sorted
sorted
0 1 2 3 4 5 6 7 8 9 10
11 14 23 24 -1 3 5 6 8 9 10
x
middle = (left + right) / 2
p si-1
p ei-1
p si
p ei
p si+1
p ei+1
4, 7
2, 4
1, 2
9, 11
12, 17
12, 15
16, 18
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Stack
1, 2
1, 4
1, 7
9, 11
1, 7
12, 15
9, 11
1, 7
12, 17
9, 11
1, 7
12, 18
9, 11
1, 7
0 1 2 3 4 5 6 7
1, 7
9, 11
12, 18
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

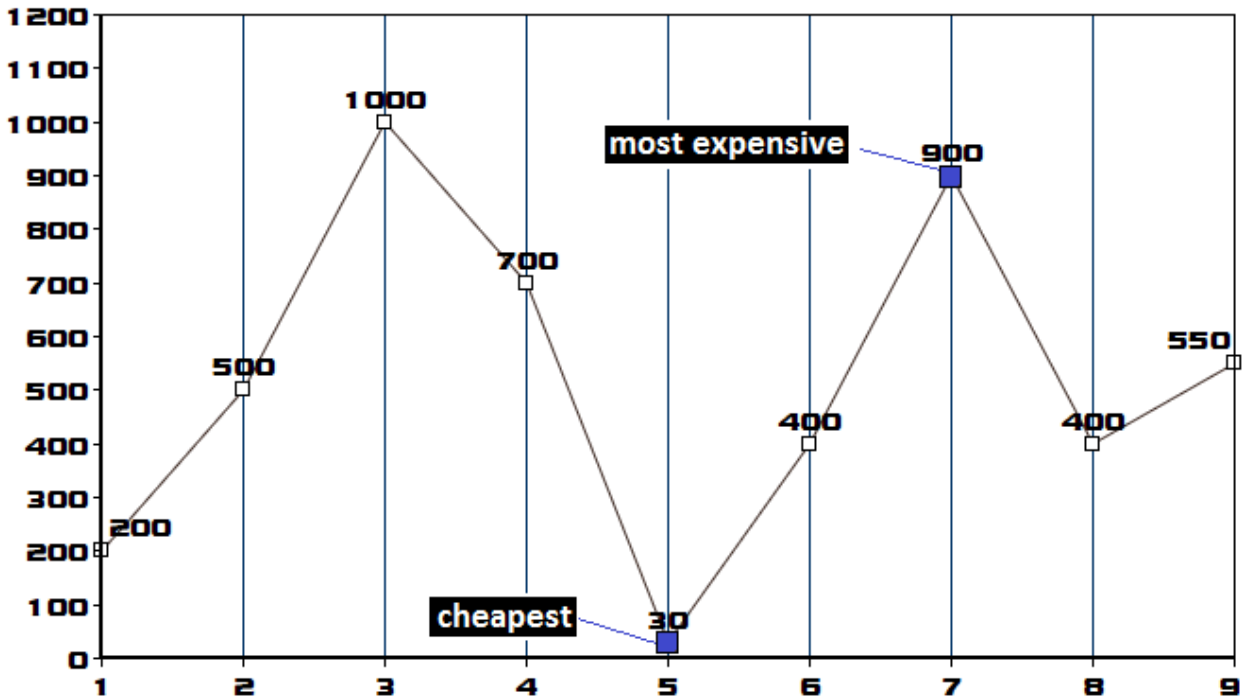

0
3
5
1
3
7
4
5
8
5
2
5
6
6
3
5
4
3
0
3
5
1
3
7
4
5
8
5
2
5
6
6
3
5
4
3


6
5
4
3
2
1
1 0 0 4 0 2 0 1 6 2 3
0 1 2 3 4 5 6 7 8 9 10


Total: 16
6
5
4
3
2
1
1 0 0 4 0 2 0 1 6 2 3
0 1 2 3 4 5 6 7 8 9 10

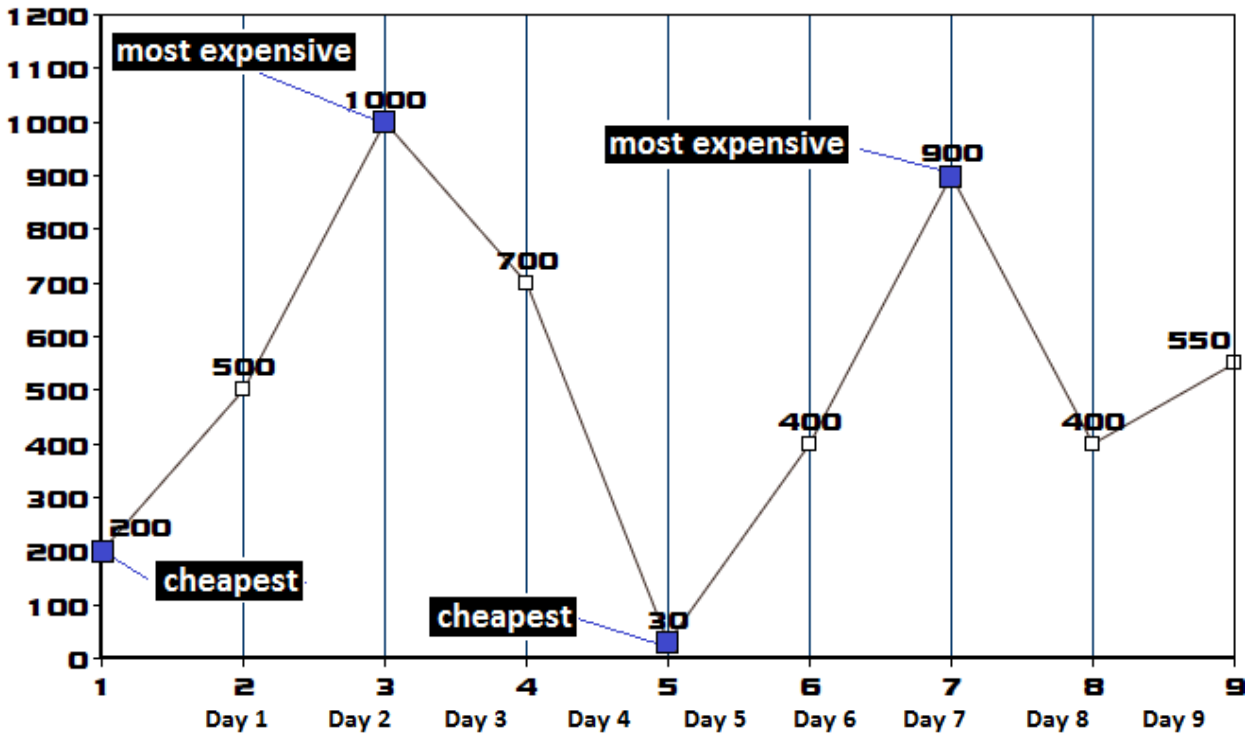

6
5
4
3
2
1
0
1
1
0
4
2
4
3
0
1
0
1 0 0 4 0 2 0 1 6 2 3
0 1 2 3 4 5 6 7 8 9 10


1
1 0 4
0 1 3
Bar 1
1
1 0 4
0 2 3
Bar 2
0
1 4 6
0 3 8
Bar 3
4
4 0 6
3 4 8
Bar 4
2
4 0 6
3 4 8
Bar 5
4
4 0 6
3 6 8
Bar 6
3
4 1 6
3 7 8
Bar 7
0
4 6 3
3 8 10
Bar 8
1
6 2 3
8 9 10
Bar 9



	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9
Price	200	500	1000	700	30	400	900	400	550
Min price (from day 1 till today)	200	200	200	200	30	30	30	30	30
Price - Min price	0	300	800	500	0	370	870	370	520
Max profit max(Max profit, Price - Min Price)	200	300	800	800	800	800	870	870	870

maximum profit

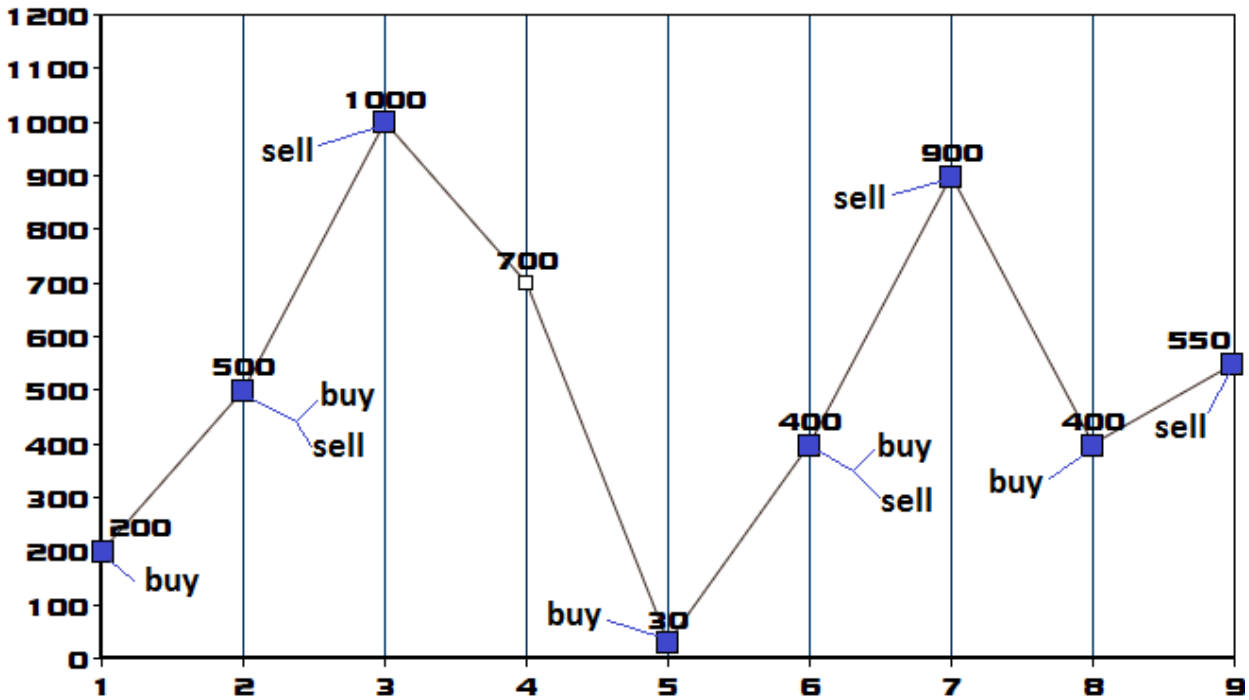


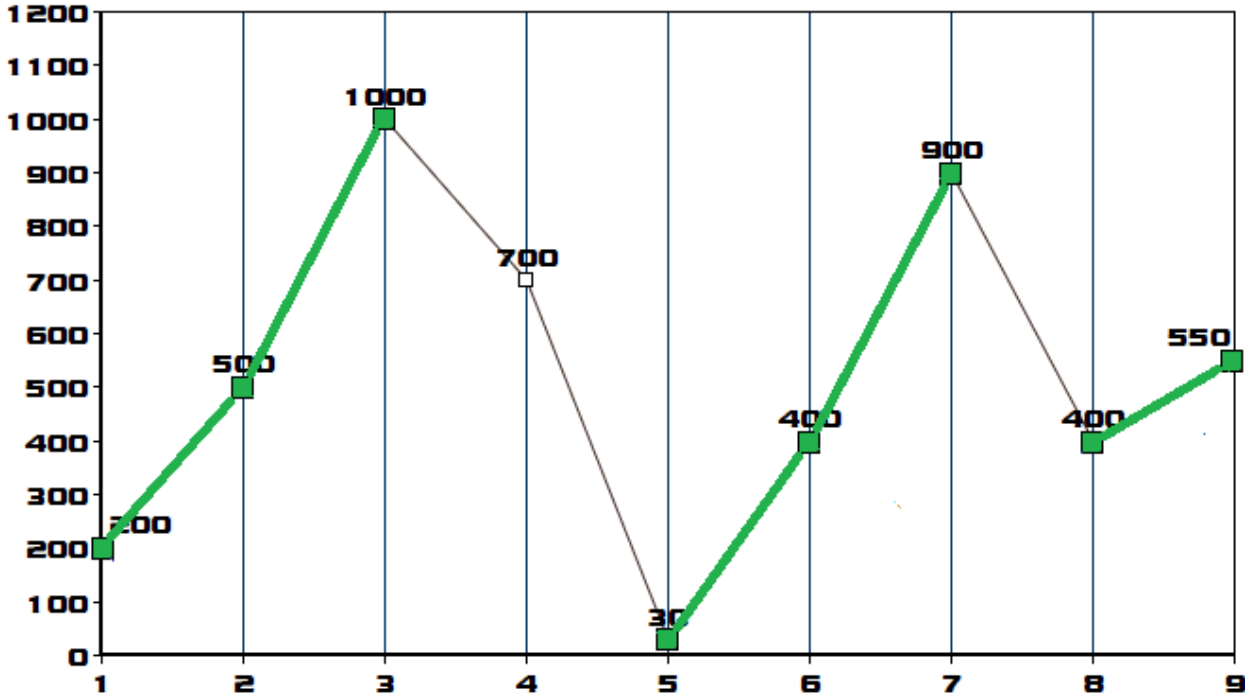
	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9
Price	200	500	1000	700	30	400	900	400	550
left[day] maximum profit till this day	0	300	800	800	800	800	870	870	870
	left[0]	left[1]	left[2]	left[3]	left[4]	left[5]	left[6]	left[7]	left[8]

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9
Price	200	500	1000	700	30	400	900	400	550
right[day] maximum profit after this day	870	870	870	870	870	500	150	150	0
	right[0]	right[1]	right[2]	right[3]	right[4]	right[5]	right[6]	right[7]	right[8]

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9
Price	200	500	1000	700	30	400	900	400	550
transaction 1, left[day]	0	300	800	800	800	800	870	870	870
transaction2, right[day]	870	870	870	870	870	500	150	150	0
max(left[day]+right[day])	870	1170	1670	1670	1670	1300	1020	1020	870

maximum profit


1200
1100
1000
900
800
700
600
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
200
buy
500
buy
sell
1000
sell
700
buy
30
400
buy
sell
900
sell
400
buy
550
sell


1200
1100
1000
900
800
700
600
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
200
500
1000
700
30
400
900
400
550

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9
Price	200	500	1000	700	30	400	900	400	550
Max profit	0	300	800	800	800	1170	1670	1670	1820

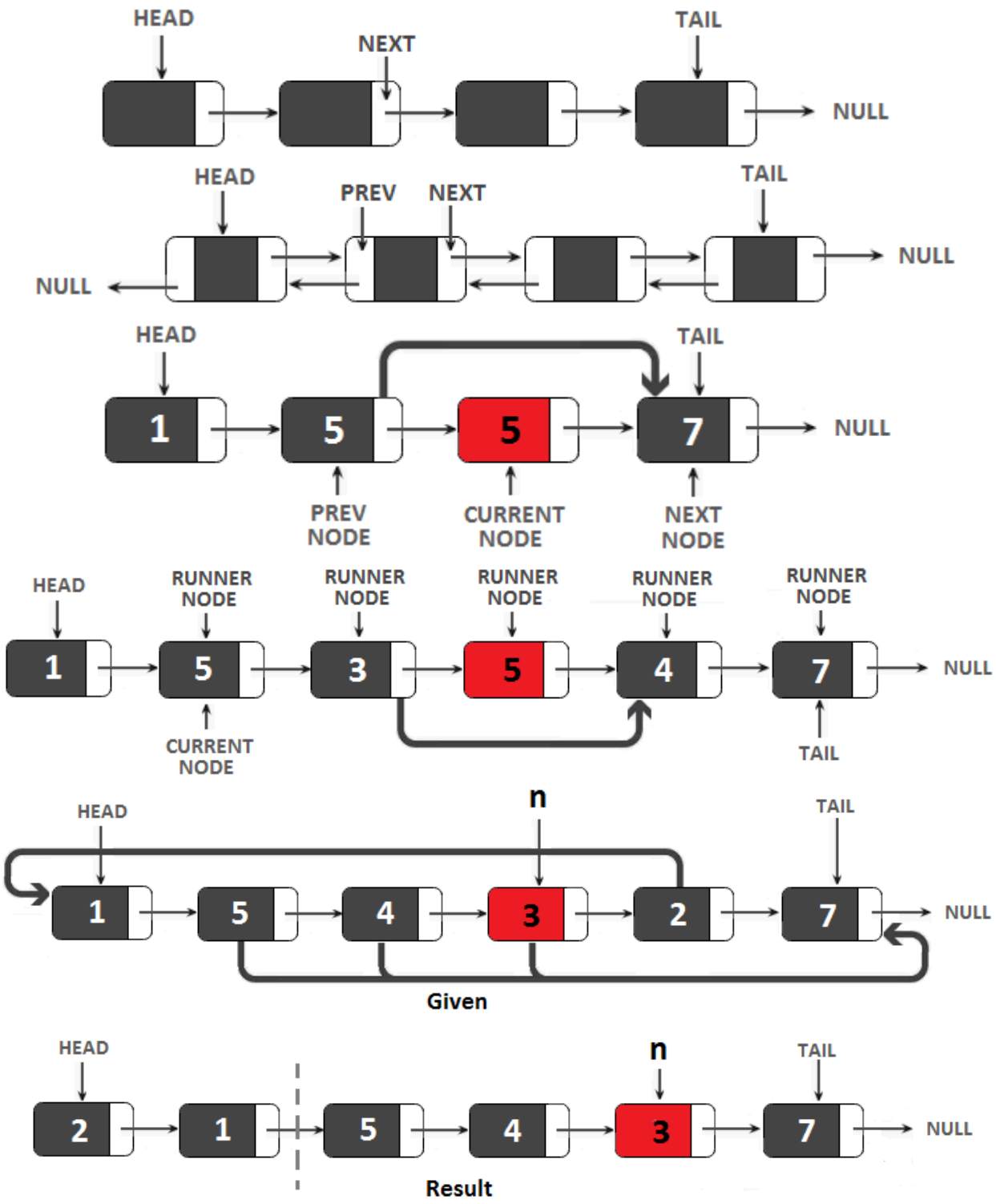
maximum profit

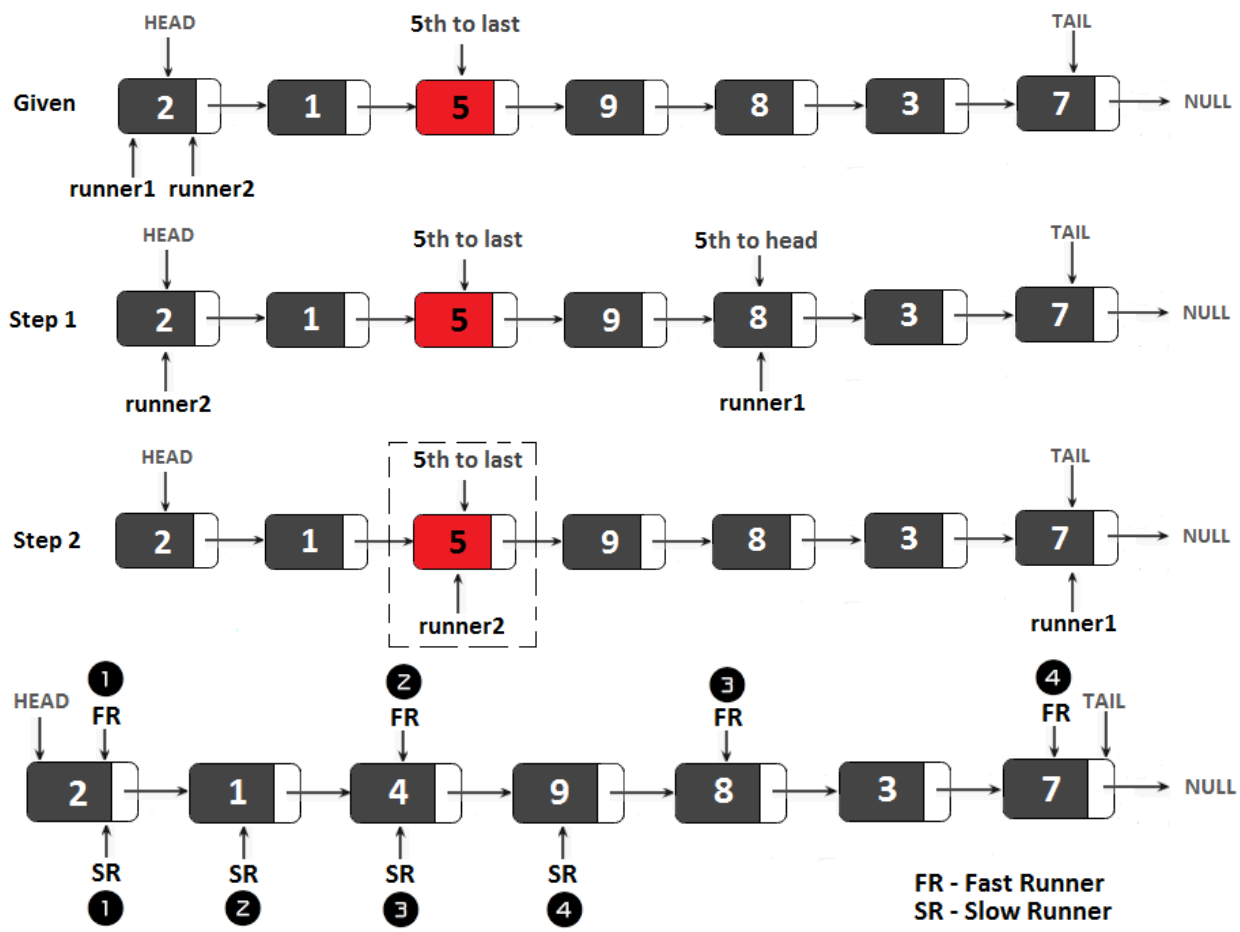
Given array:
[4, 2, 9, 5, 12, 6, 8] → Set:

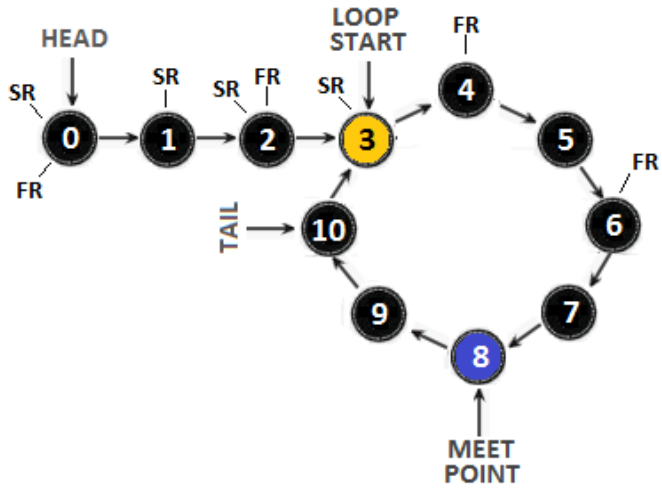
?11	?8	?7	?4	?5	?3	?1
12	9	8	5	6	4	2
		9			5	
					6	

Longest sequence has 3 elements

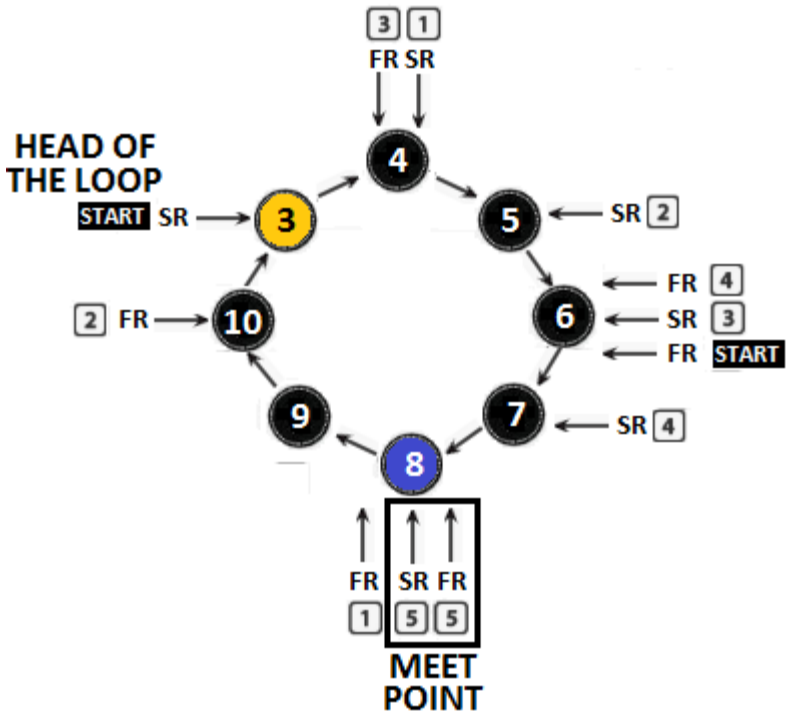
Chapter 11: Linked Lists and Maps

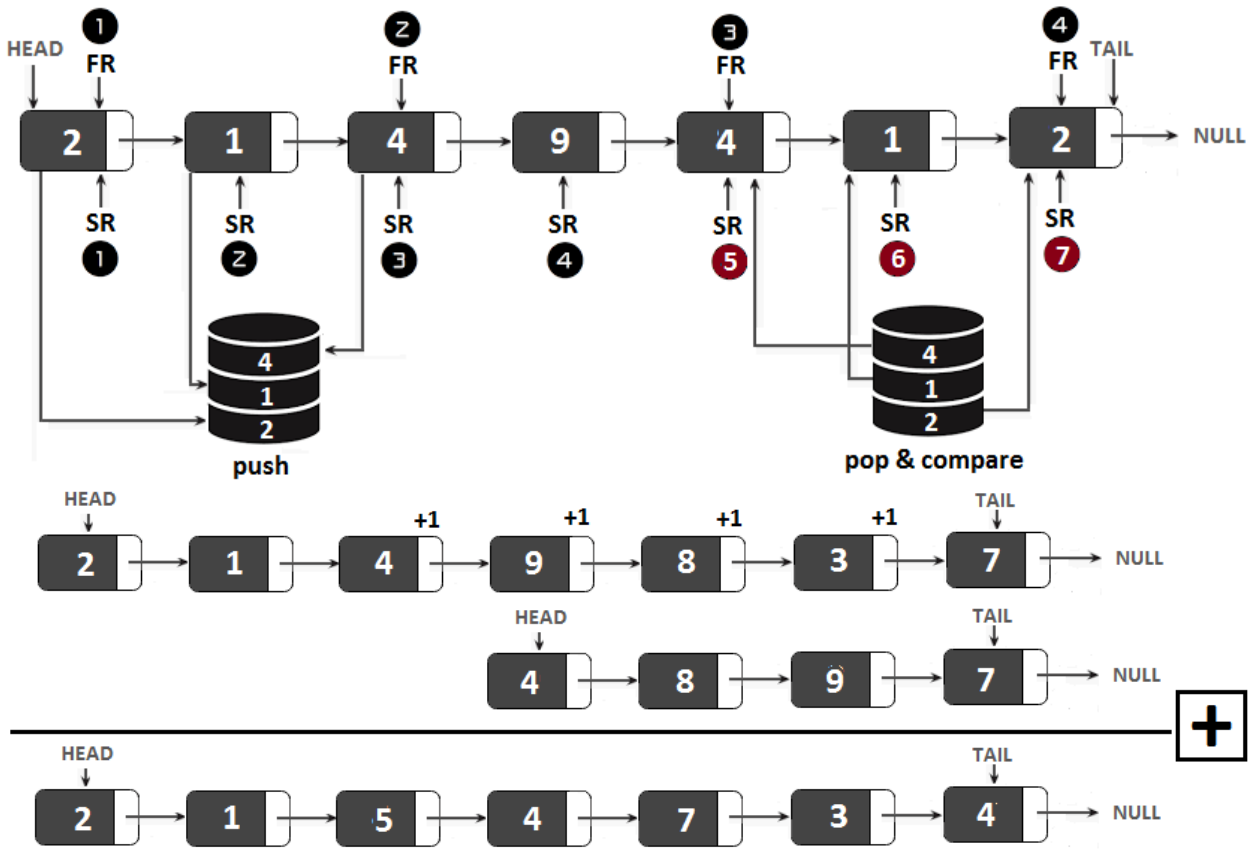


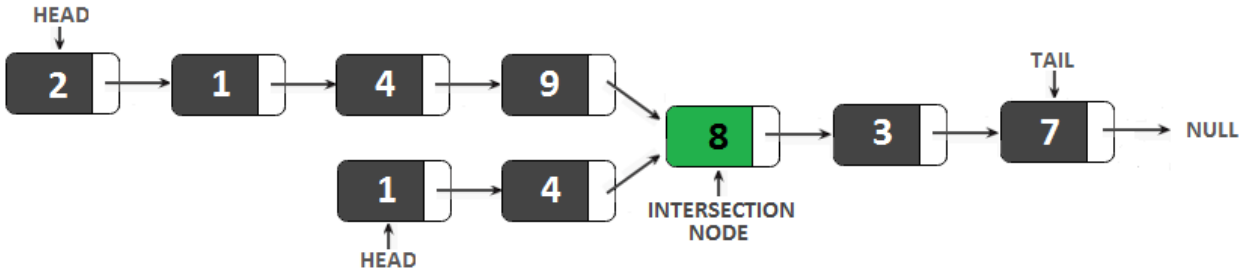


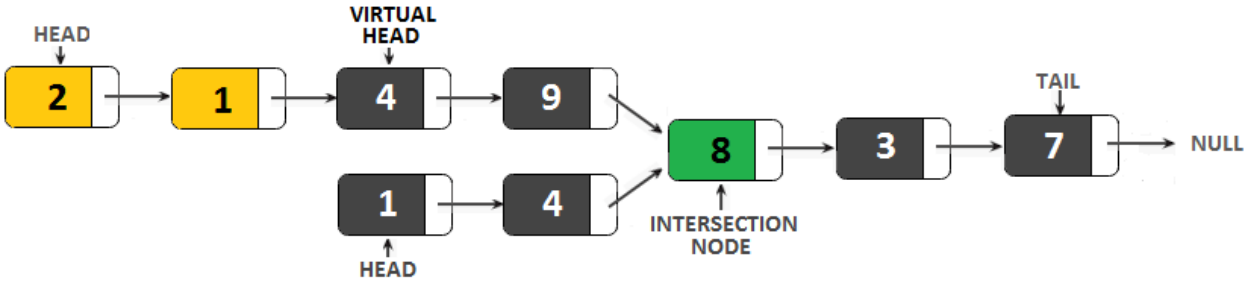


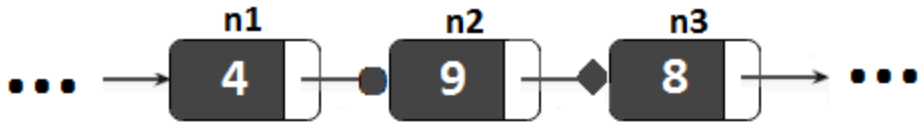
FR is three nodes away from the start of the loop when SR is at the start of the loop.
So it is three nodes from the start of the list.

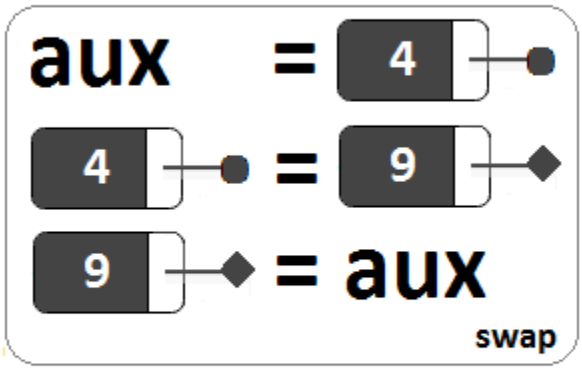

HEAD OF THE LOOP
START SR
3
4
5
6
7
8
9
10
3 1
FR SR
SR 2
FR 4
SR 3
FR START
SR 4
2 FR
FR SR FR
1 5 5
MEET POINT

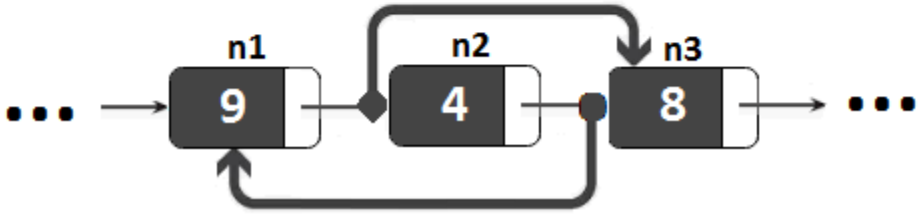

HEAD
TAIL
NULL
FR
SR
push
pop & compare
+1
+1
+1
+1
+

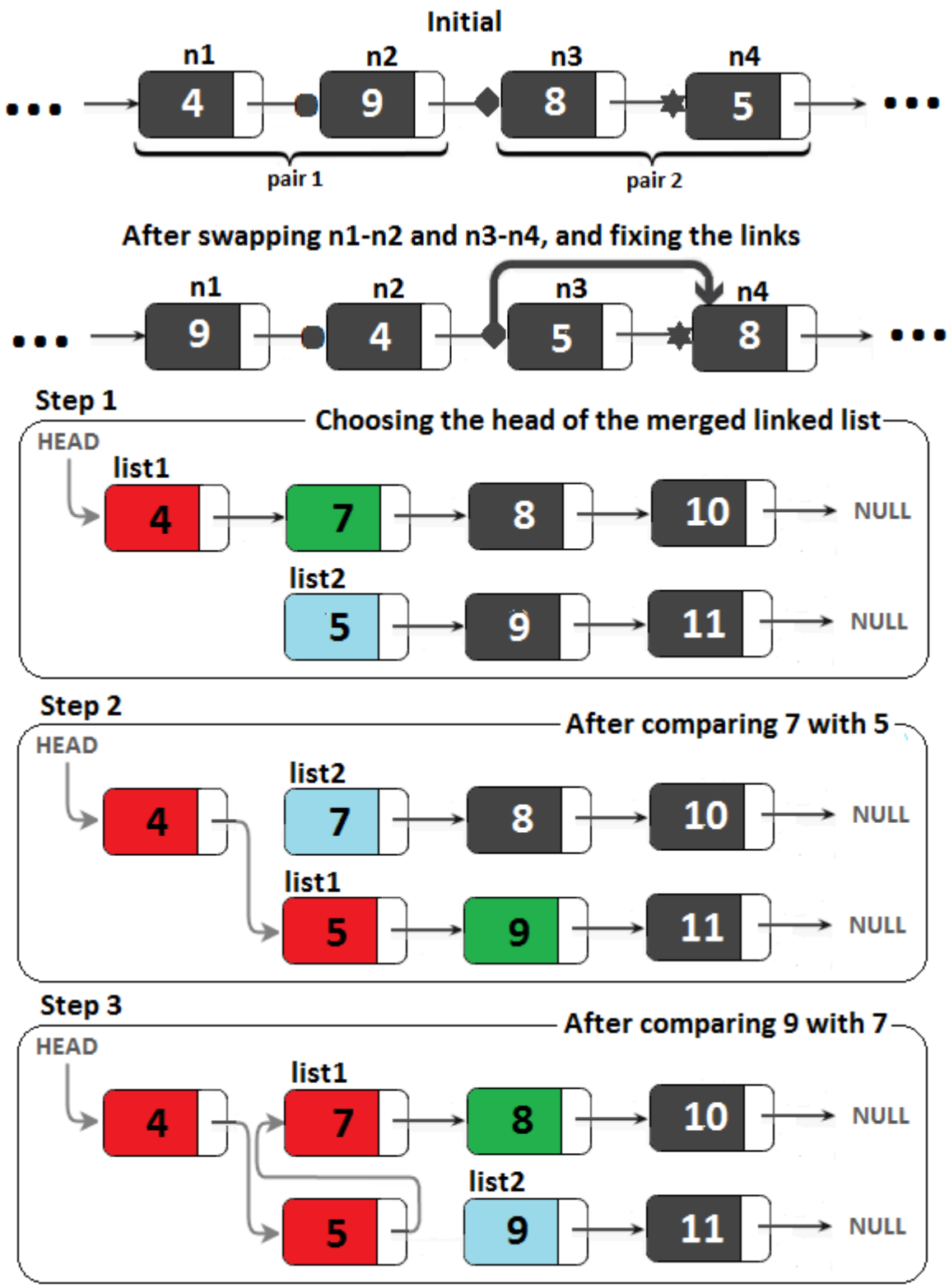

HEAD
2
1
4
9
8
3
7
TAIL
NULL
1
4
HEAD
INTERSECTION NODE

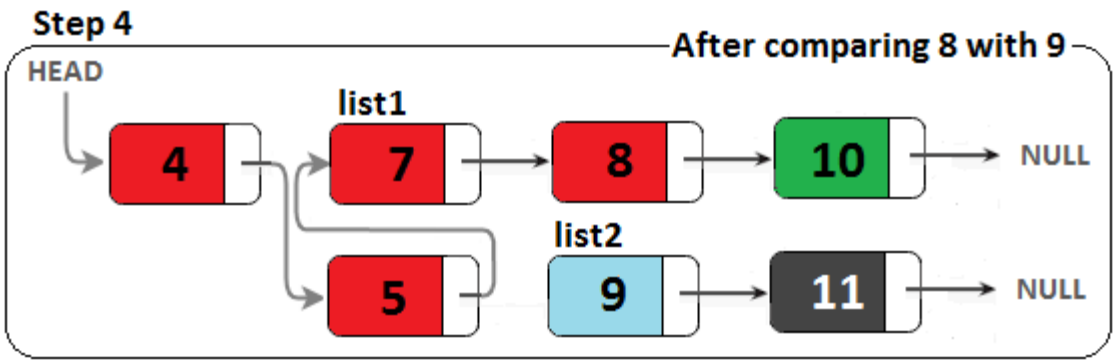

HEAD
VIRTUAL HEAD
2
1
4
9
8
3
7
TAIL
NULL
1
4
HEAD
INTERSECTION NODE

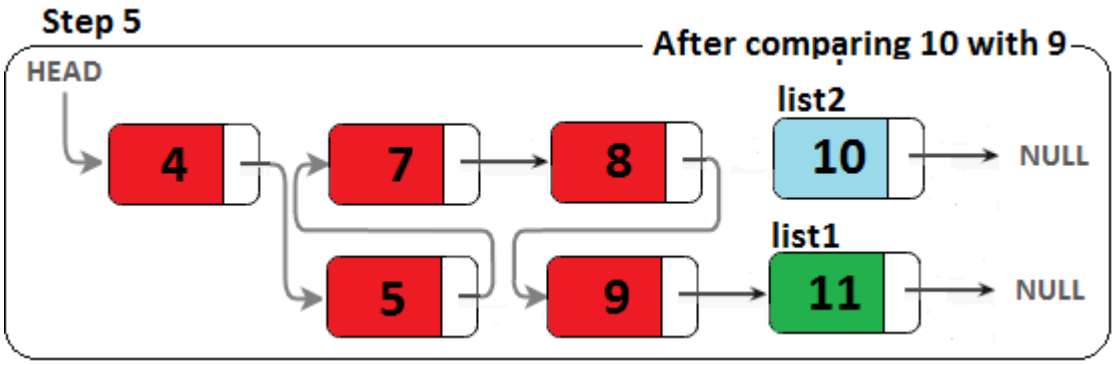

n1
n2
n3
4
9
8

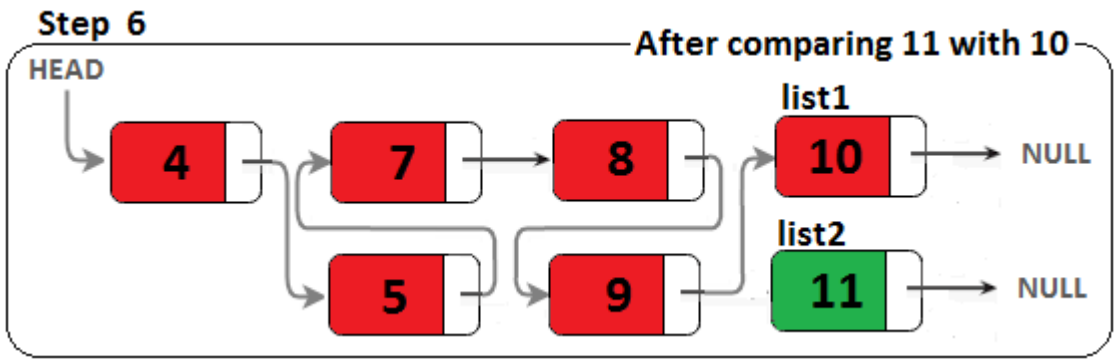

aux = 4
4 = 9
9 = aux
swap

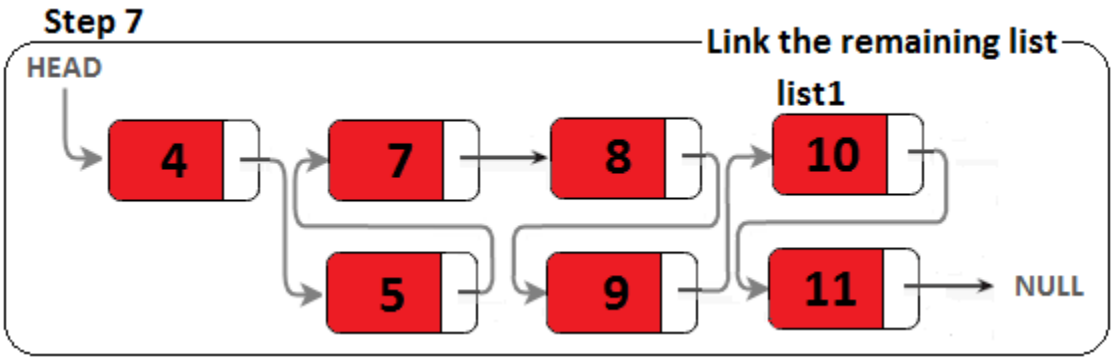

n1
n2
n3
9
4
8

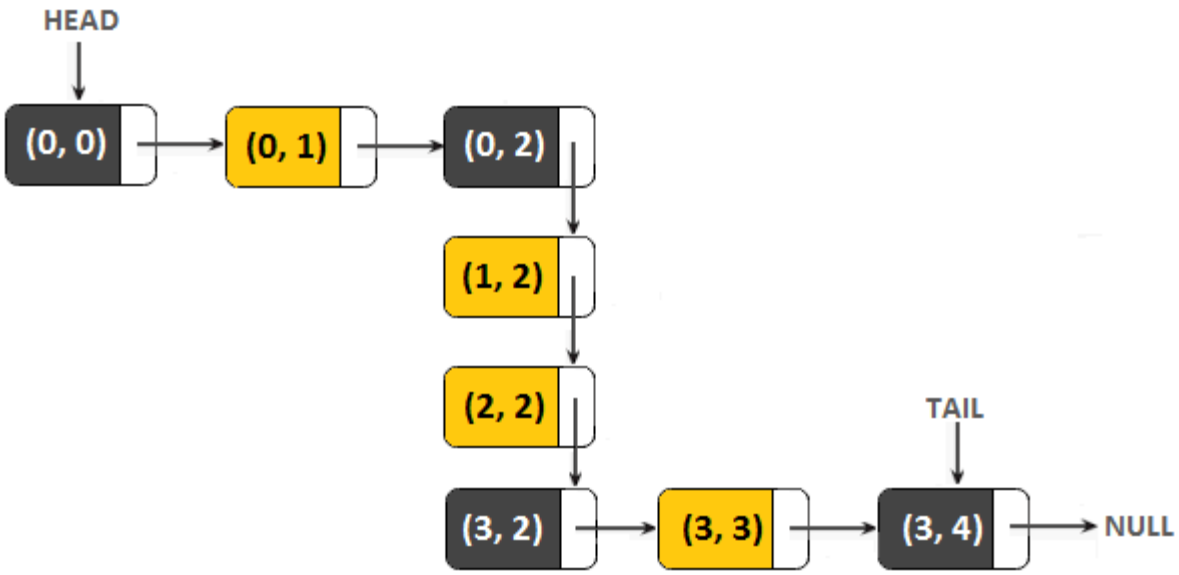

Initial
n1
n2
n3
n4
4
9
8
5
pair 1
pair 2
After swapping n1-n2 and n3-n4, and fixing the links
n1
n2
n3
n4
9
4
5
8
Step 1
Choosing the head of the merged linked list
HEAD
list1
4
7
8
10
NULL
list2
5
9
11
NULL
Step 2
After comparing 7 with 5
HEAD
list2
4
7
8
10
NULL
list1
5
9
11
NULL
Step 3
After comparing 9 with 7
HEAD
list1
4
7
8
10
NULL
list2
5
9
11
NULL

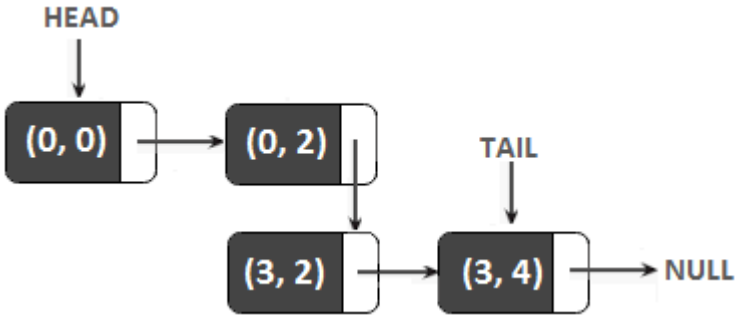

Step 4
After comparing 8 with 9
HEAD
list1
4
7
8
10
NULL
list2
5
9
11
NULL

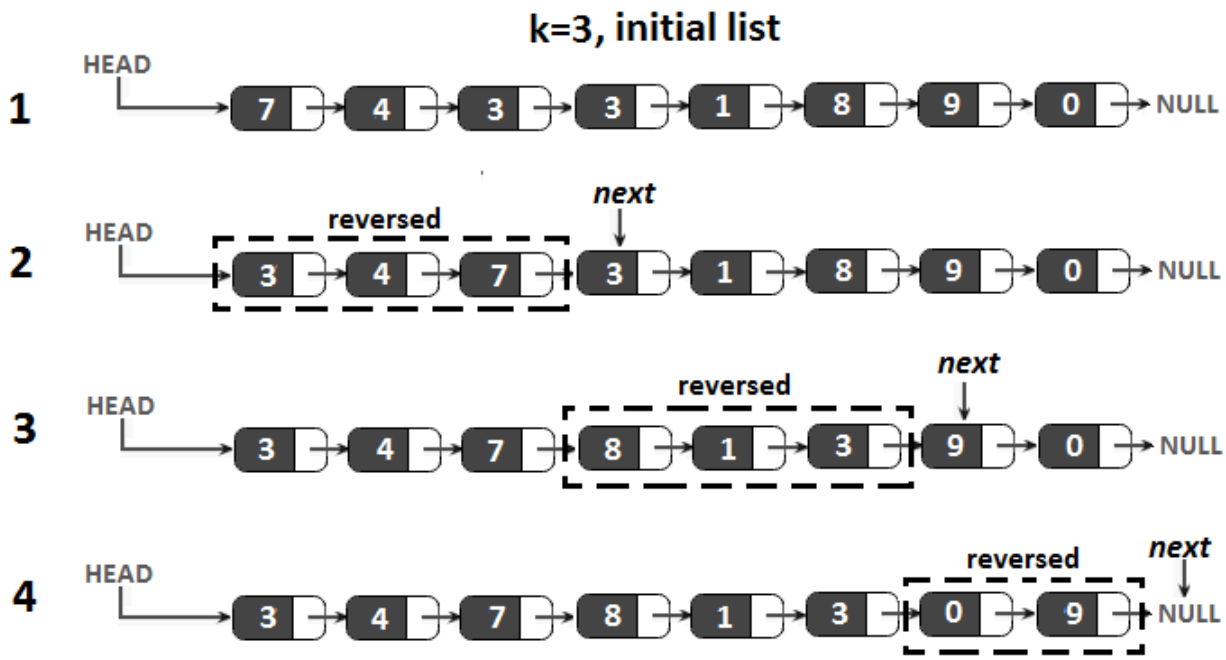

Step 5
After comparing 10 with 9
HEAD
list2
4
7
8
10
NULL
list1
5
9
11
NULL

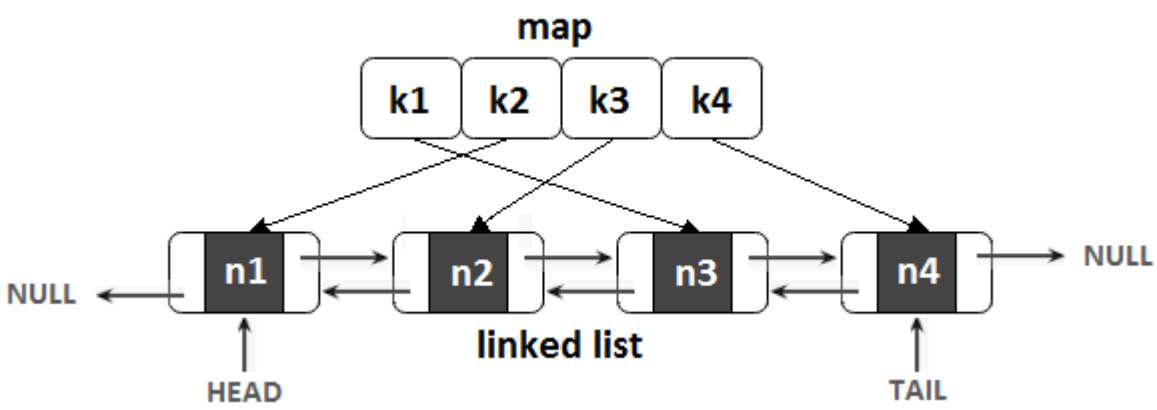

Step 6
After comparing 11 with 10
HEAD
list1
4
7
8
10
NULL
list2
5
9
11
NULL


Step 7
Link the remaining list
HEAD
list1
4
7
8
10
5
9
11
NULL

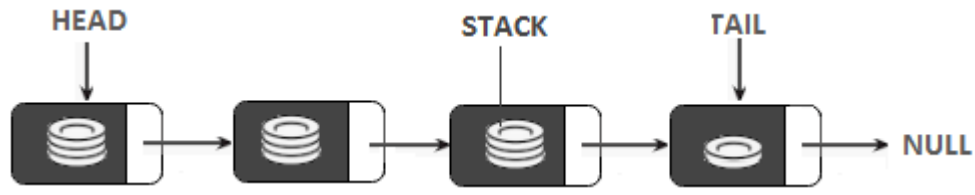

HEAD
(0, 0)
(0, 1)
(0, 2)
(1, 2)
(2, 2)
TAIL
(3, 2)
(3, 3)
(3, 4)
NULL

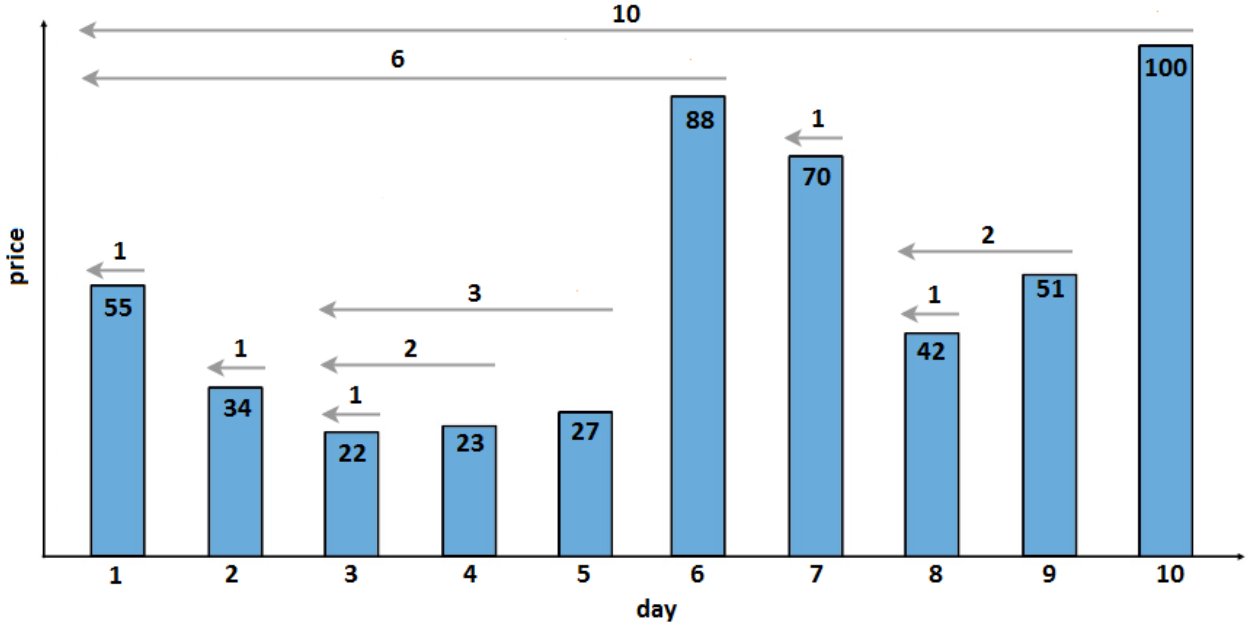

HEAD
(0, 0)
(0, 2)
TAIL
(3, 2)
(3, 4)
NULL

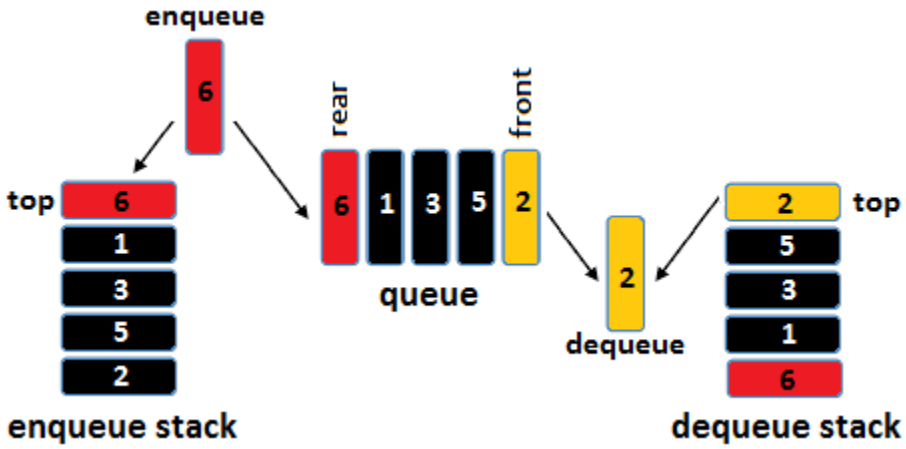

k=3, initial list
1
HEAD
7
4
3
3
1
8
9
0
NULL
2
HEAD
reversed
next
3
4
7
3
1
8
9
0
NULL
3
HEAD
reversed
next
3
4
7
8
1
3
9
0
NULL
4
HEAD
reversed
next
3
4
7
8
1
3
0
9
NULL

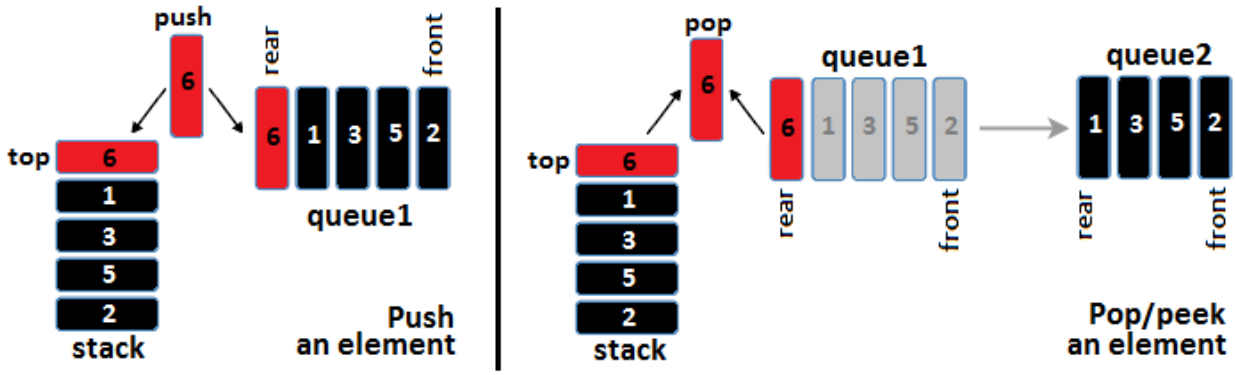

map
k1
k2
k3
k4
n1
n2
n3
n4
NULL
NULL
linked list
HEAD
TAIL

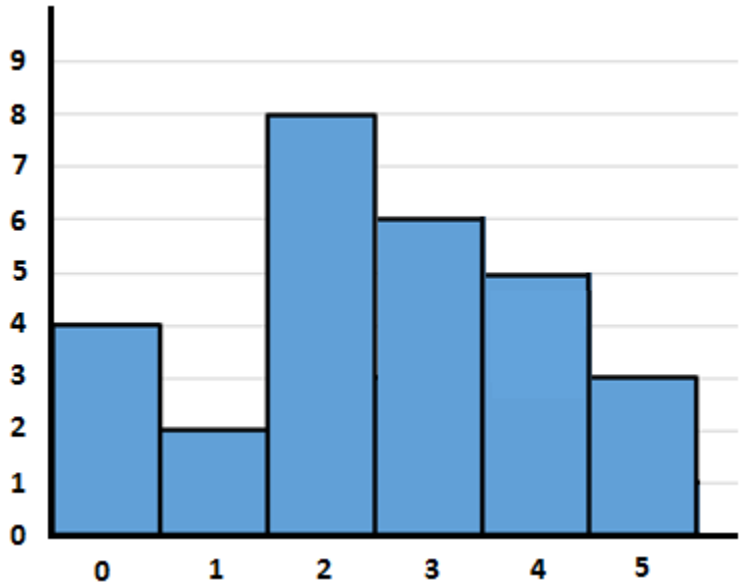
Chapter 12: Stacks and Queues

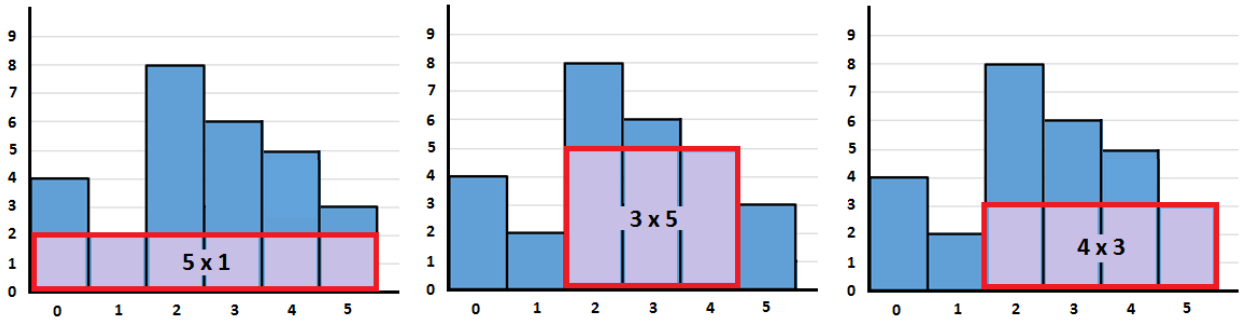


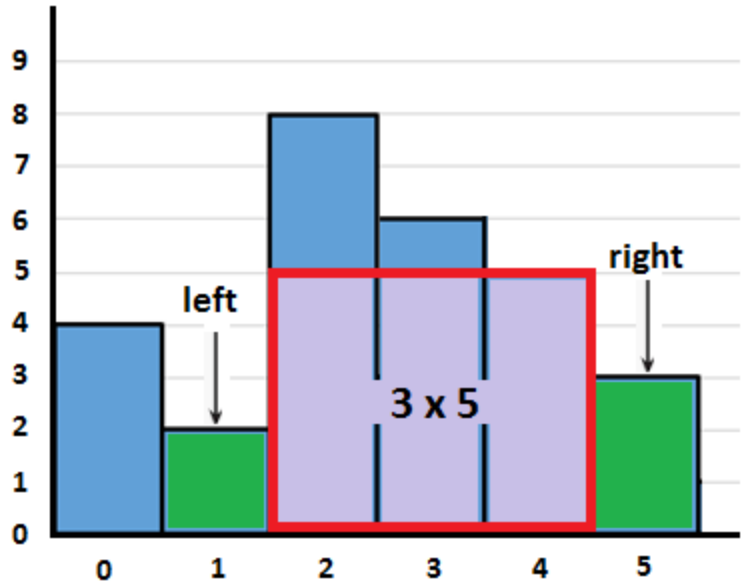





push
6
top
6
1
3
5
2
stack
rear
front
6
1
3
5
2
queue1
Push
an element
pop
6
top
6
1
3
5
2
stack
queue1
6
1
3
5
2
rear
front
queue2
1
3
5
2
rear
front
Pop/peek
an element


9
8
7
6
5
4
3
2
1
0
0
1
2
3
4
5


5 x 1
3 x 5
4 x 3



1	1	1				1	1		3
	1	1		1		1			
1	1	1	1			1		2	
1			1		1				
1	1	1	1	1	1				4
			1			1	1		4
					1	1			
5	5		6				1	1	
5				6	6		1		
5	5		6		6		1	1	1

water

land

```
1, 1, 1, 0, 0, 0, 1, 1, 0, 1
0, 1, 1, 0, 1, 0, 1, 0, 0, 0
1, 1, 1, 1, 0, 0, 1, 0, 1, 0
1, 0, 0, 1, 0, 1, 0, 0, 0, 0
1, 1, 1, 1, 1, 1, 0, 0, 0, 1
0, 0, 0, 1, 0, 0, 1, 1, 0, 1
0, 0, 0, 0, 0, 1, 1, 0, 0, 0
1, 1, 0, 1, 0, 0, 0, 1, 1, 0
1, 0, 0, 0, 1, 1, 0, 1, 0, 0
1, 1, 0, 1, 0, 1, 0, 1, 1, 1
```

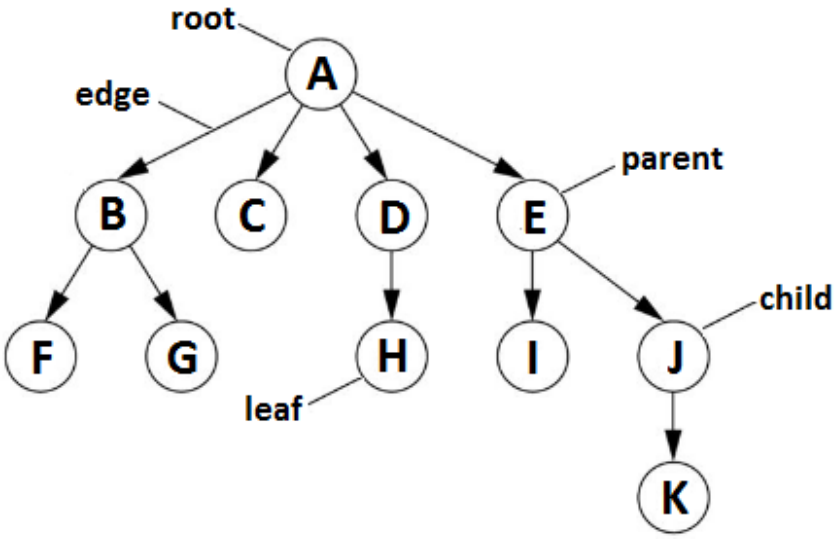
	0	1	2	3	4	5	6	7	8	9
0	0	1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
2	1	1	0	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	0	1	1	0	1
5	1	1	1	1	1	1	1	1	1	1
6	1	0	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	0
8	1	1	1	1	1	0	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1

	0	1	2	3	4	5	6	7	8	9
0	0	0	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1
2	1	0	0	0	1	1	1	1	1	1
3	1	0	0	0	0	0	0	0	0	0
4	1^{0}	1^{1}	1^{2}	1^{3}	0	0	0	0	0	0
5	0	0	0	1^{4}	0	0	0	0	0	0
6	0	0	0	1^{5}	1^{6}	1^{7}	1^{8}	1^{9}	0	0
7	0	0	0	1	0	0	0	1^{10}	0	0
8	1	1	1	1	0	0	0	1^{11}	0	0
9	1	1	1	1	0	0	0	1^{12}	1^{13}	1^{14}

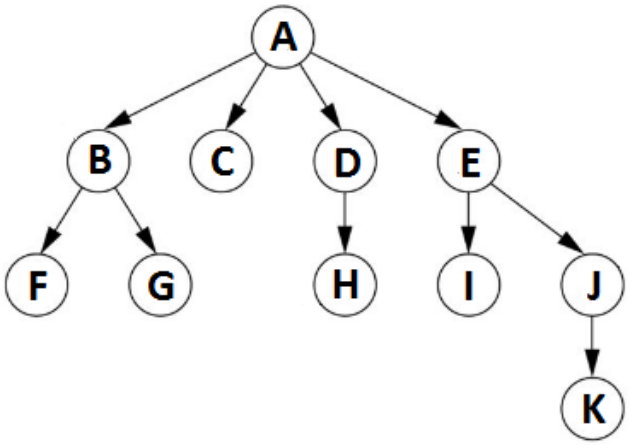
Shortest path: 15

Infix	Postfix	Prefix
(a + b) * c	a b + c *	* + a b c
a + (b * c)	a b c * +	+ a * b c

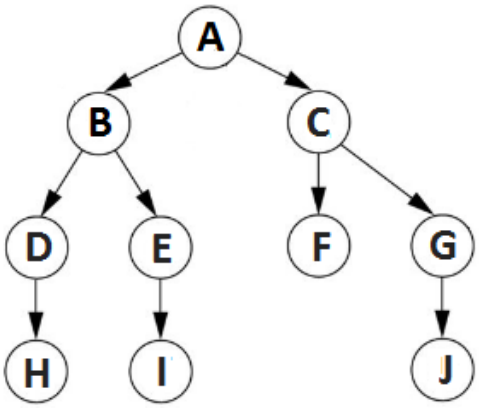
Chapter 13: Trees and Graphs



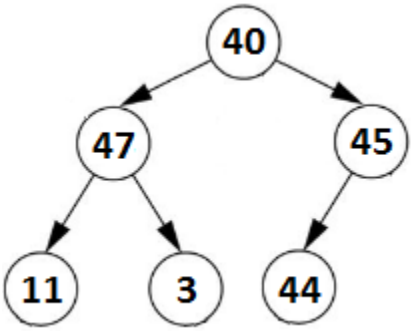
Node	Height	Depth
A	3	0
B	1	1
C	0	1
D	1	1
E	2	1
F	0	2
G	0	2
H	0	2
I	0	2
J	1	2
K	0	3



non-binary tree



binary tree



BFS

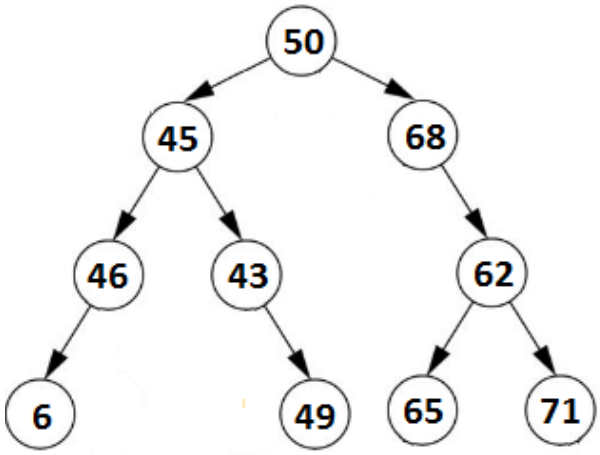
Level-Order: 40 47 45 11 3 44

DFS

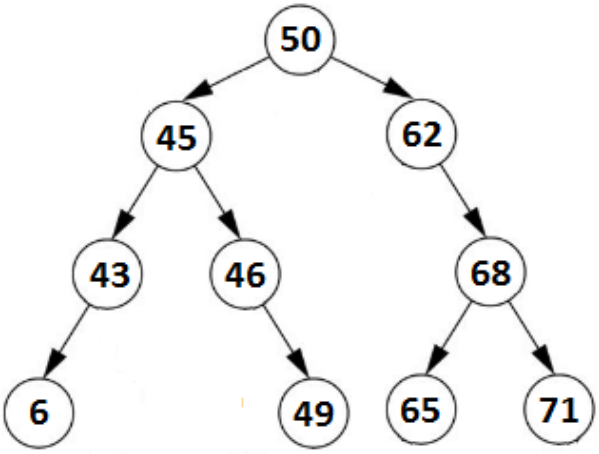
Pre-Order: 40 47 11 3 45 44

In-Order: 11 47 3 40 44 45

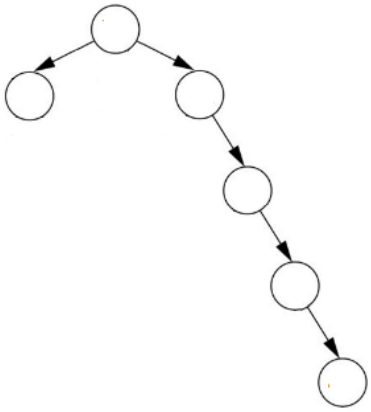
Post-Order: 11 3 47 44 45 40



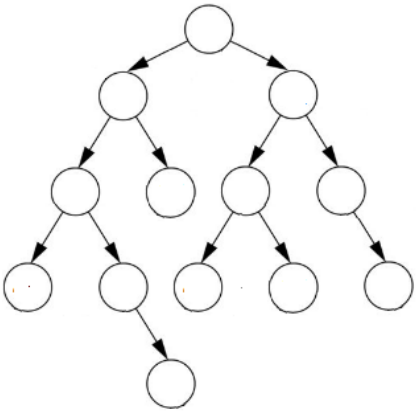
binary tree



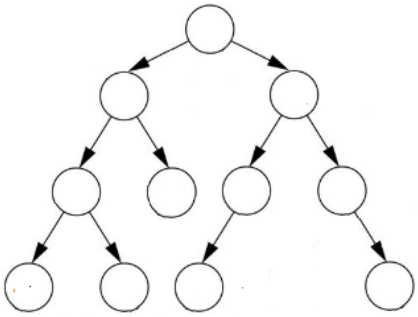
binary search tree



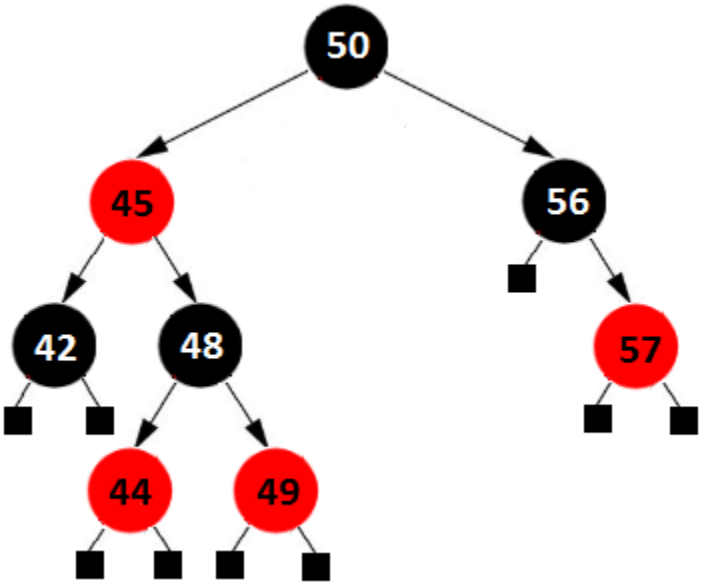
unbalanced
binary tree

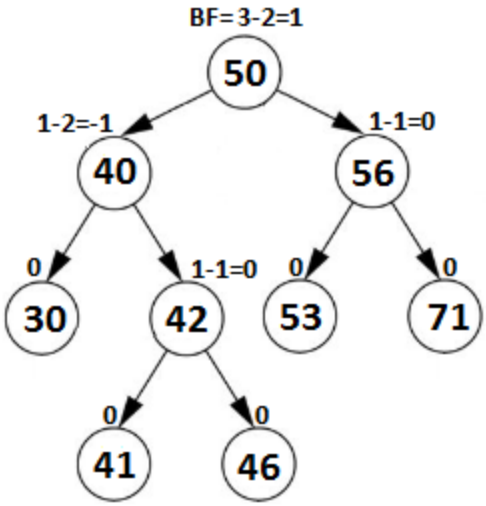


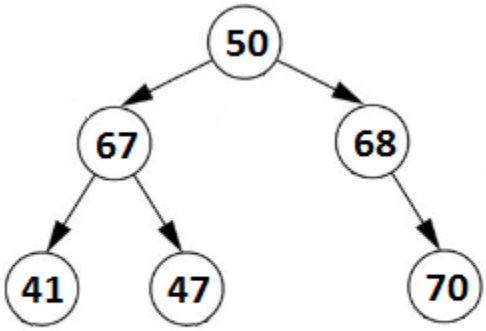
balanced
binary tree



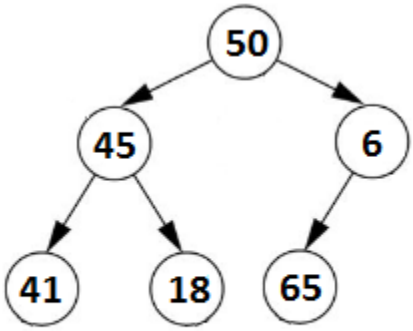
height
balanced
binary tree



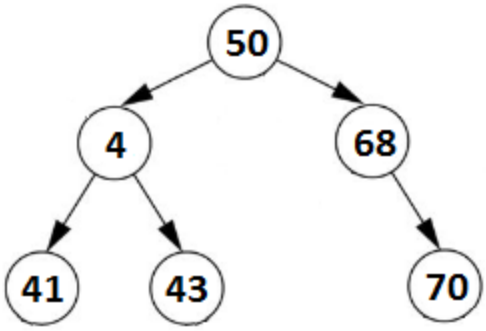




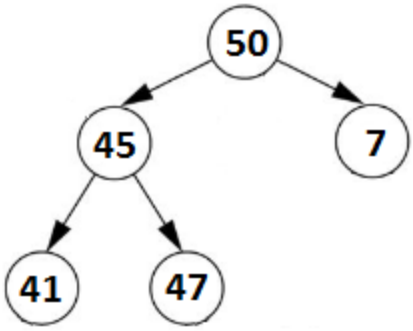
non-complete
binary tree



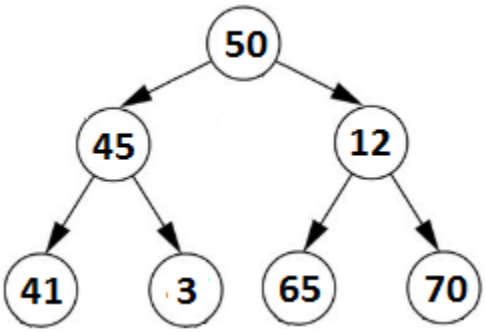
complete
binary tree

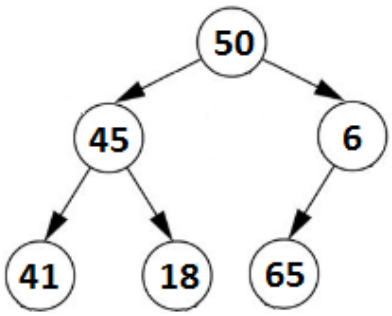


non-full
binary tree

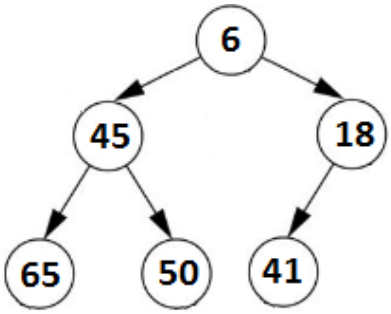


full
binary tree

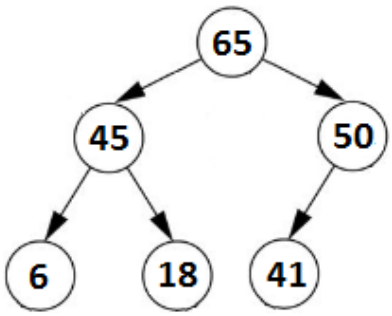




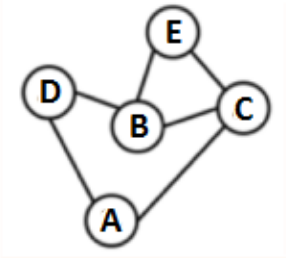
Complete Binary Tree



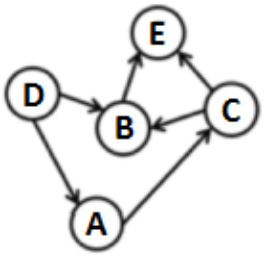
Min Binary Heap



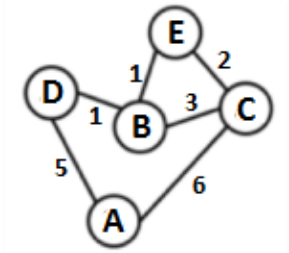
Max Binary Heap



Undirected graph

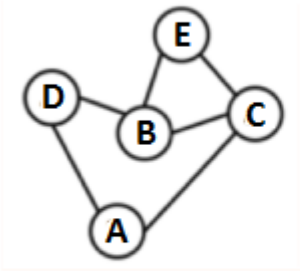


Directed graph



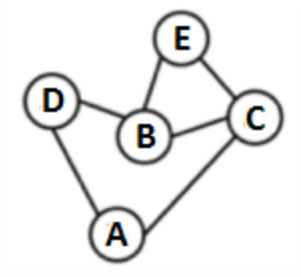
Weighted Graph

Self-loop Graph



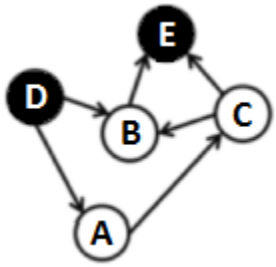
Undirected graph

	A	B	C	D	E
A	0	0	1	1	0
B	0	0	1	1	1
C	1	1	0	0	1
D	1	1	0	0	0
E	0	1	1	0	0



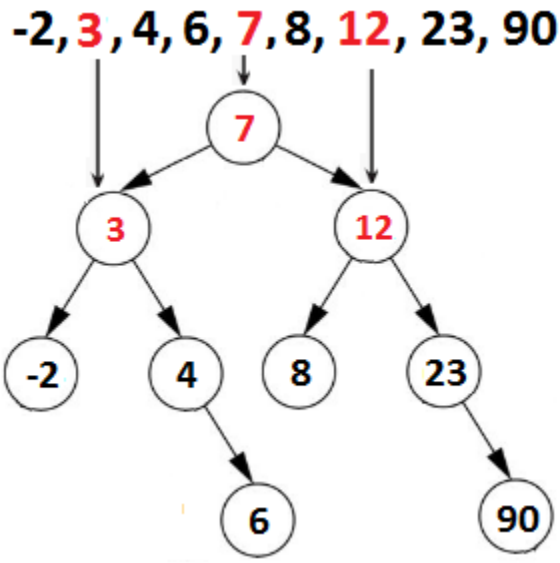
Undirected graph

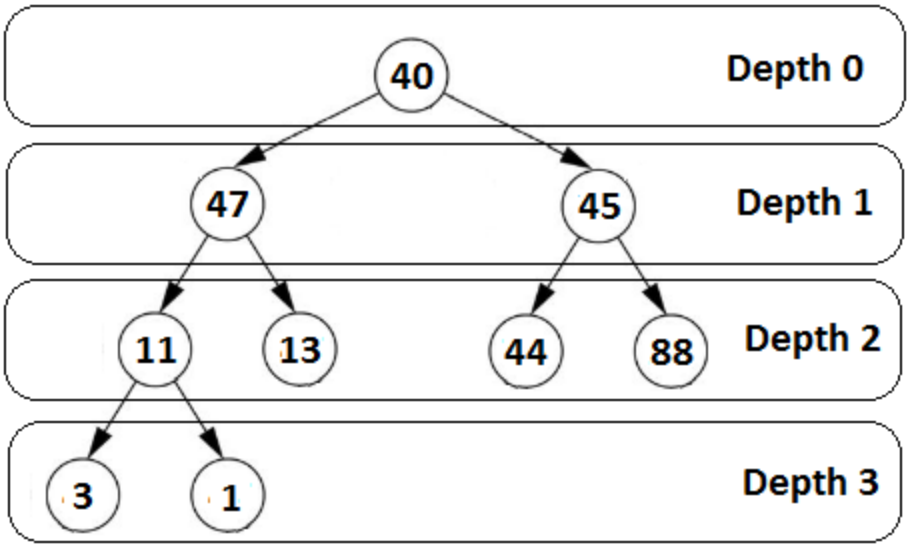
A	C	D	
B	C	E	D
C	A	B	E
D	A	B	
E	C	B	

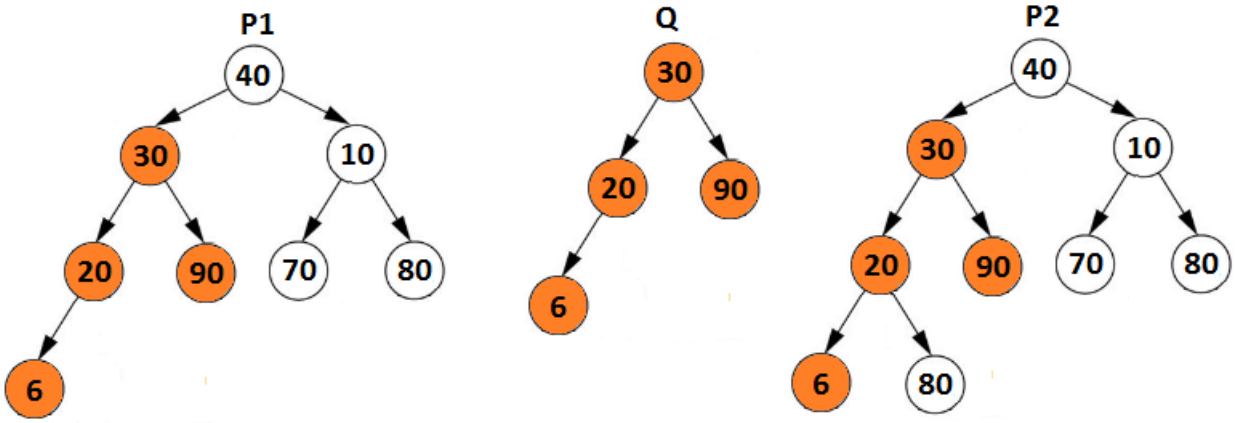


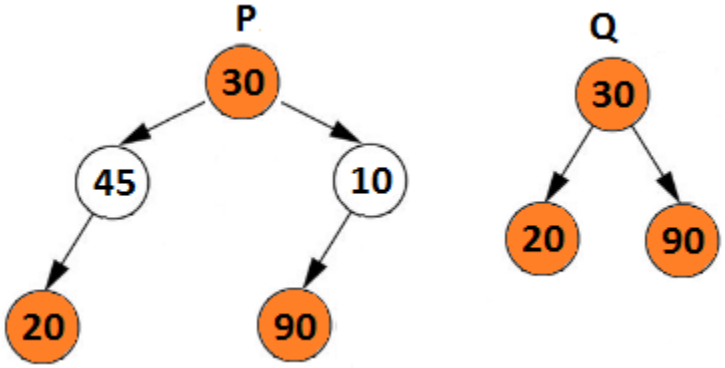
Path from D to E: D -> A -> C -> E
D -> A -> C -> B -> E
D -> B -> E

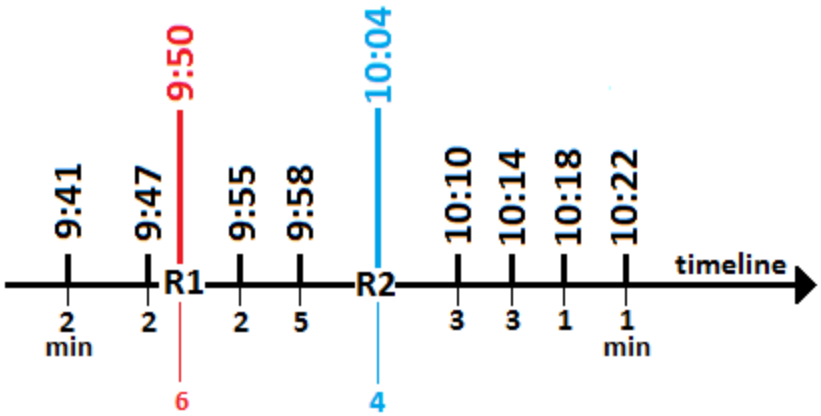
Path from E to D: -

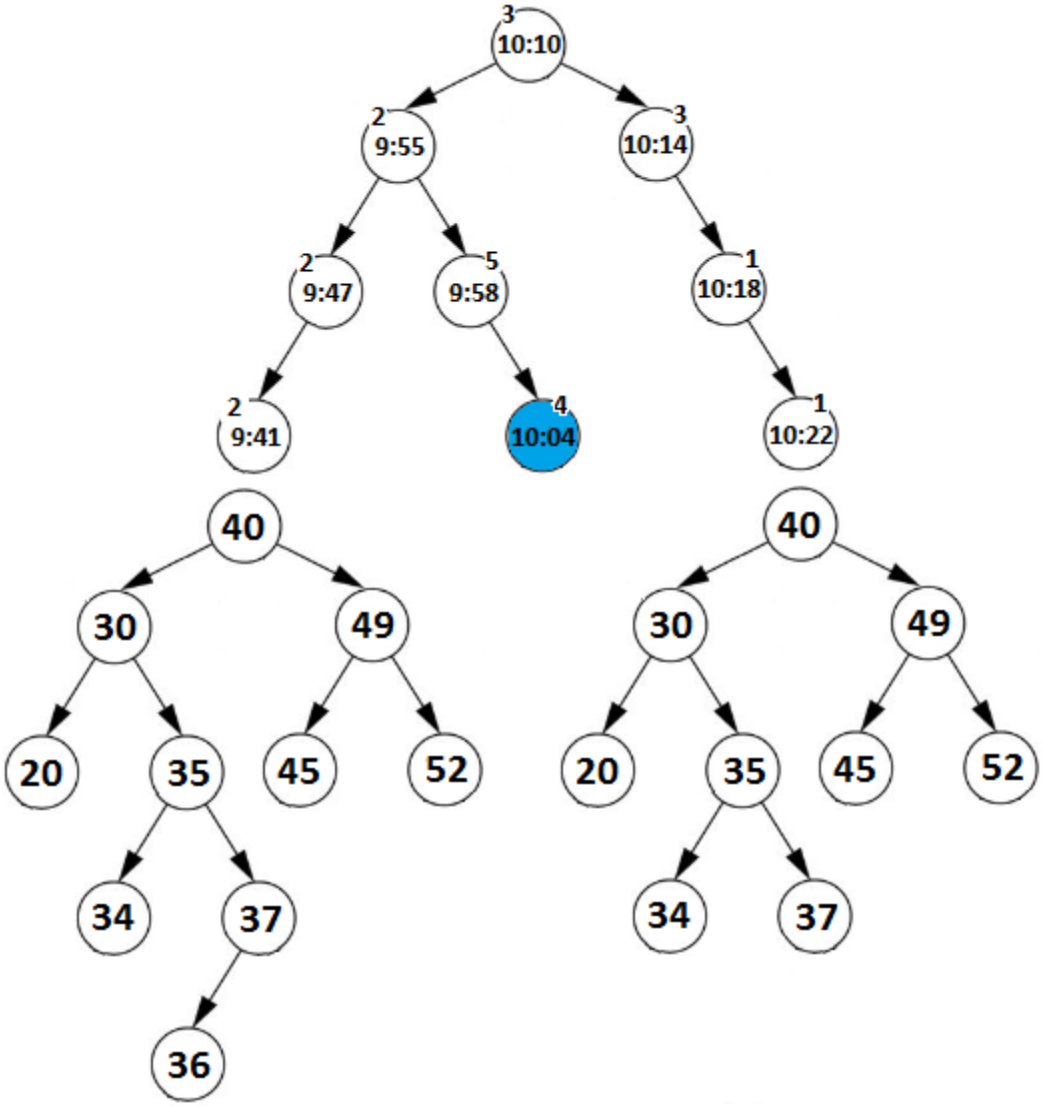



Depth 0
40
Depth 1
47
45
Depth 2
11
13
44
88
Depth 3
3
1


P1
40
30
10
20
90
70
80
6
Q
30
20
90
6
P2
40
30
10
20
90
70
80
6
80


P
30
45
10
20
90
Q
30
20
90

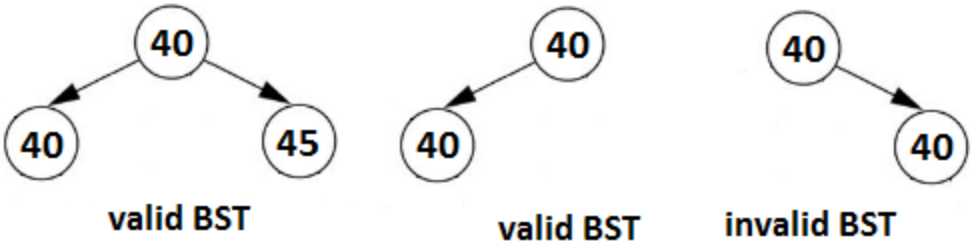




unbalanced binary tree

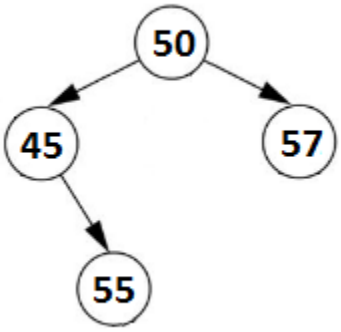
balanced binary tree

left descendants of n ≤ n < right descendants of n

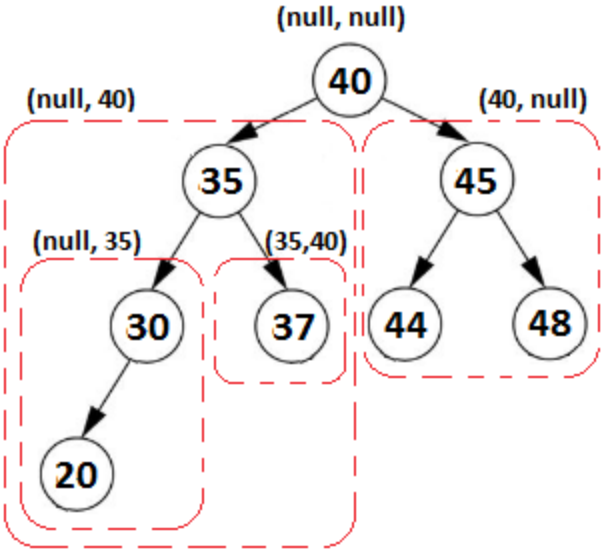


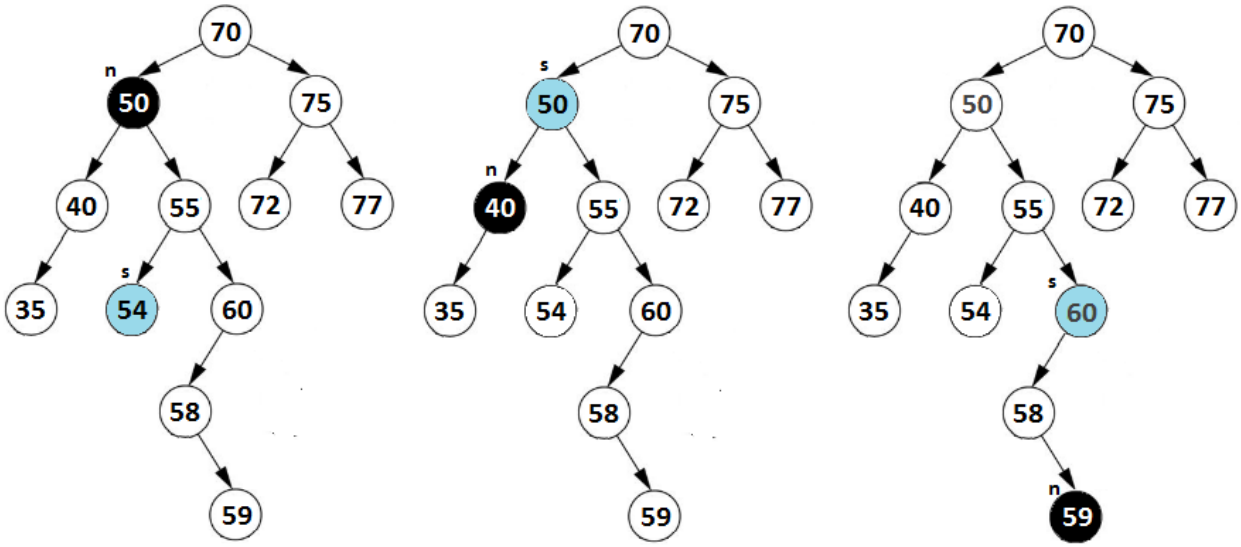
valid BST valid BST invalid BST

left descendants of n ≤ n < right descendants of n

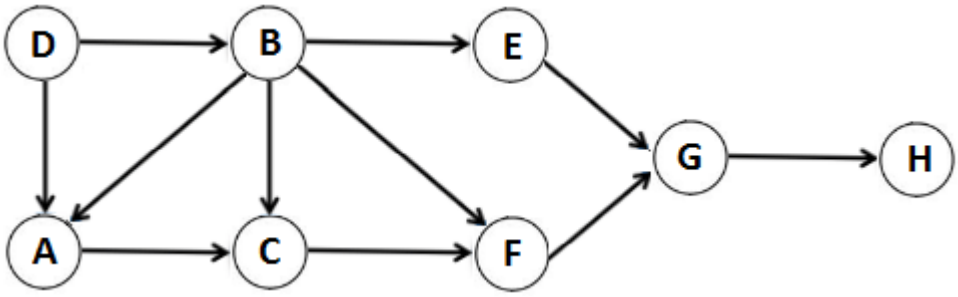


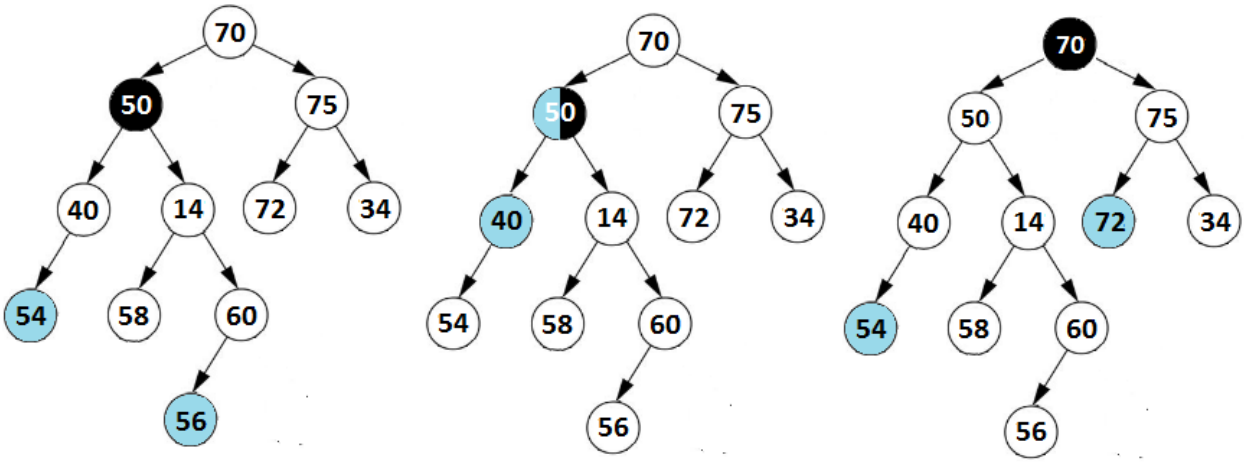
invalid BST



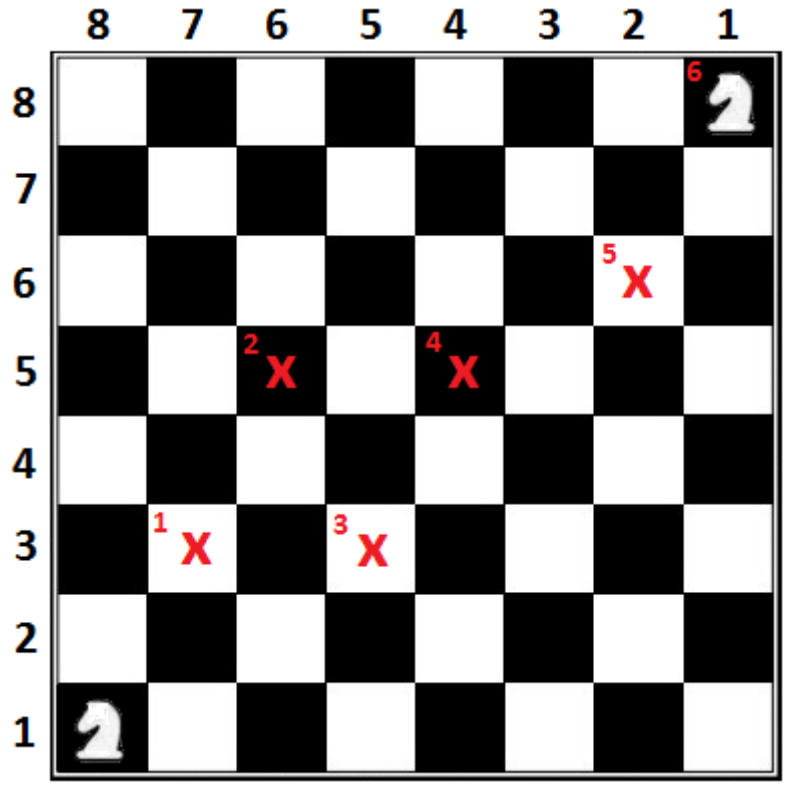


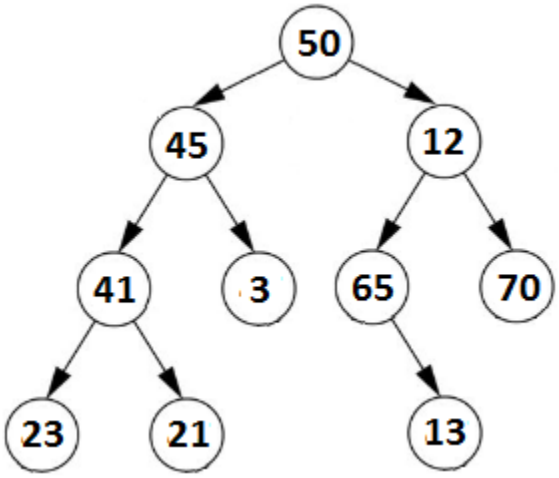
(a) (b)



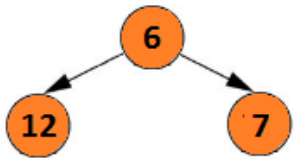


n1=54, n2=56, r=50 n1=40, n2=50, r=50 n1=54, n2=72, r= 70

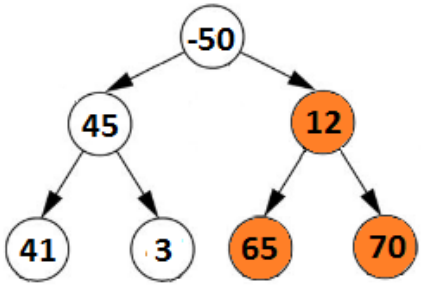




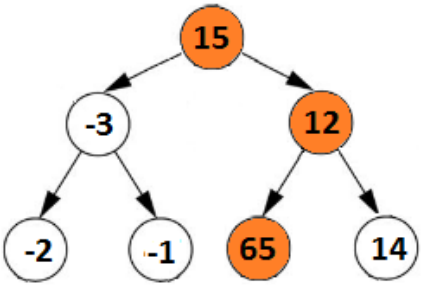
Level 0: 50
Level 1: 45, 12
Level 2: 41, 70
Level 3: 23, 13



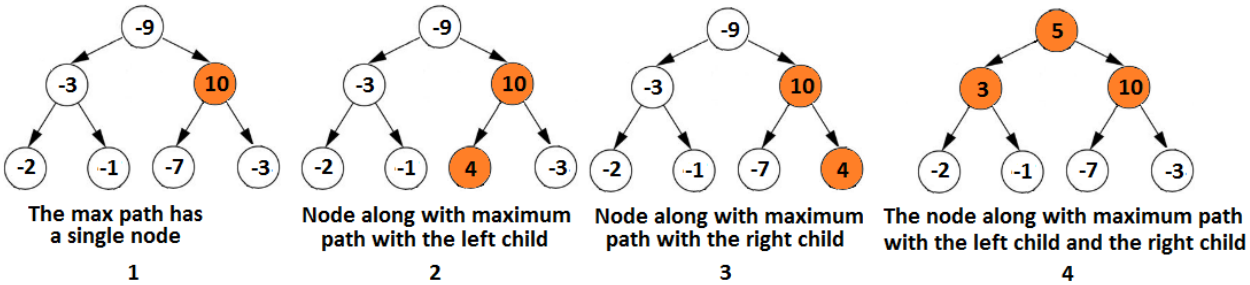
Max path sum: 25



Max path sum: 147



Max path sum: 92



The max path has a single node

1

Node along with maximum path with the left child

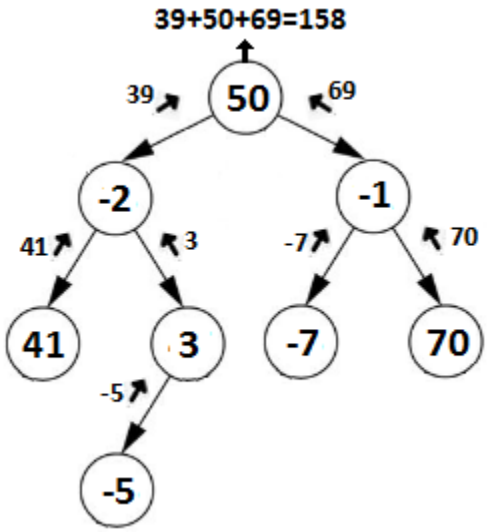
2

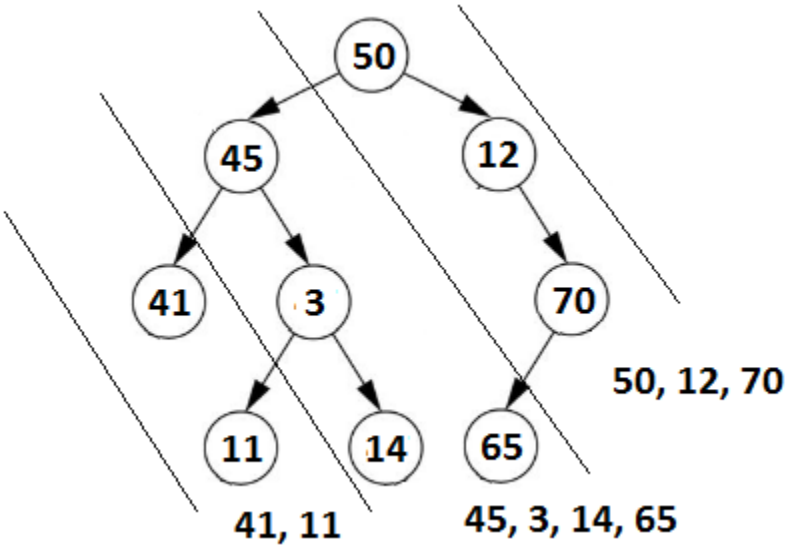
Node along with maximum path with the right child

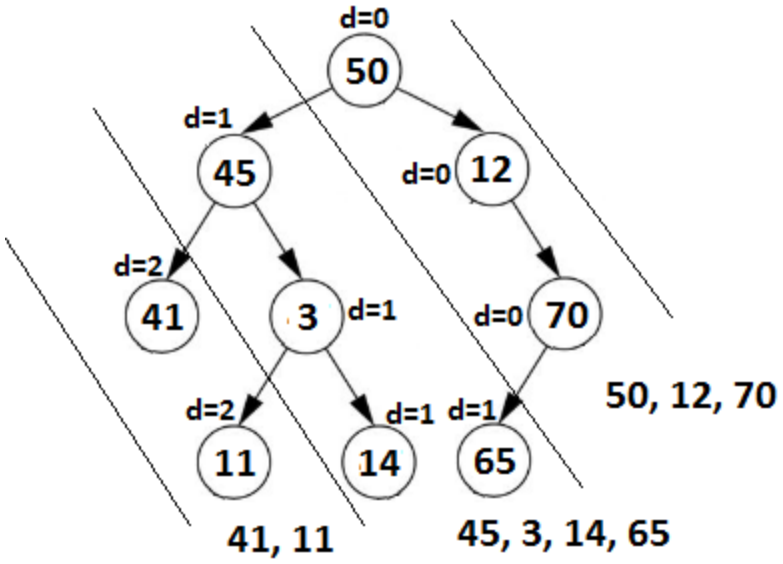
3

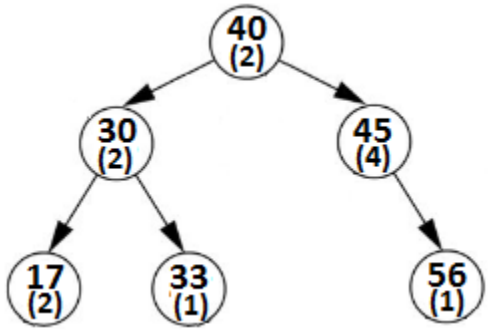
The node along with maximum path with the left child and the right child

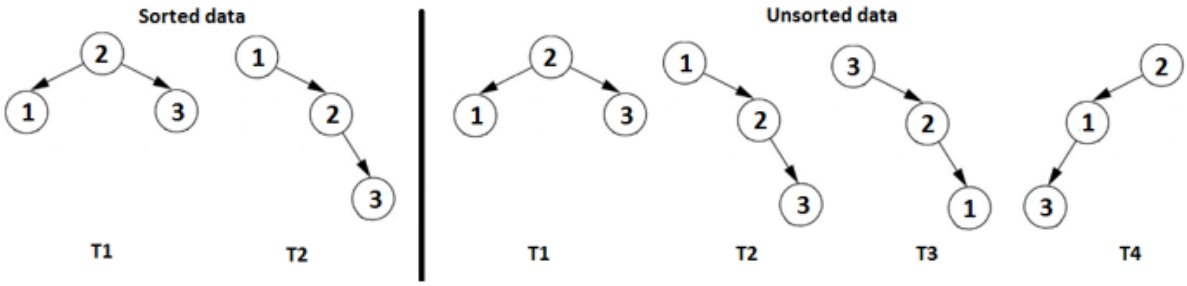
4

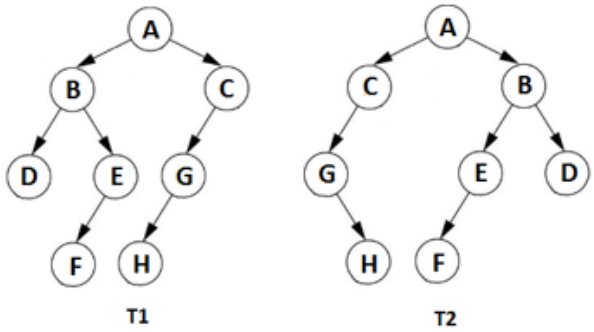


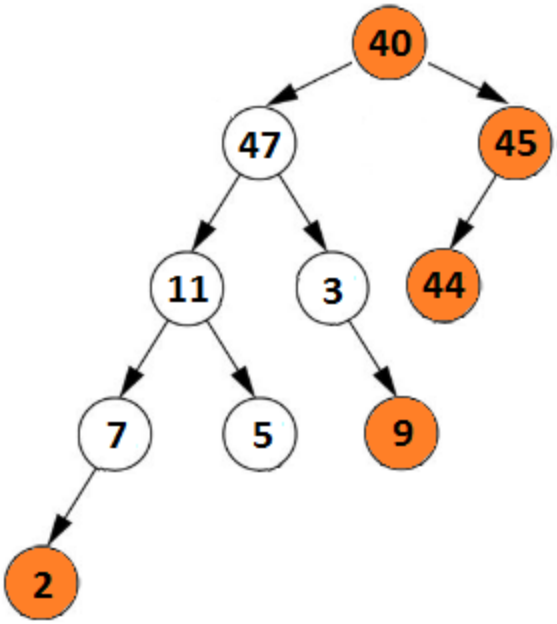


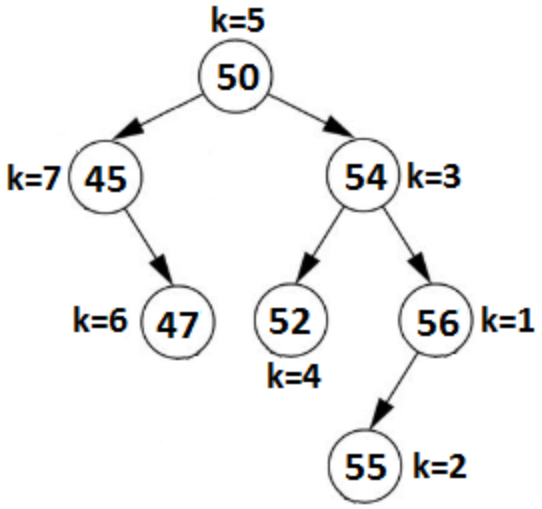

d=0
50
d=1
45
d=0
12
d=2
41
3
d=1
d=0
70
d=2
11
d=1
14
d=1
65
50, 12, 70
41, 11
45, 3, 14, 65

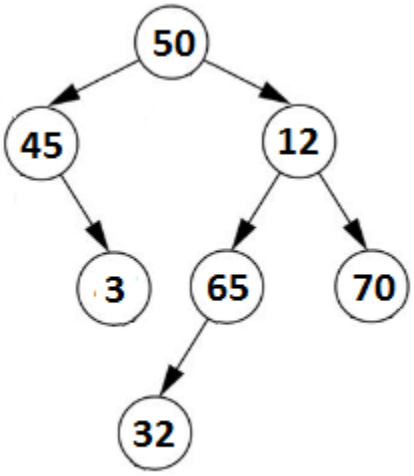

40
(2)
30
(2)
45
(4)
17
(2)
33
(1)
56
(1)


Sorted data
Unsorted data
T1
T2
T1
T2
T3
T4

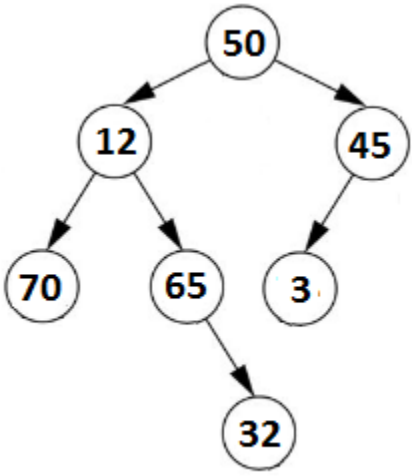

T1
T2



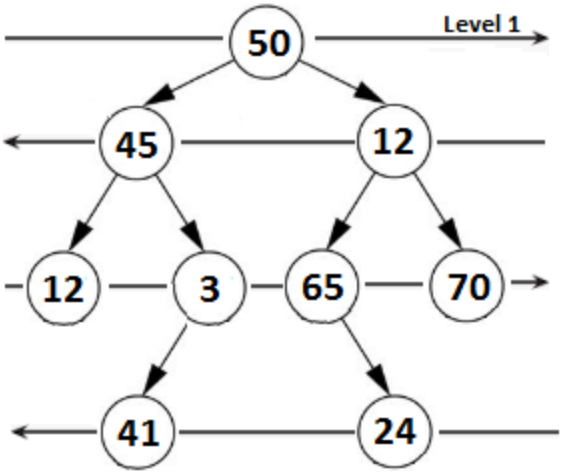




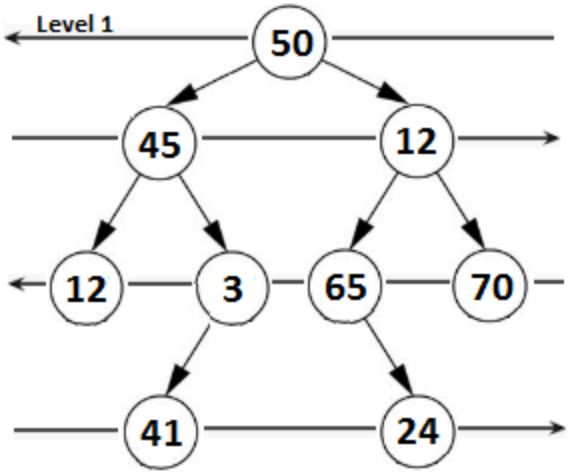
Given tree



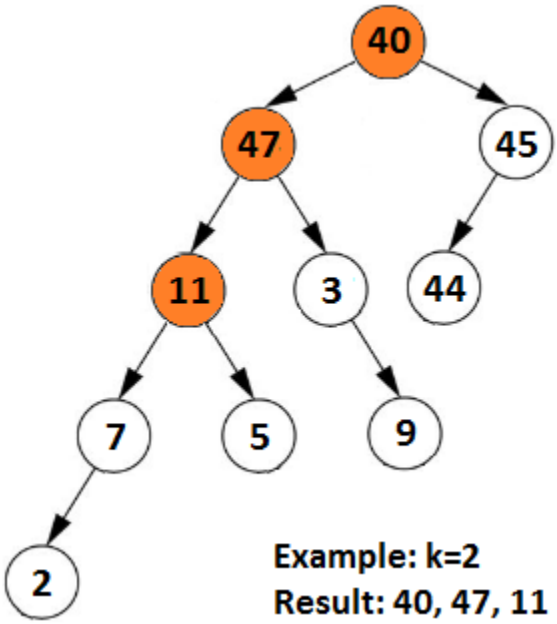
Mirror tree



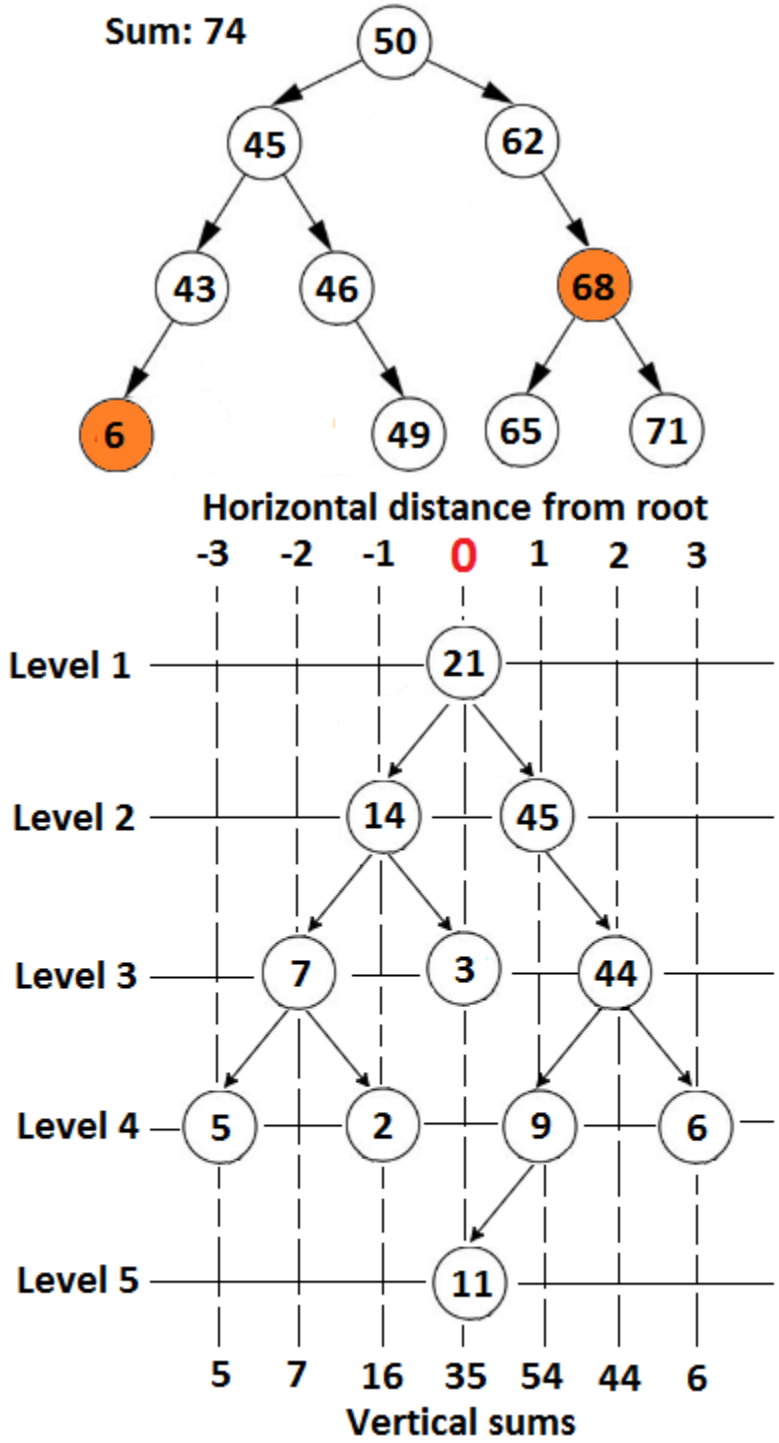
odd levels printed
from left to right

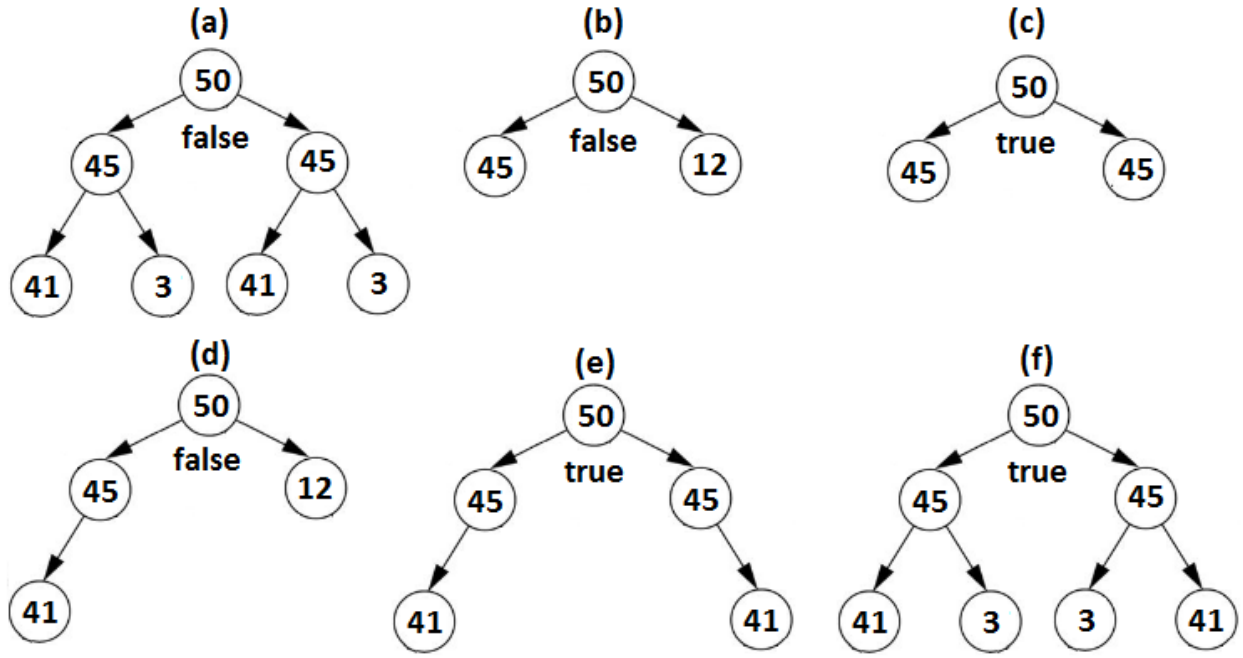


odd levels printed
from right to left

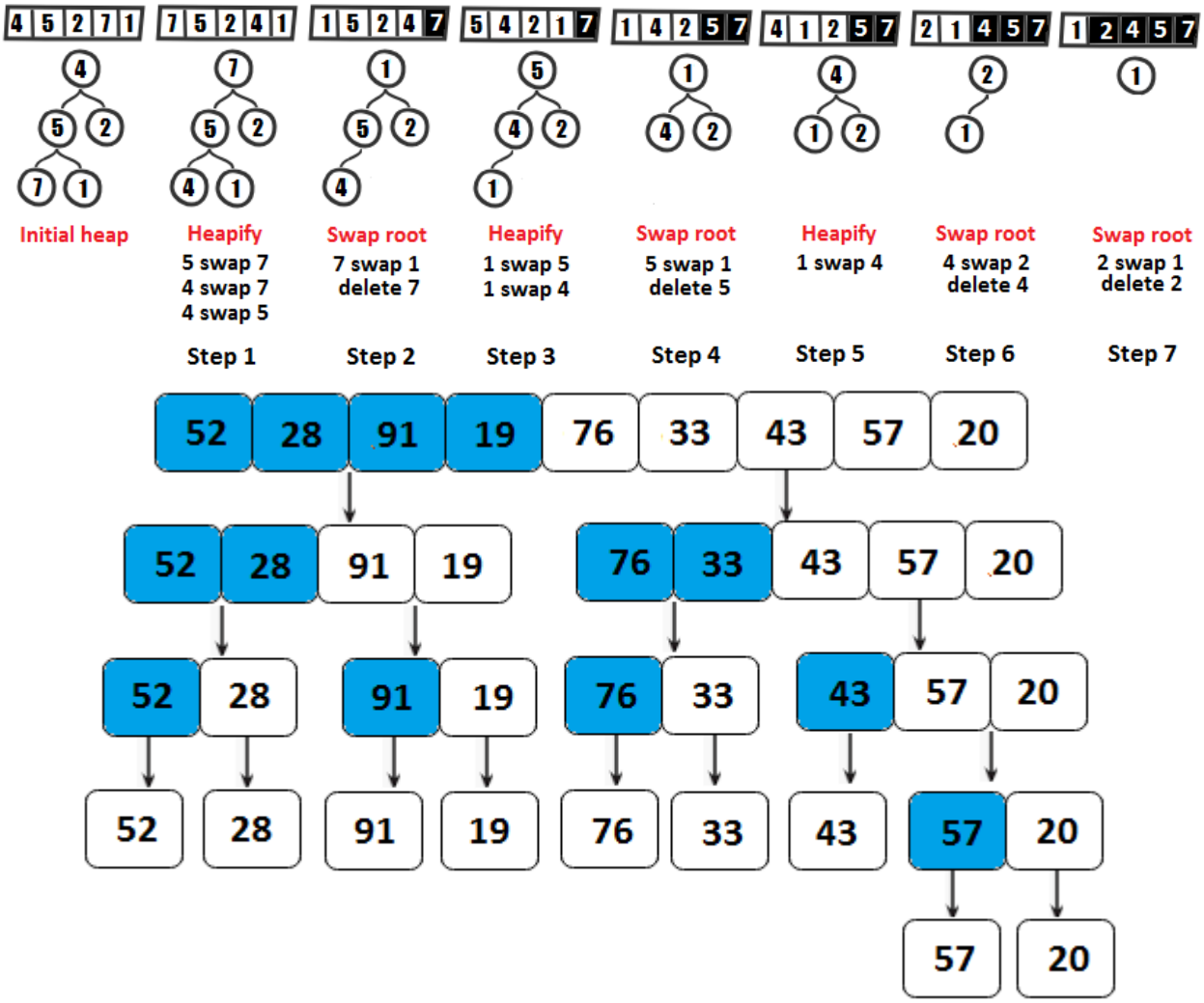


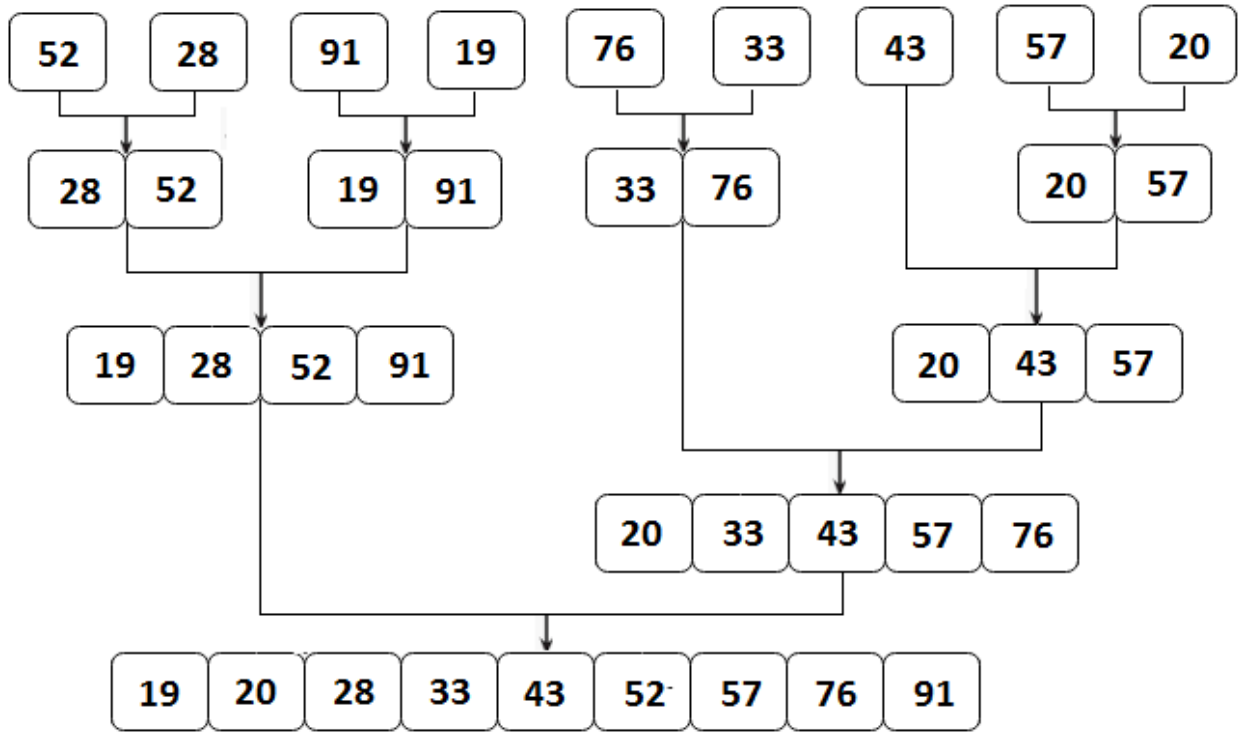
Example: k=2
Result: 40, 47, 11

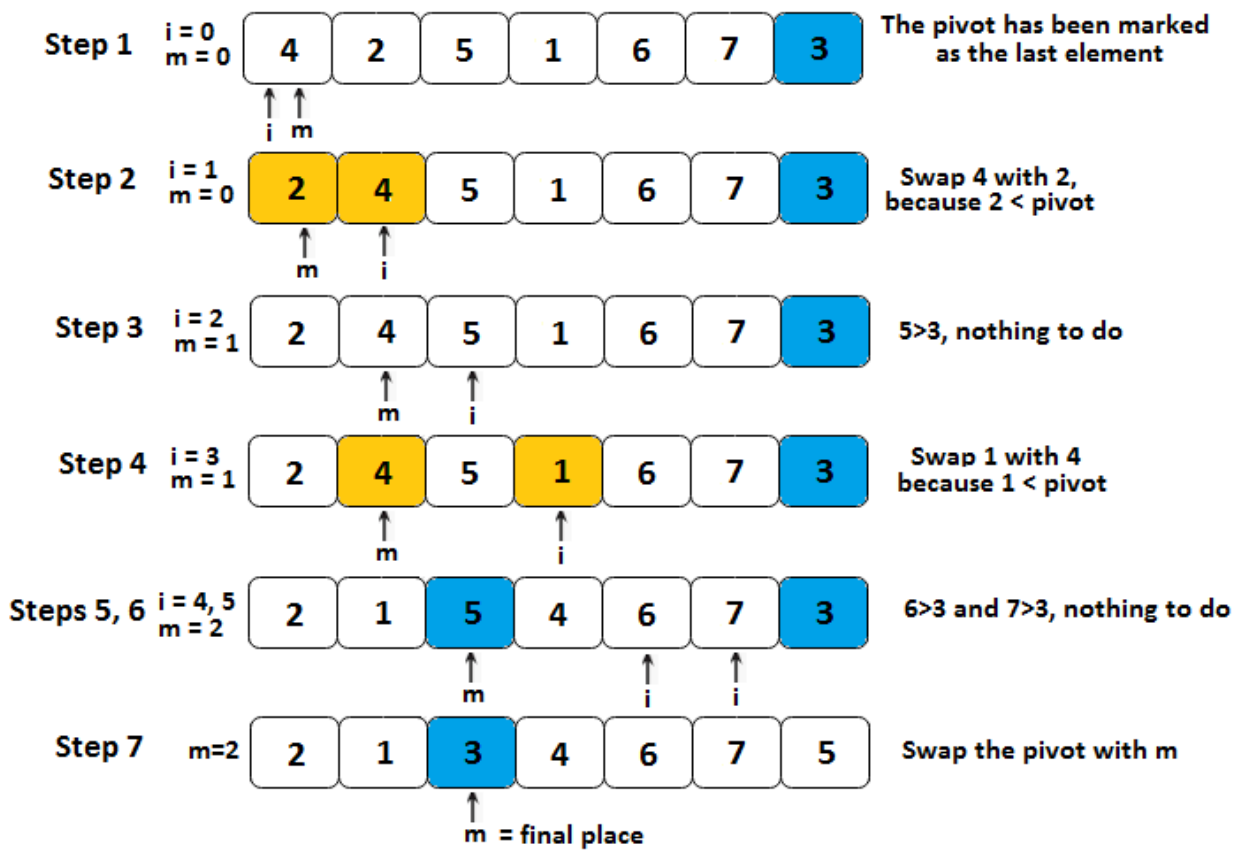

Sum: 74
50
45
62
43
46
68
6
49
65
71
Horizontal distance from root
-3 -2 -1 0 1 2 3
Level 1
21
Level 2
14
45
Level 3
7
3
44
Level 4
5
2
9
6
Level 5
11
5 7 16 35 54 44 6
Vertical sums

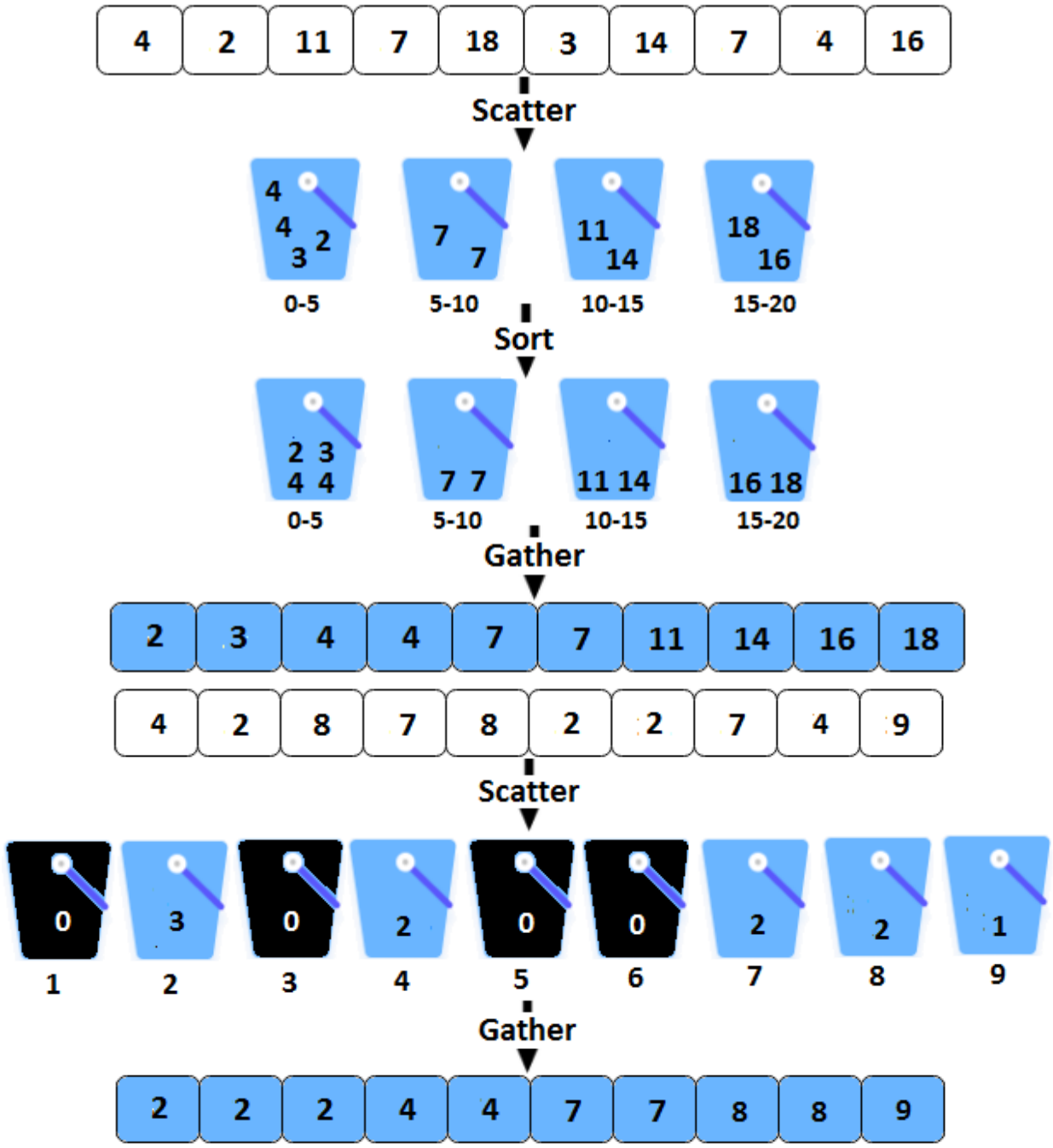

(a)
50
false
45
45
41
3
41
3
(b)
50
false
45
12
(c)
50
true
45
45
(d)
50
false
45
12
41
(e)
50
true
45
45
41
41
(f)
50
true
45
45
41
3
3
41

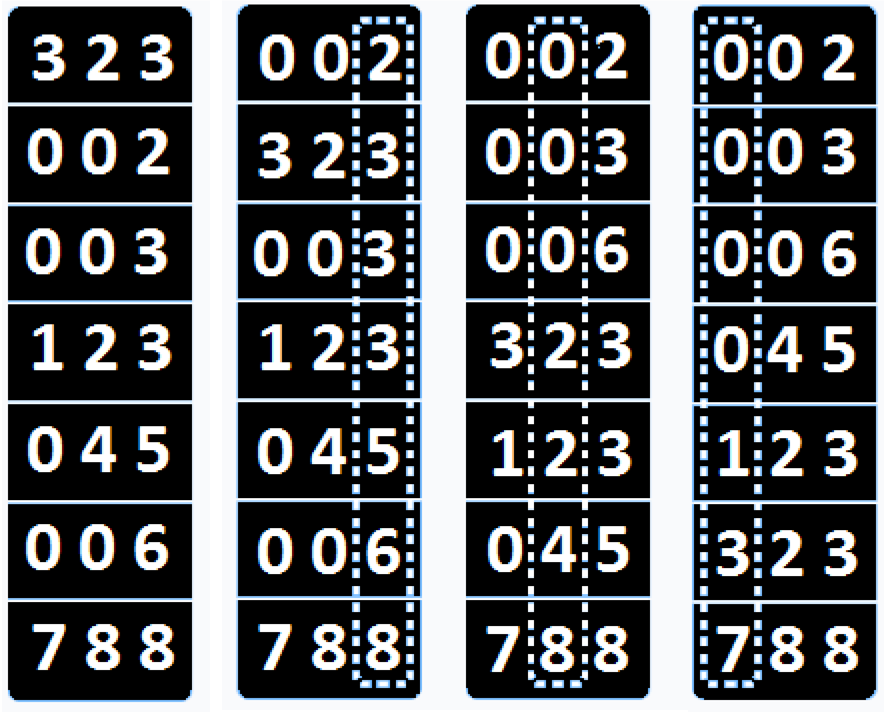
Chapter 14: Sorting and Searching



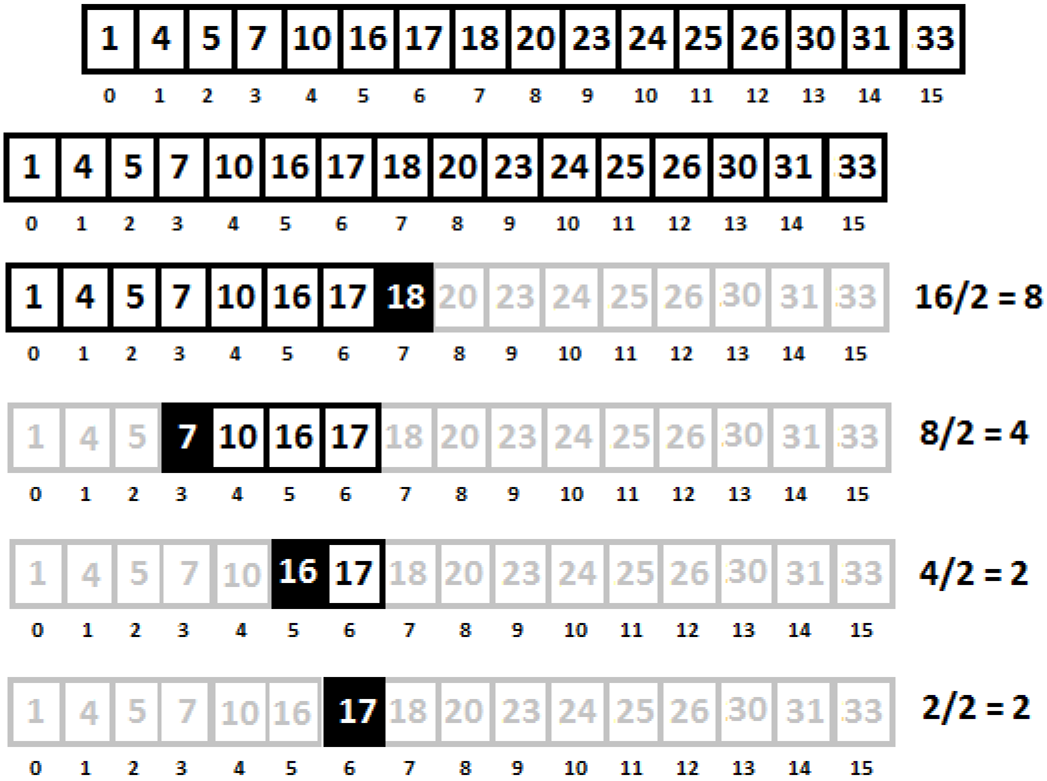

52 28 91 19 76 33 43 57 20
28 52 19 91 33 76 20 57
19 28 52 91 20 43 57
20 33 43 57 76
19 20 28 33 43 52 57 76 91

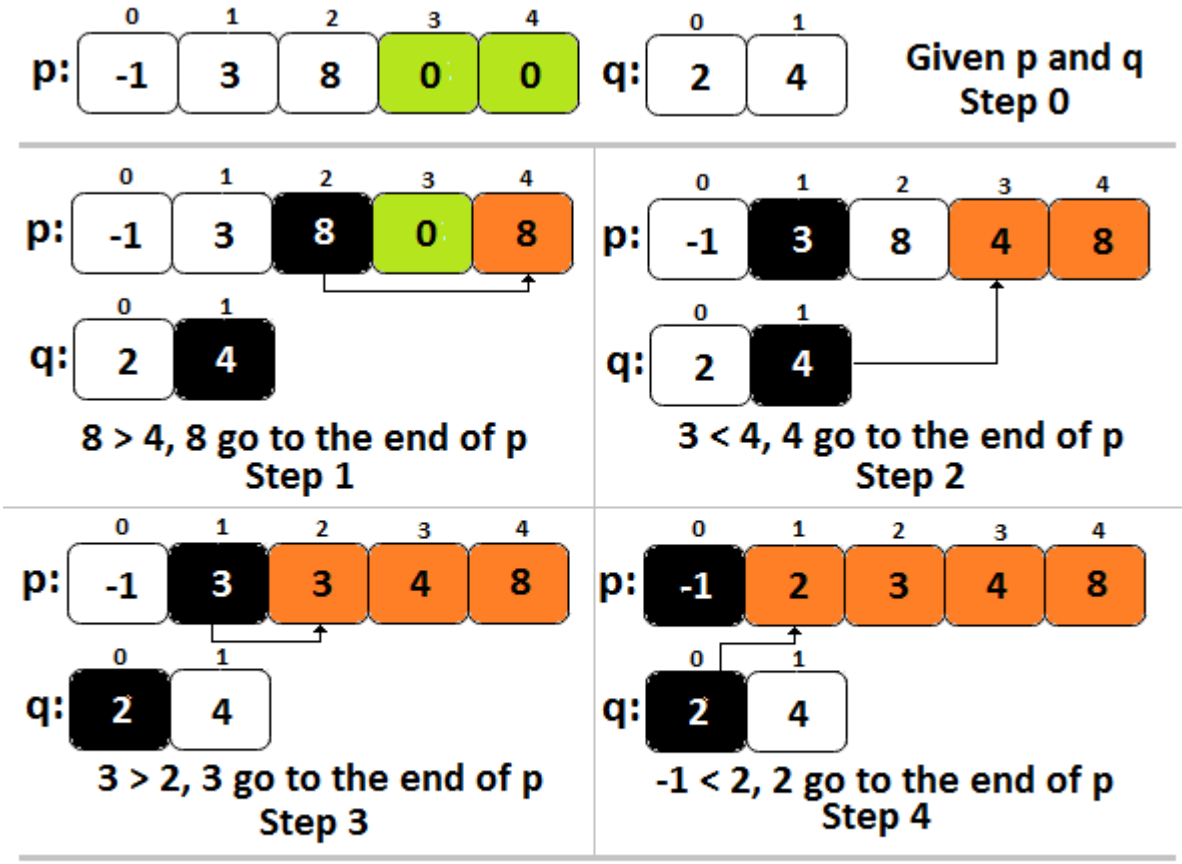

Step 1 i = 0 m = 0 4 2 5 1 6 7 3 The pivot has been marked as the last element
i m
Step 2 i = 1 m = 0 2 4 5 1 6 7 3 Swap 4 with 2, because 2 < pivot
m i
Step 3 i = 2 m = 1 2 4 5 1 6 7 3 5>3, nothing to do
m i
Step 4 i = 3 m = 1 2 4 5 1 6 7 3 Swap 1 with 4 because 1 < pivot
m i
Steps 5, 6 i = 4, 5 m = 2 2 1 5 4 6 7 3 6>3 and 7>3, nothing to do
m i i
Step 7 m=2 2 1 3 4 6 7 5 Swap the pivot with m
m = final place

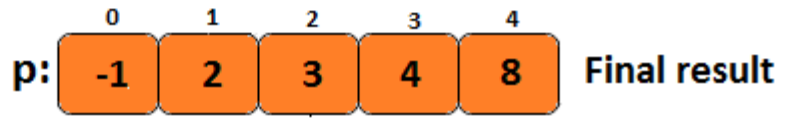

4 2 11 7 18 3 14 7 4 16
Scatter
4 4 3 2
7 7
11 14
18 16
0-5
5-10
10-15
15-20
Sort
2 3 4 4
7 7
11 14
16 18
0-5
5-10
10-15
15-20
Gather
2 3 4 4 7 7 11 14 16 18
4 2 8 7 8 2 2 7 4 9
Scatter
0 3 0 2 0 0 2 2 1
1 2 3 4 5 6 7 8 9
Gather
2 2 2 4 4 7 7 8 8 9

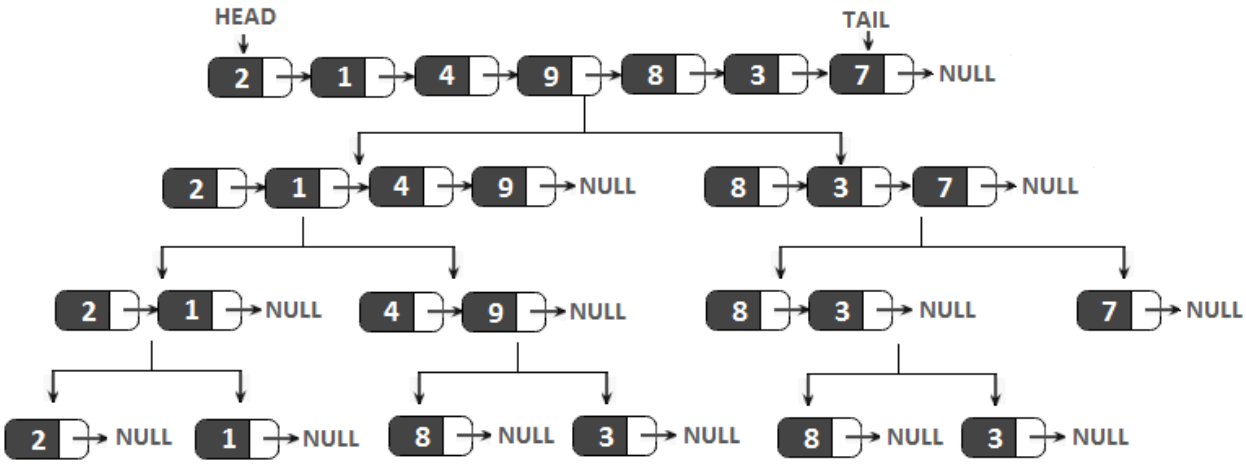


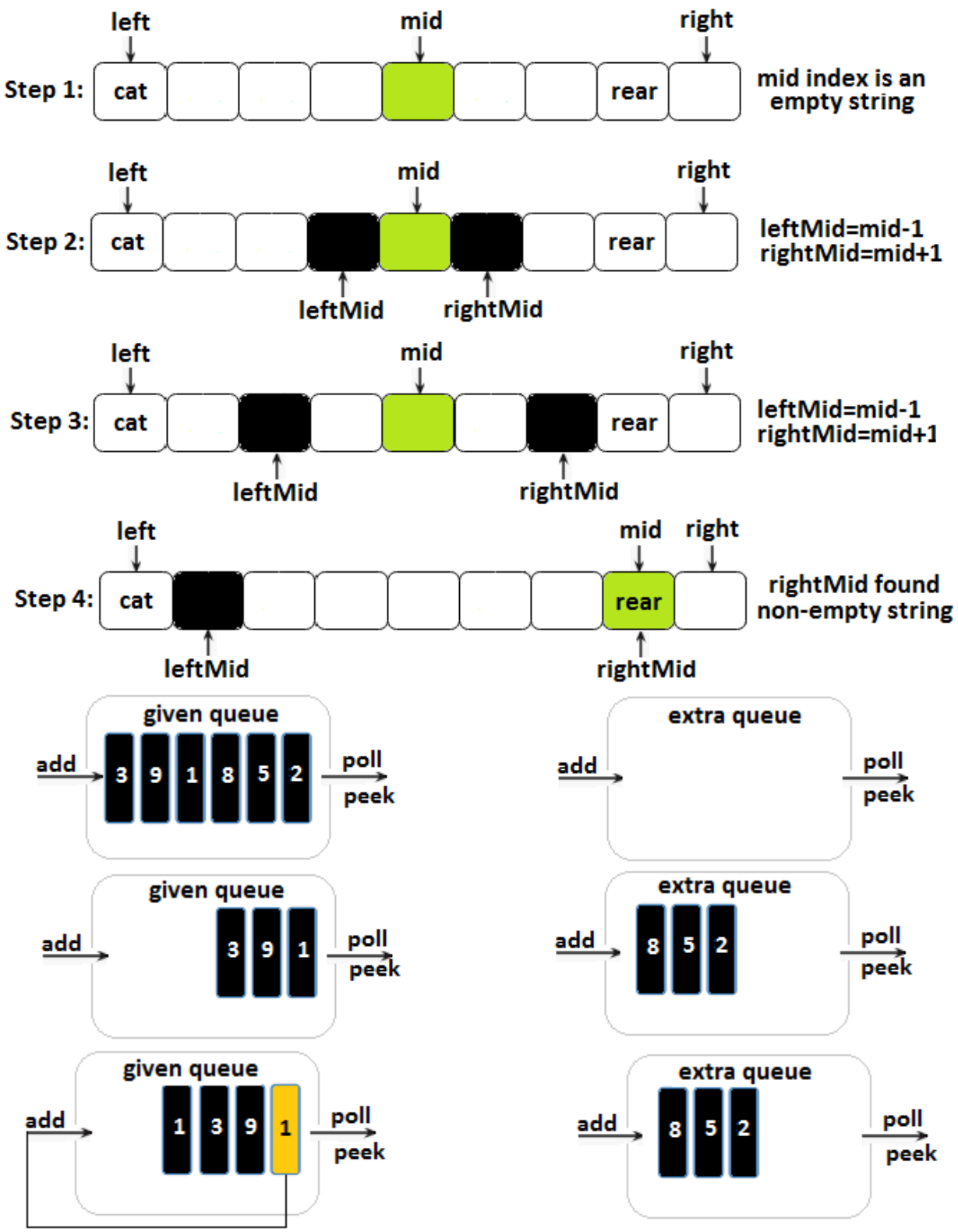
Initial array units sort tens sort hundreds sort

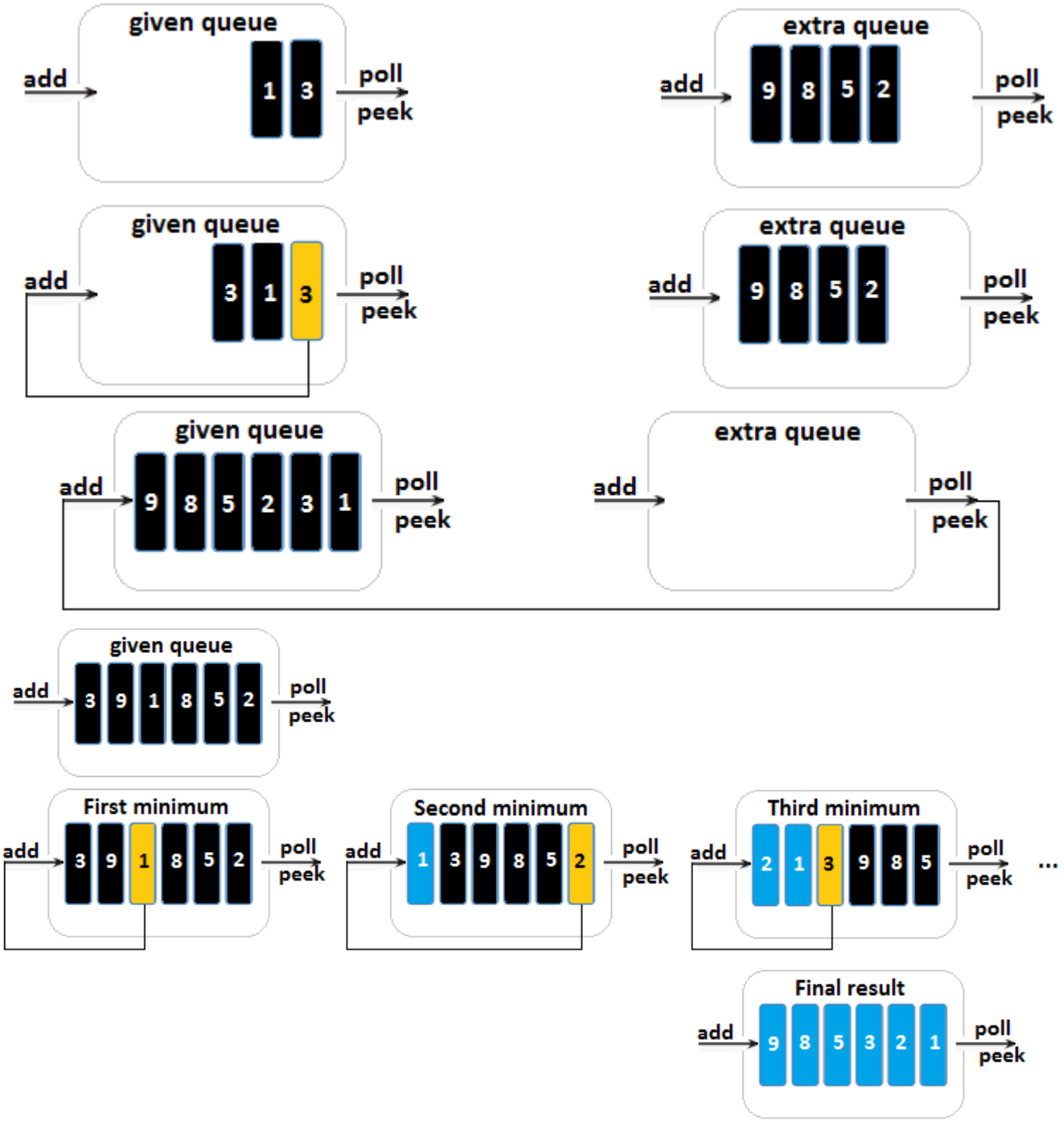


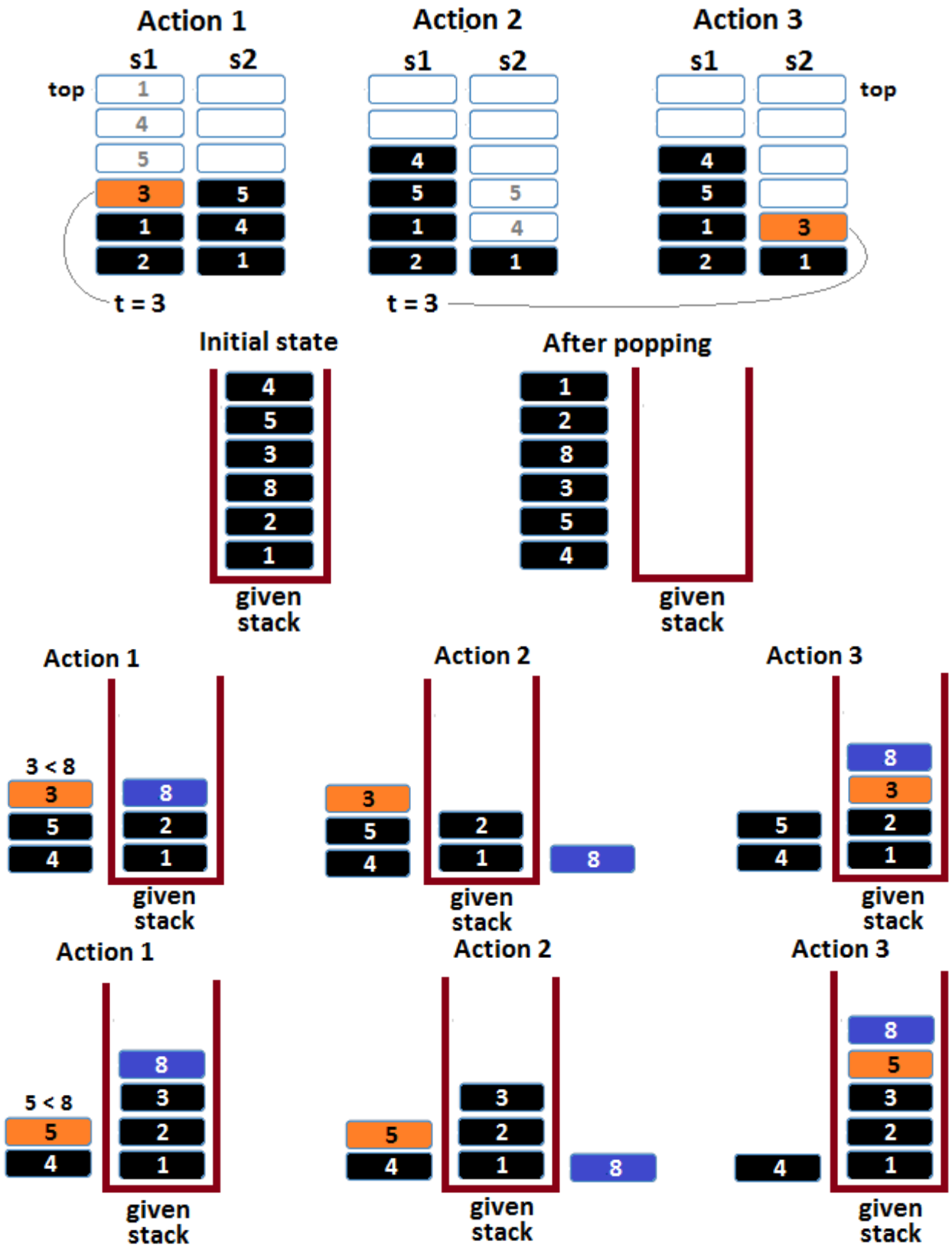

p: -1 3 8 0 0
q: 2 4
Given p and q
Step 0
p: -1 3 8 0 8
q: 2 4
8 > 4, 8 go to the end of p
Step 1
p: -1 3 8 4 8
q: 2 4
3 < 4, 4 go to the end of p
Step 2
p: -1 3 3 4 8
q: 2 4
3 > 2, 3 go to the end of p
Step 3
p: -1 2 3 4 8
q: 2 4
-1 < 2, 2 go to the end of p
Step 4

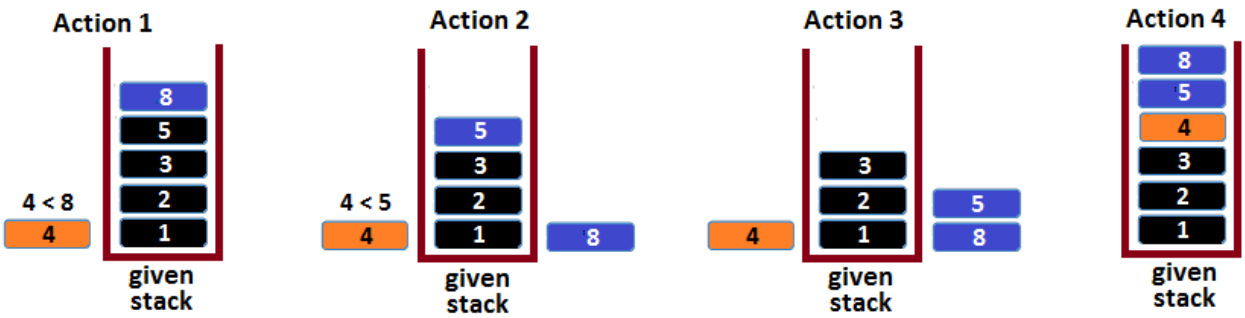

p: -1 2 3 4 8
Final result

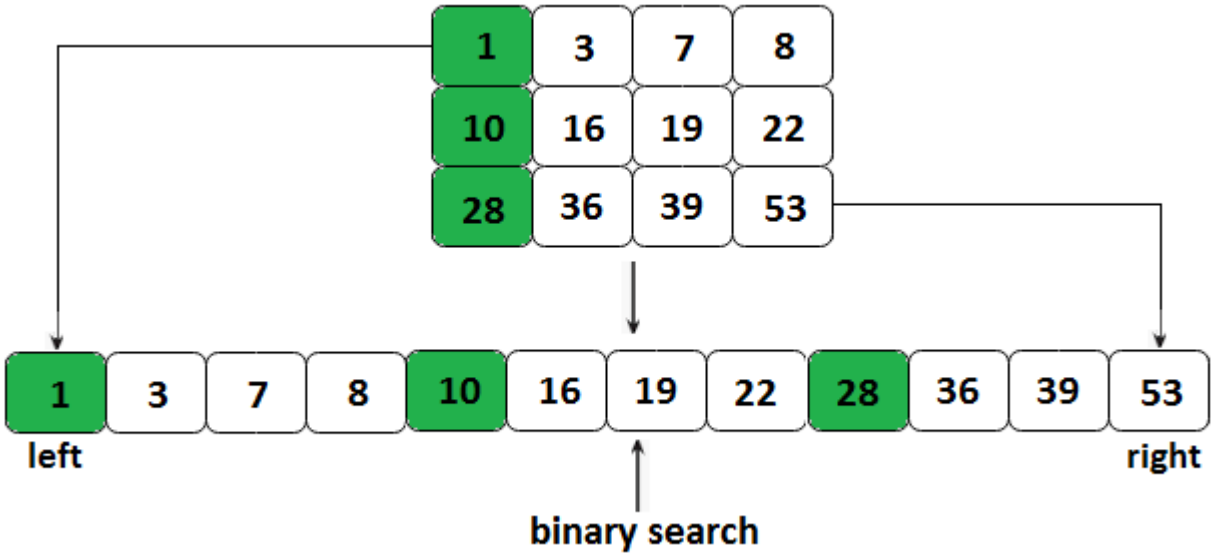

HEAD
TAIL
2 1 4 9 8 3 7 NULL
2 1 4 9 NULL
8 3 7 NULL
2 1 NULL
4 9 NULL
8 3 NULL
7 NULL
2 NULL
1 NULL
8 NULL
3 NULL
8 NULL
3 NULL


left
mid
right
Step 1:
cat
rear
mid index is an empty string
left
mid
right
Step 2:
cat
rear
leftMid=mid-1
rightMid=mid+1
leftMid
rightMid
left
mid
right
Step 3:
cat
rear
leftMid=mid-1
rightMid=mid+1
leftMid
rightMid
left
mid
right
Step 4:
cat
rear
rightMid found non-empty string
leftMid
rightMid
given queue
add
3 9 1 8 5 2
poll
peek
extra queue
add
poll
peek
given queue
add
3 9 1
poll
peek
extra queue
add
8 5 2
poll
peek
given queue
add
1 3 9 1
poll
peek
extra queue
add
8 5 2
poll
peek


given queue
add
1 3
poll
peek
extra queue
add
9 8 5 2
poll
peek
given queue
add
3 1 3
poll
peek
extra queue
add
9 8 5 2
poll
peek
given queue
add
9 8 5 2 3 1
poll
peek
extra queue
add
poll
peek
given queue
add
3 9 1 8 5 2
poll
peek
First minimum
add
3 9 1 8 5 2
poll
peek
Second minimum
add
1 3 9 8 5 2
poll
peek
Third minimum
add
2 1 3 9 8 5
poll
peek
...
Final result
add
9 8 5 3 2 1
poll
peek

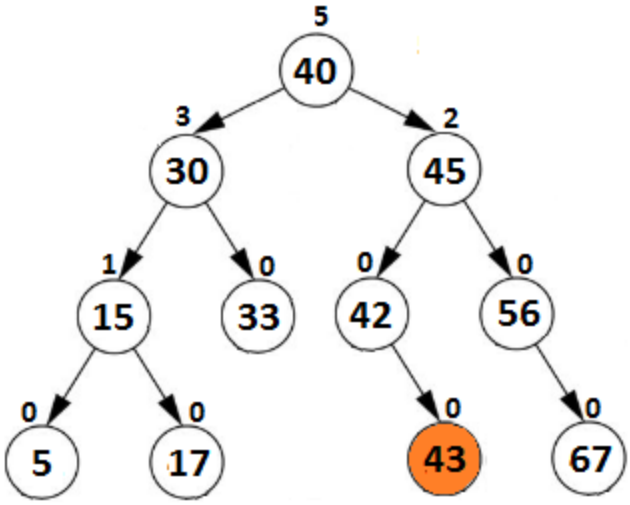

Action 1
s1
s2
top
1
4
5
3
5
1
4
2
1
t = 3
Action 2
s1
s2
4
5
5
1
4
2
1
t = 3
Action 3
s1
s2
top
4
5
1
3
2
1
Initial state
4
5
3
8
2
1
given stack
After popping
1
2
8
3
5
4
given stack
Action 1
3 < 8
3
5
4
8
2
1
given stack
Action 2
3
5
4
2
1
8
given stack
Action 3
8
3
2
1
5
4
given stack
Action 1
5 < 8
5
4
8
3
2
1
given stack
Action 2
5
4
3
2
1
8
given stack
Action 3
8
5
3
2
1
4
given stack



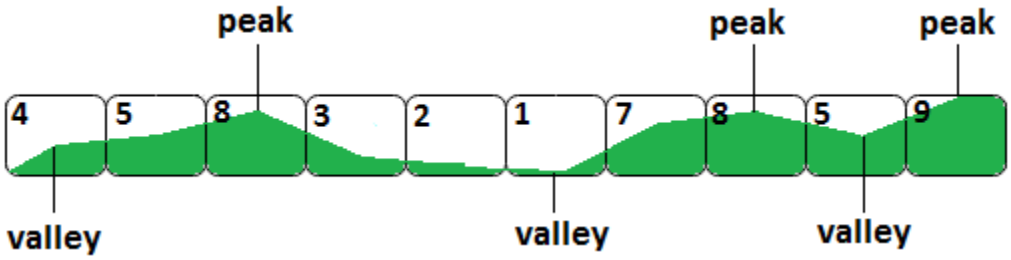


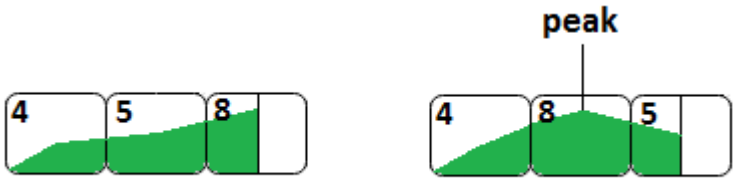
	0	1	2	3	4	5
0	11	22	48	77	78	84
1	12	24	55	78	83	90
2	25	56	58	80	85	95
3	33	57	60	85	86	99

	0	1	2	3	4	5
0	11	22	48	77	78	84
1	12	24	55	78	83	90
2	25	56	58	80	85	95
3	33	57	60	85	86	99



rank(43)=7



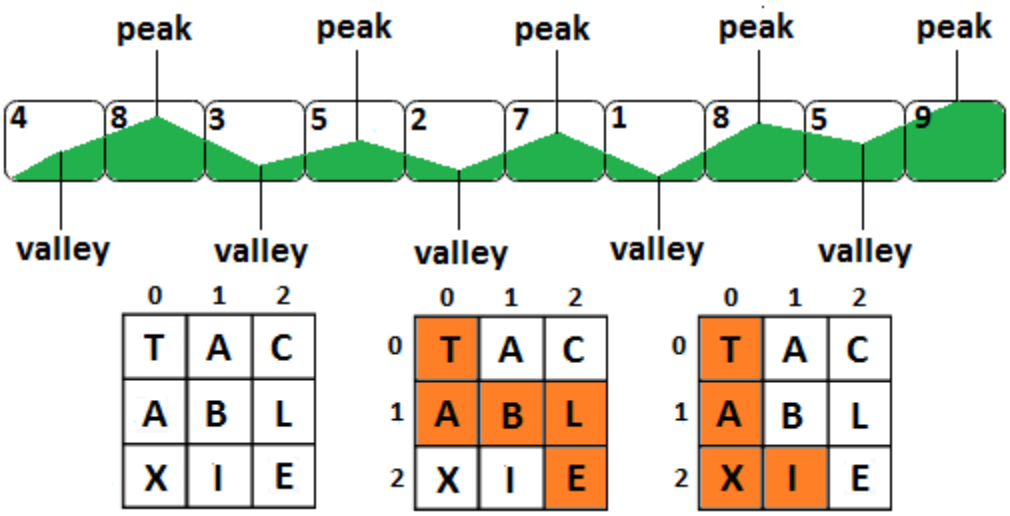


Before swapping

After swapping

Before swapping

After swapping


peak
peak
peak
peak
peak
4
8
3
5
2
7
1
8
5
9
valley
valley
valley
valley
valley
0 1 2
T A C
A B L
X I E
0 1 2
0 T A C
1 A B L
2 X I E
0 1 2
0 T A C
1 A B L
2 X I E

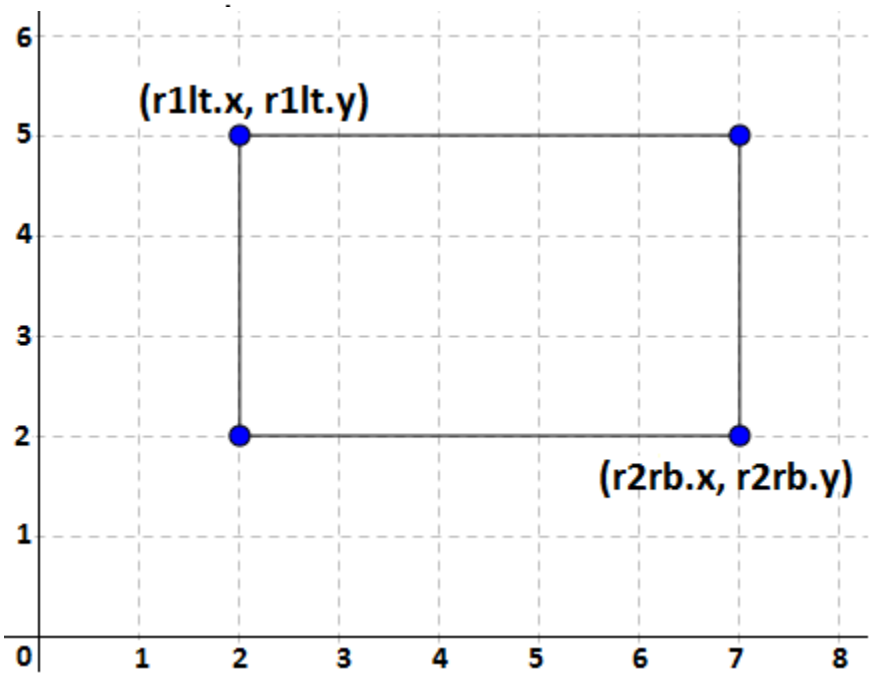
Chapter 15: Mathematics and Puzzles

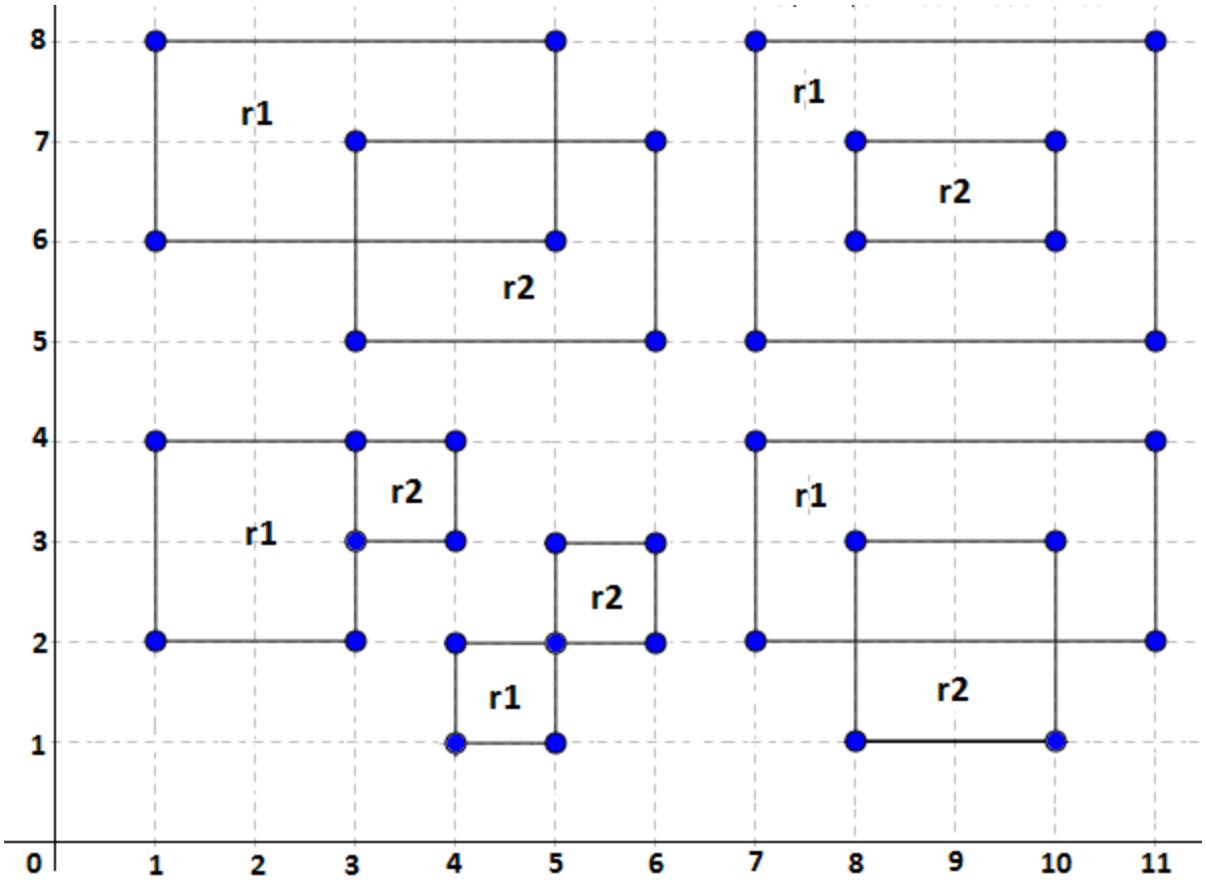
SYMBOL	VALUE
I	1
IV	4
V	5
IX	9
X	10
XL	40
L	50
XC	90
C	100
CD	400
D	500
CM	900
M	1000

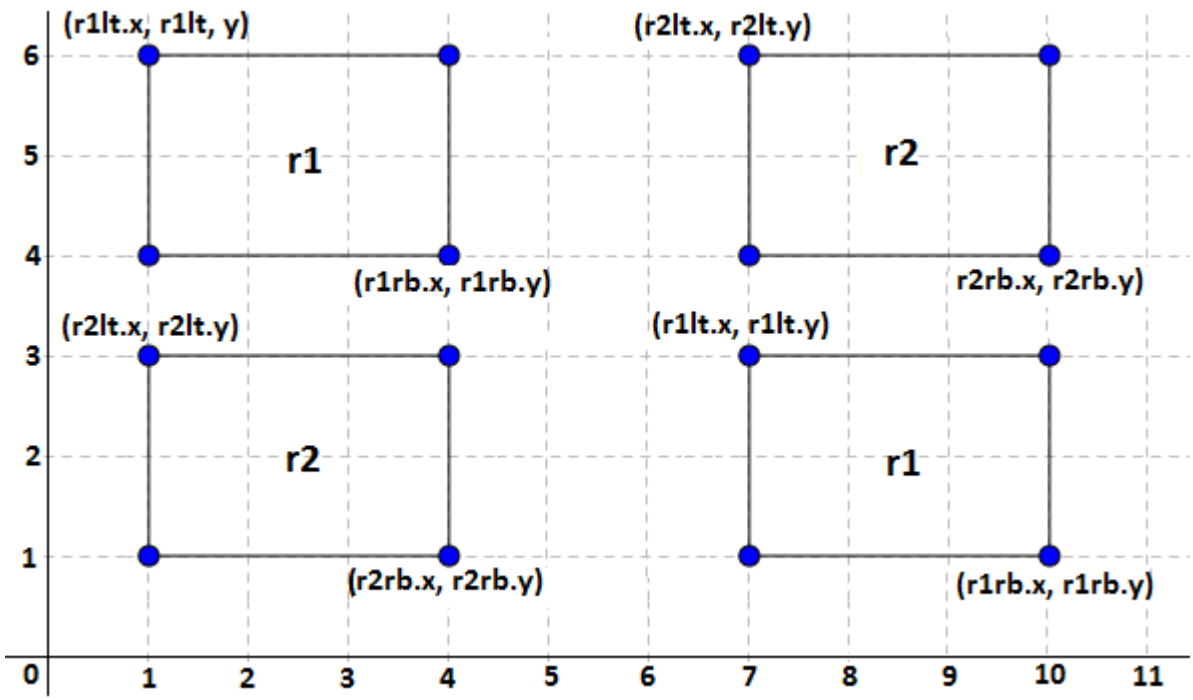
```
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010
1000111000111000111000111000111000111000111000111000111000111000111000111000111000111000111000111000111000111000
1001000010000001000000001000000000010000000000001000000000000001000000000000000010000000000000000001
```

2*2 3*3 4*4 5*5 6*6 7*7 8*8 9*9 10*10

Step 1	11111111111	$\boxed{1}$	111111111 ...
Step 2	10101010101	$\boxed{0}$	101010101 ...
Step 3	10001110001	$\boxed{1}$	100011100 ...
Step 4	10011111001	$\boxed{0}$	100111110 ...
	10010111011	0	101111100 ...
Step 6	10010011011	$\boxed{1}$	101110100 ...
	10010001011	1	111110101 ...
	10010000011	1	111010101 ...
	10010000111	1	111011101 ...
	10010000101	1	111011111 ...
	10010000100	1	111011111 ...
Step 12	10010000100	$\boxed{0}$	111011111 ...
	10010000100	0	011011111 ...
	10010000100	0	001011111 ...
	10010000100	0	000011111 ...




r1
r2
r1
r2
r2
r1
r1
r2
r2
r1
r2
0
1
2
3
4
5
6
7
8
9
10
11


(r1lt.x, r1lt, y)
r1
(r1rb.x, r1rb.y)
(r2lt.x, r2lt.y)
r2
r2rb.x, r2rb.y)
(r2lt.x, r2lt.y)
r2
(r2rb.x, r2rb.y)
(r1lt.x, r1lt.y)
r1
(r1rb.x, r1rb.y)
0
1
2
3
4
5
6
7
8
9
10
11

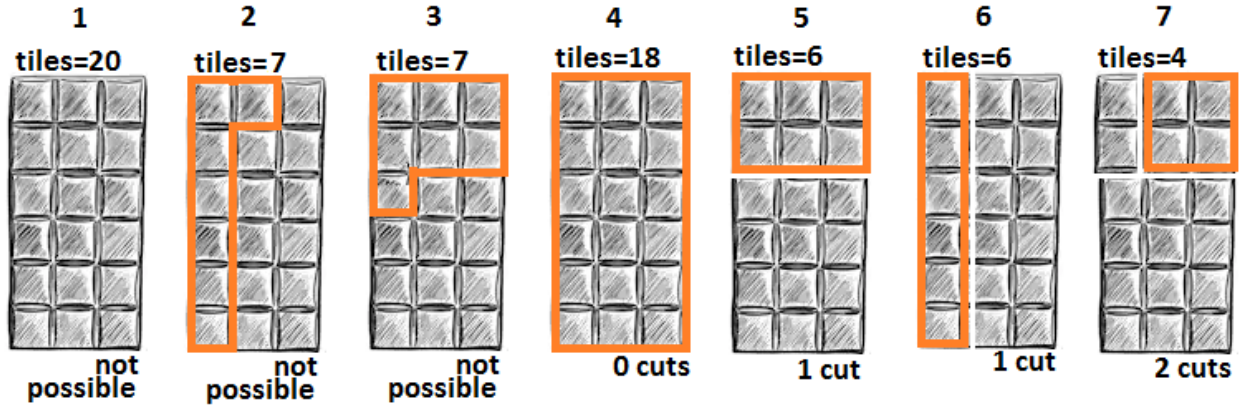
4145775
771467

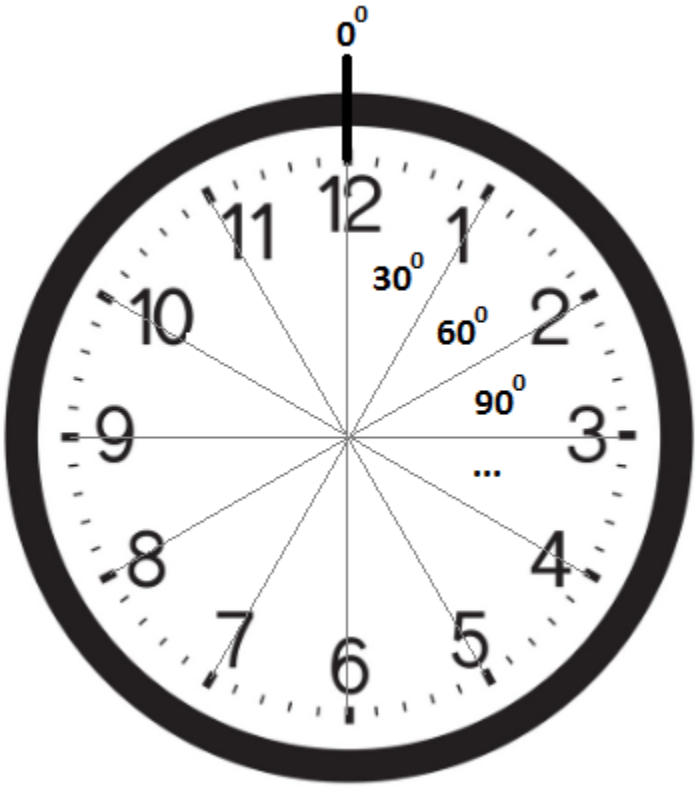
*

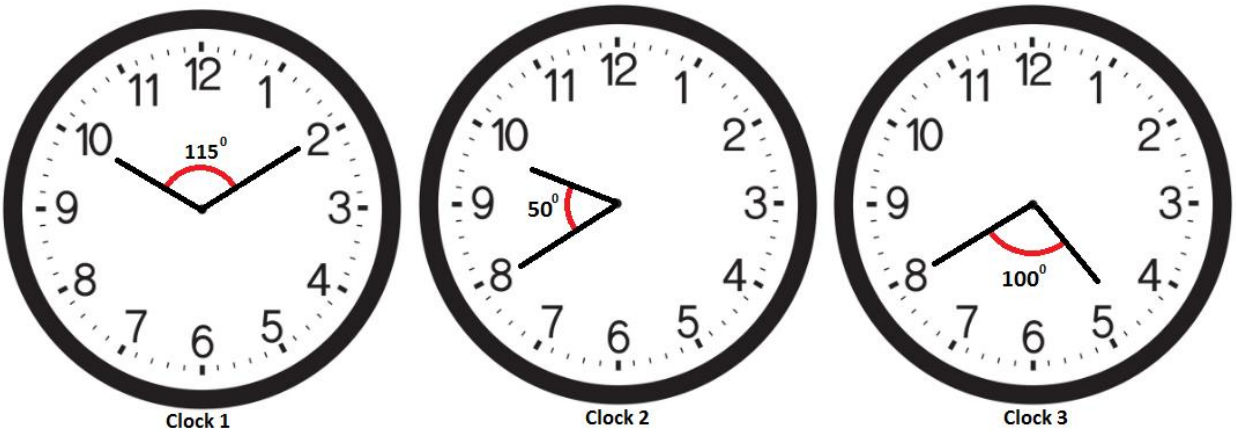
29020425 = 4145775 x 7
248746500 = 4145775 x 60
1658310000 = 4145775 x 400
4145775000 = 4145775 x 1000
290204250000 = 4145775 x 70000
2902042500000 = 4145775 x 700000

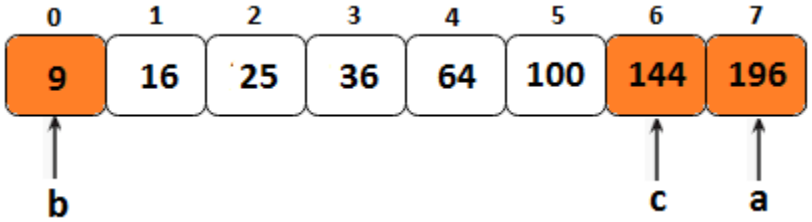
+

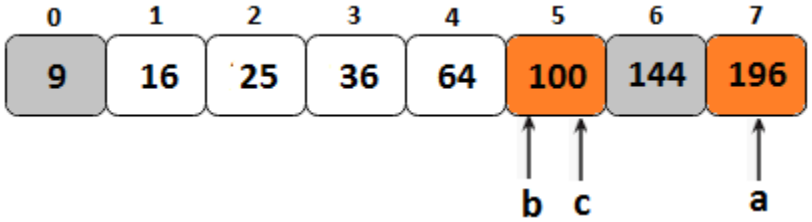
3198328601925

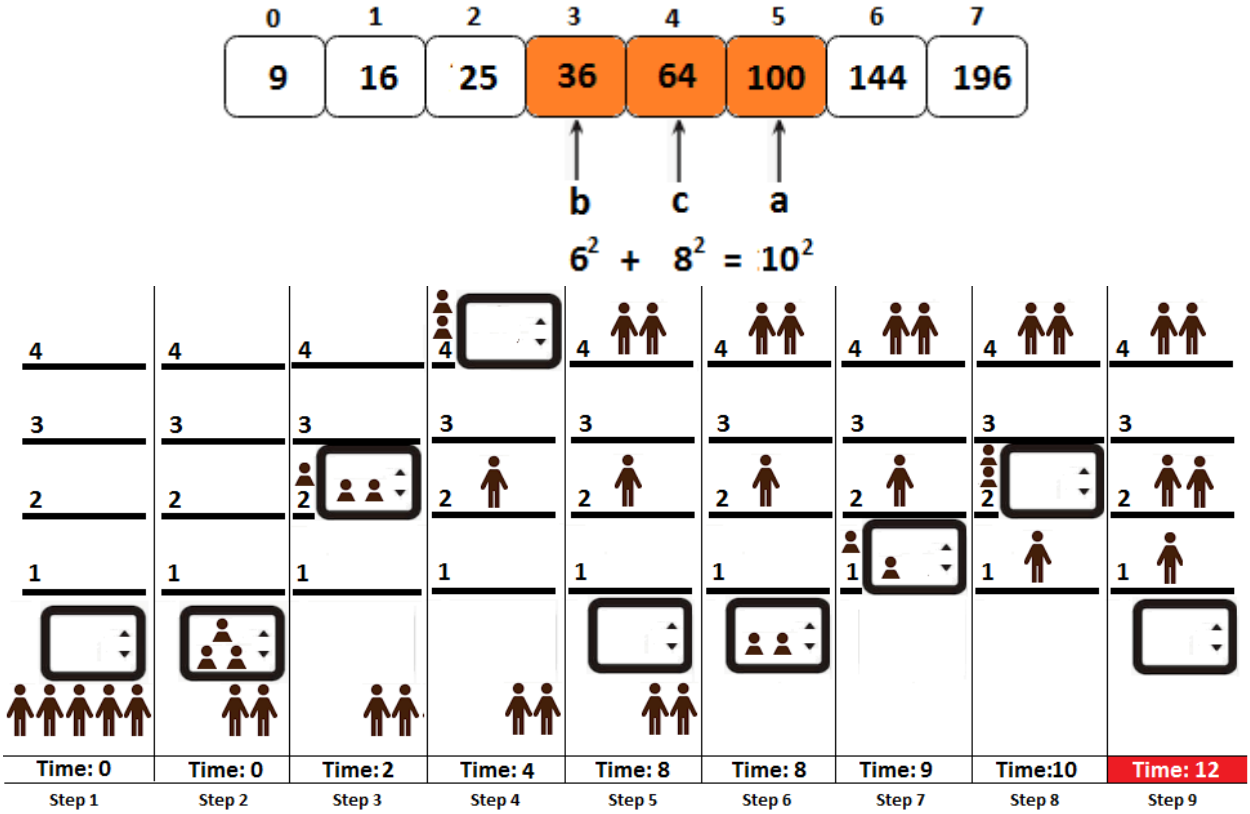




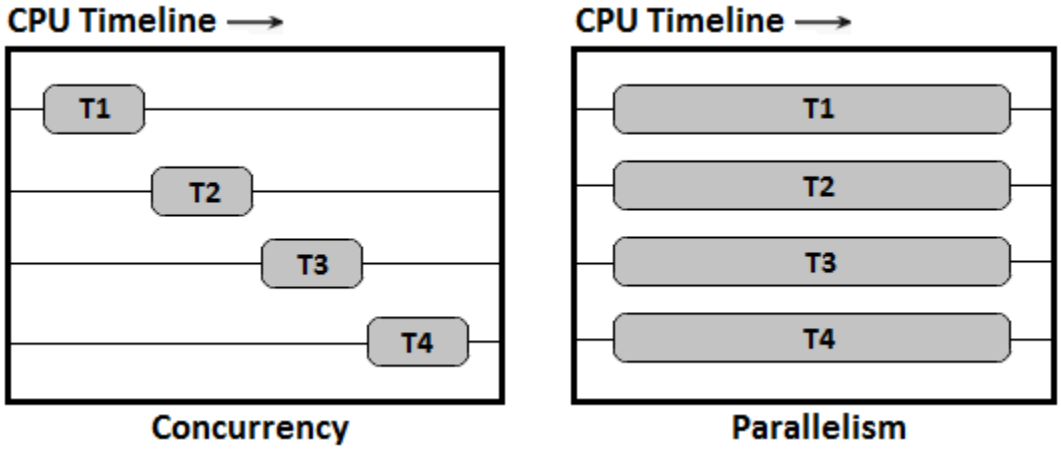

115⁰
50⁰
100⁰
Clock 1
Clock 2
Clock 3

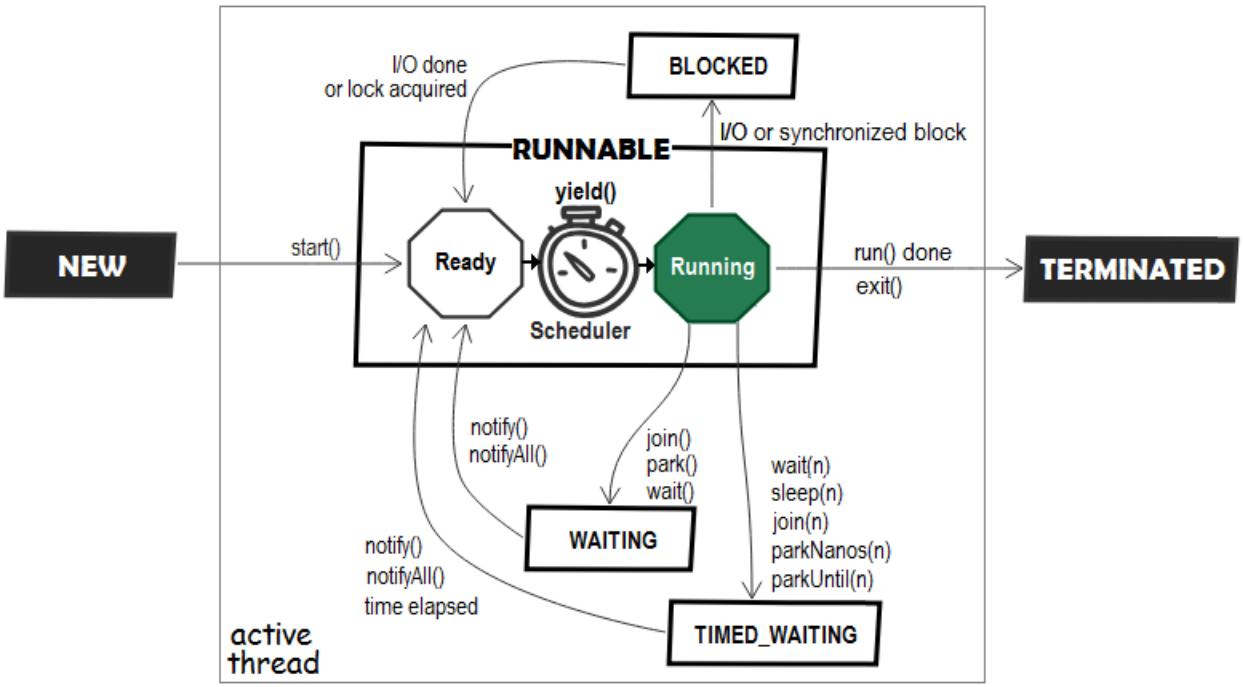

0 1 2 3 4 5 6 7
9 16 25 36 64 100 144 196
b c a

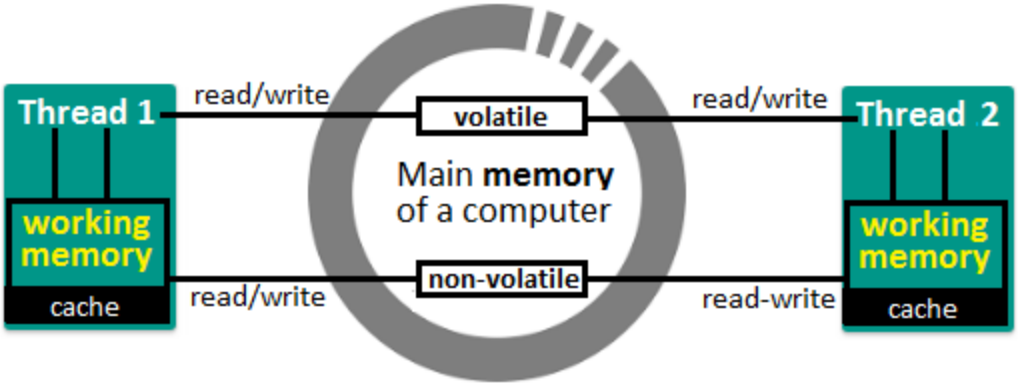

0 1 2 3 4 5 6 7
9 16 25 36 64 100 144 196
b c a

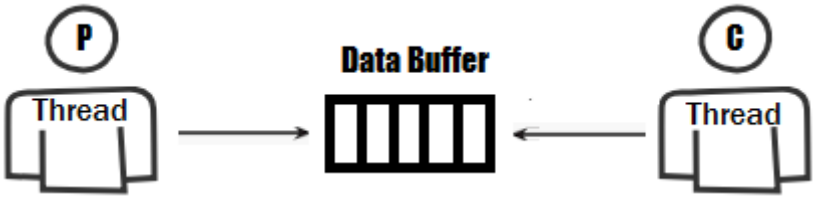

0 1 2 3 4 5 6 7
9 16 25 36 64 100 144 196
b c a
$6^2 + 8^2 = 10^2$
Time: 0
Time: 0
Time: 2
Time: 4
Time: 8
Time: 8
Time: 9
Time:10
Time: 12
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Step 7
Step 8
Step 9

Chapter 16: Concurrency

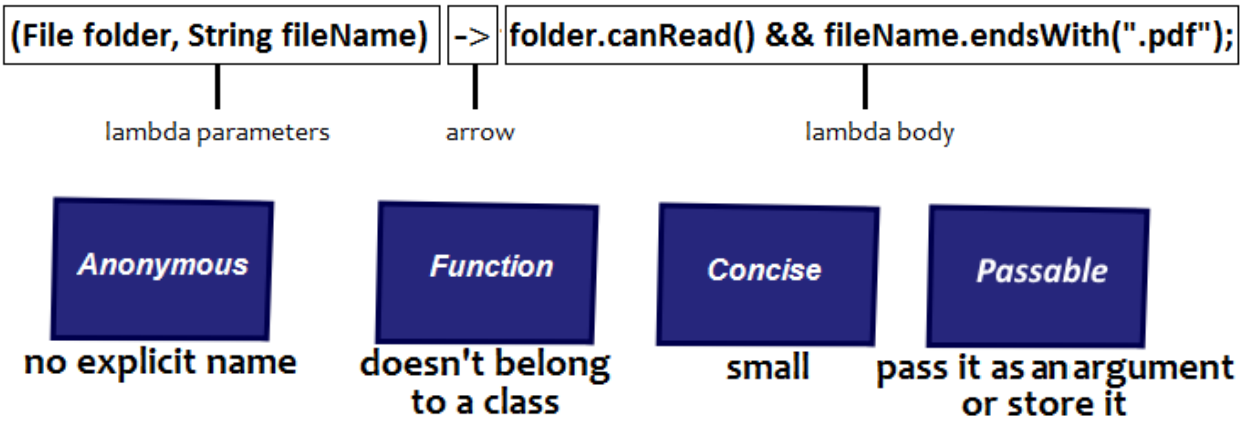




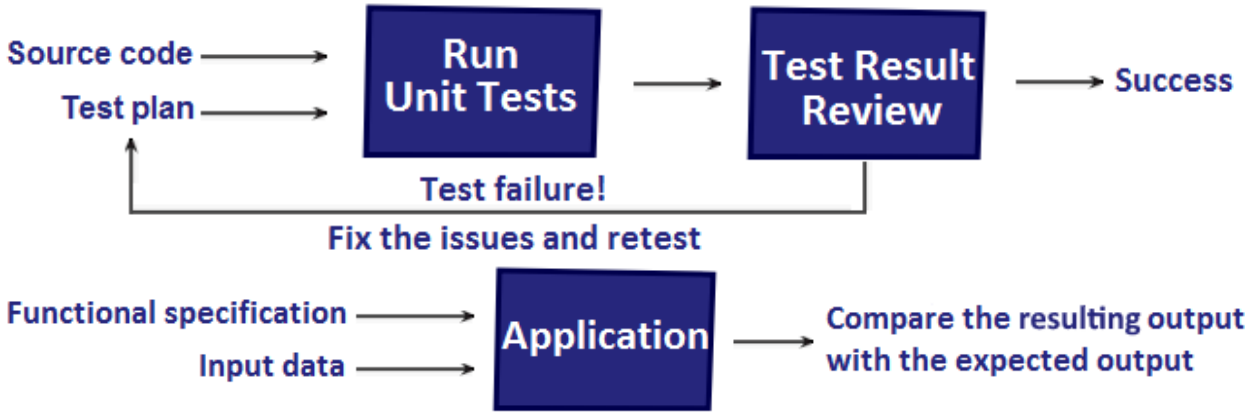

Thread 1
read/write
volatile
read/write
Thread 2
working memory
Main memory of a computer
working memory
cache
read/write
non-volatile
read-write
cache

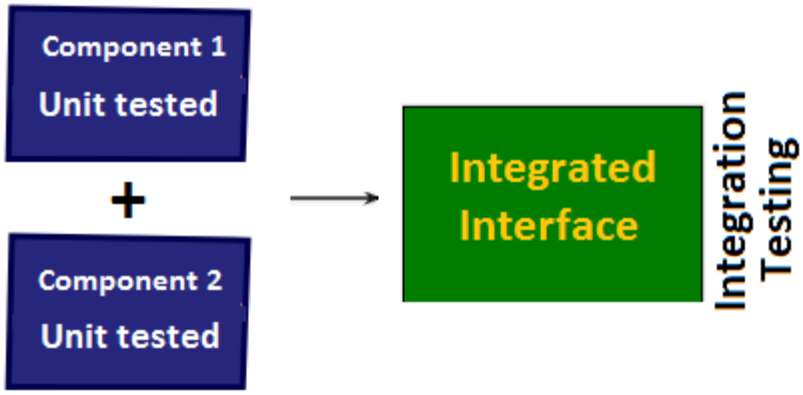

P
Data Buffer
C
Thread
Thread

Chapter 17: Functional-Style Programming



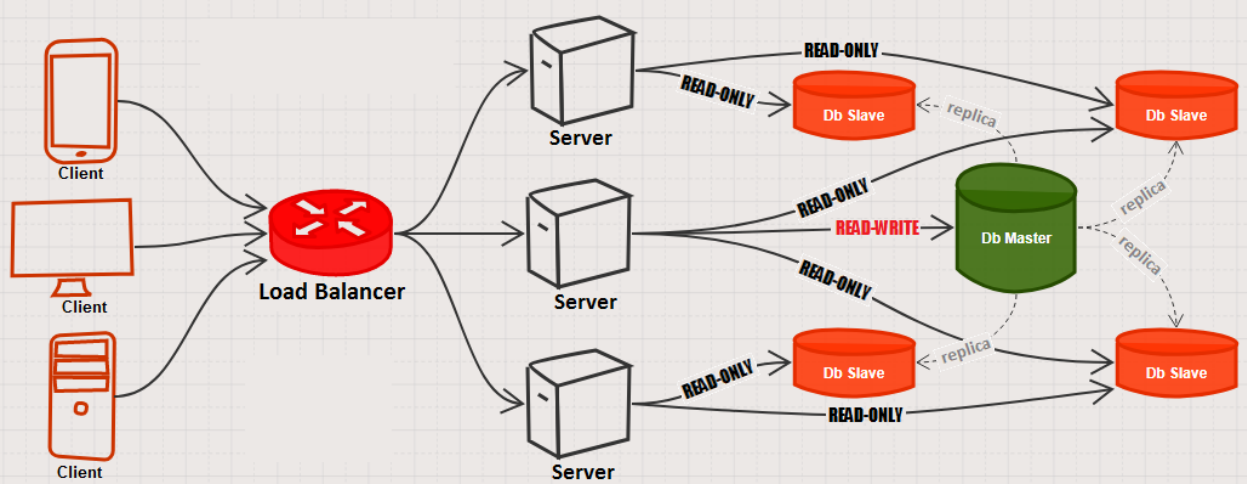
Chapter 18: Unit Testing

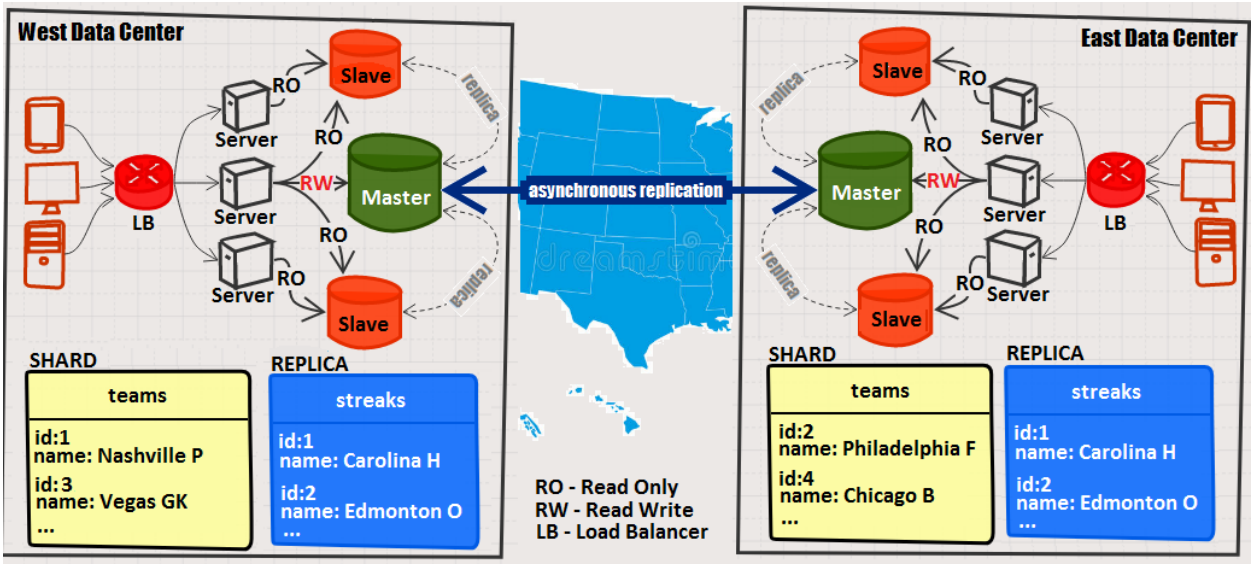


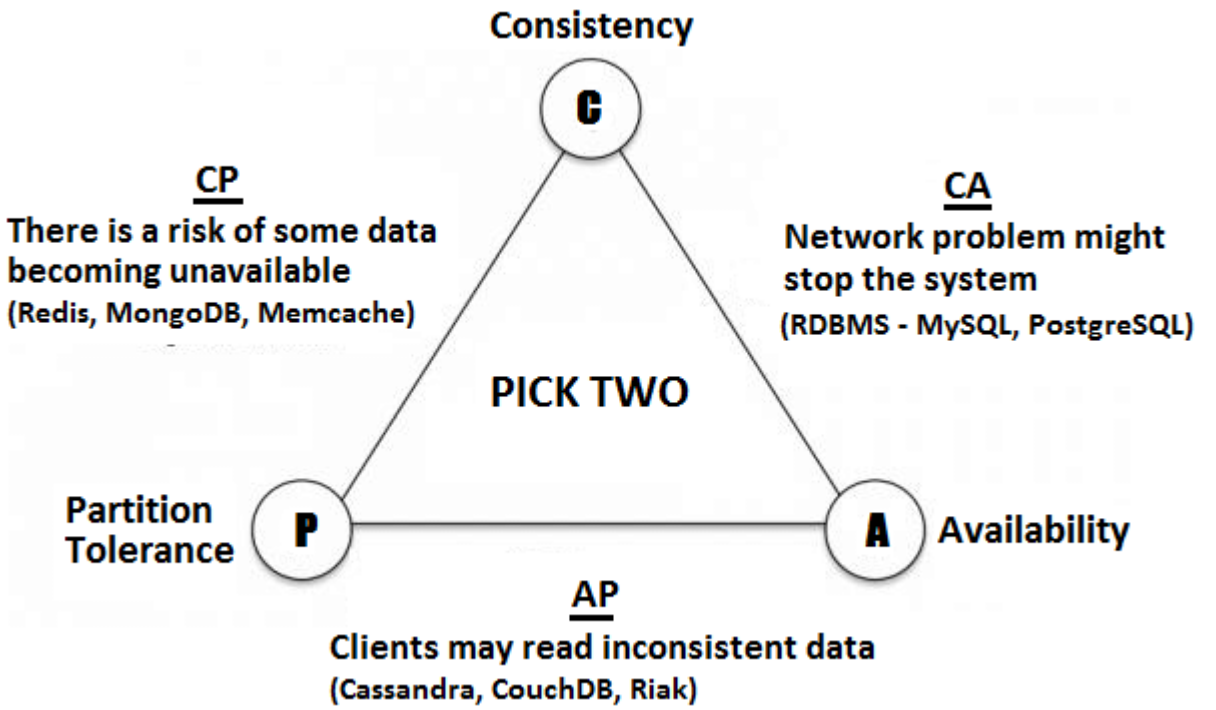


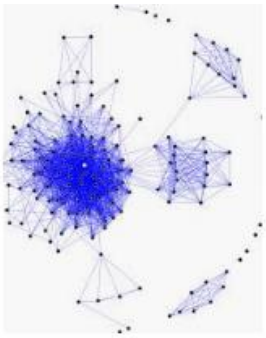
Unit test	Integration test
Typically, useful to developers.	Useful to QA, DevOps, and Help Desk as well
Results depend only on Java code.	Results may depend on external systems.
Fairly easy to write.	May be quite complicated to set up
Units are tested in isolation.	One or more components are tested.
Can use mocking for dependencies.	Mocking should be avoided.
Only test the implementation of code.	Test the implementation of the components and the interconnection behavior between them
Uses JUnit/TestNG.	Uses real environments via tools such as Arquillian and DbUnit.
A failed test is a regression problem.	A failed test can be caused by changes in the environment
Such tests shouldn't take too long to run.	Such tests can take quite a long time (for example, 1 hour)

Chapter 19: System Scalability

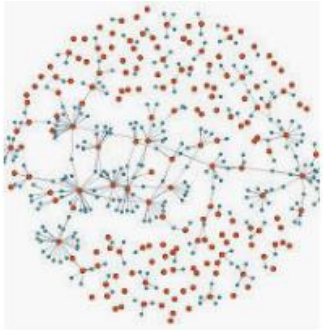








Social graph - Wikipedia
en.wikipedia.org



Social network graph - Adobe Su...
community.adobe.com

Discovering Organization...
orgnet.com

Social network directed graph...
stackoverflow.com