Design And Analysis of Algorithms

Homework 2

Outline

Question 1.	**Basic**
Question 2.	**Basic**
Question 3.	**Moderate**
Question 4.	**Moderate**
Question 5.	**Challenging**

Question1

- Our genius friend, John, has invented the algorithm for sorting the array $A[i..j]$:

```
JohnSort(A, i, j)
   Set l = j − i + 1, k = l % 3, m = (l − k)/3;
   if (k != 0) {
        Find smallest k items;
        Swap them with items in A[i..i + k − 1]
        in increasing order ;
   }
   if (m == 0)    return;

   /* Sort remaining 3m items by recursion */
   JohnSort(A, i+k, j−m)   // Sort first 2m items
   JohnSort(A, i + k+m, j) // Sort last 2m items
   JohnSort(A, i+k, j−m)   // Sort first 2m items
```

Question1

(a) Show that the above algorithm is correct.

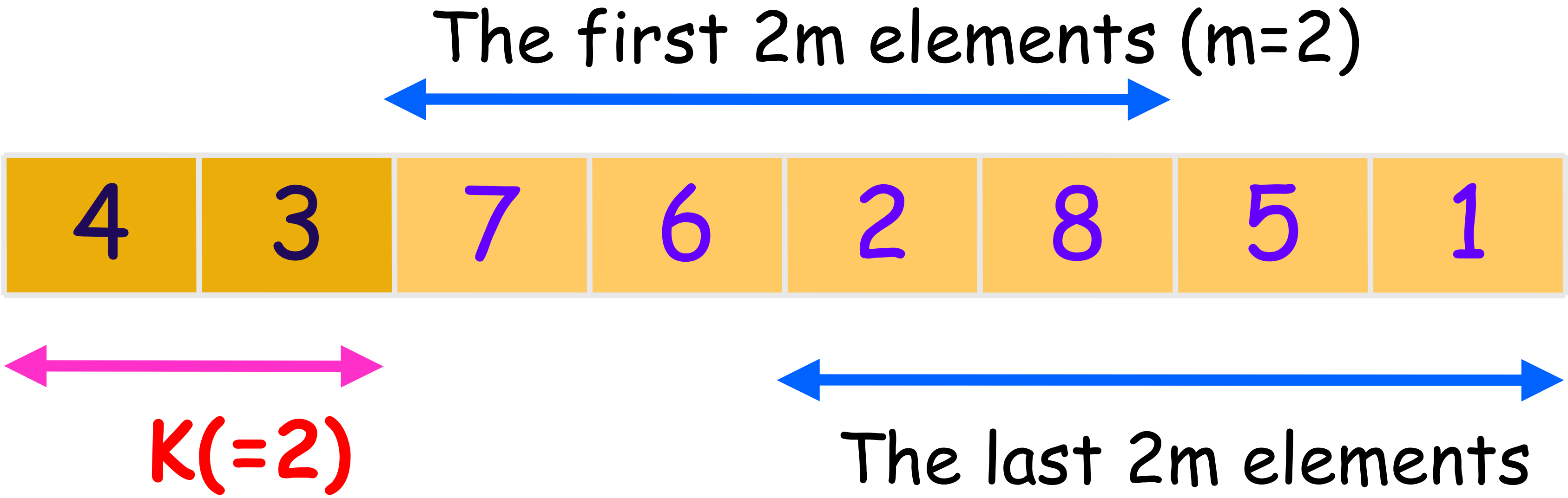
(b) Give a recurrence for the worst-case running time of JohnSort.

(c) Obtain a tight asymptotic (Θ-notation) bound on the worst-case running time.

  - How does it compare with the worst-case running time of insertion sort and merge sort?

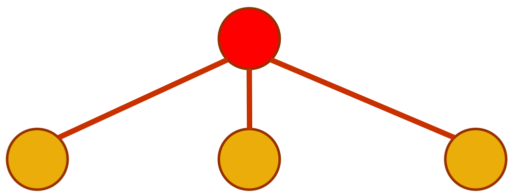
Question1

An example:

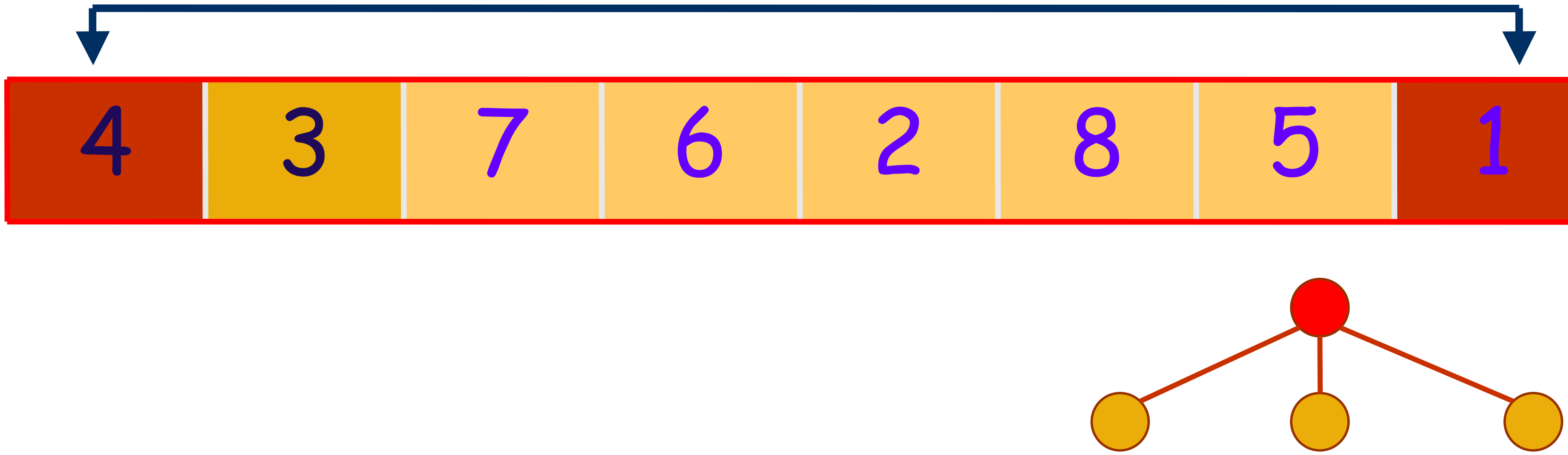


Question1

4	3	7	6	2	8	5	1

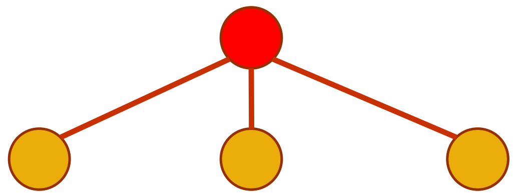


Question1

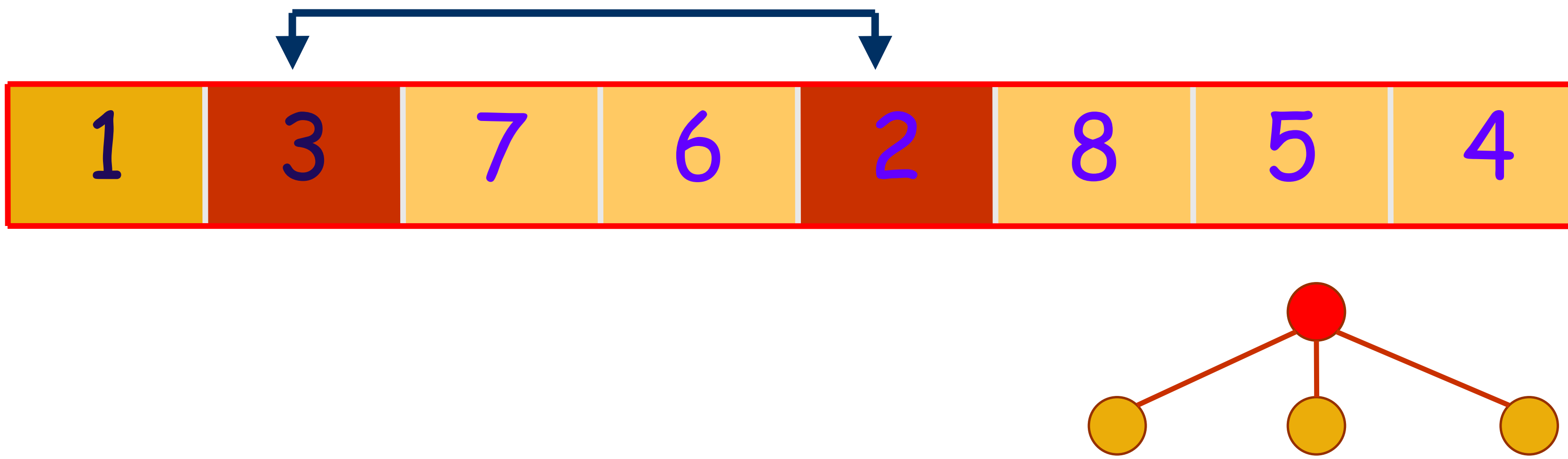


Question1

1	3	7	6	2	8	5	4

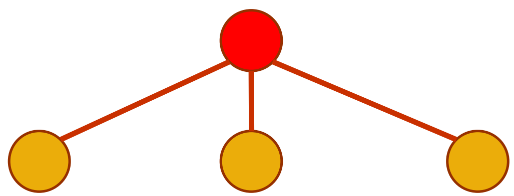


Question1

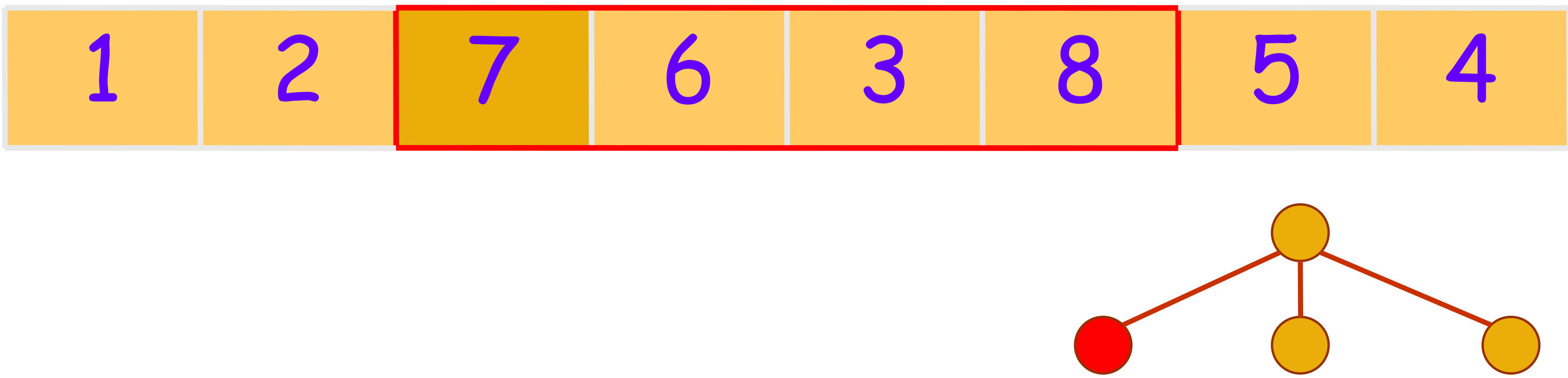


Question1

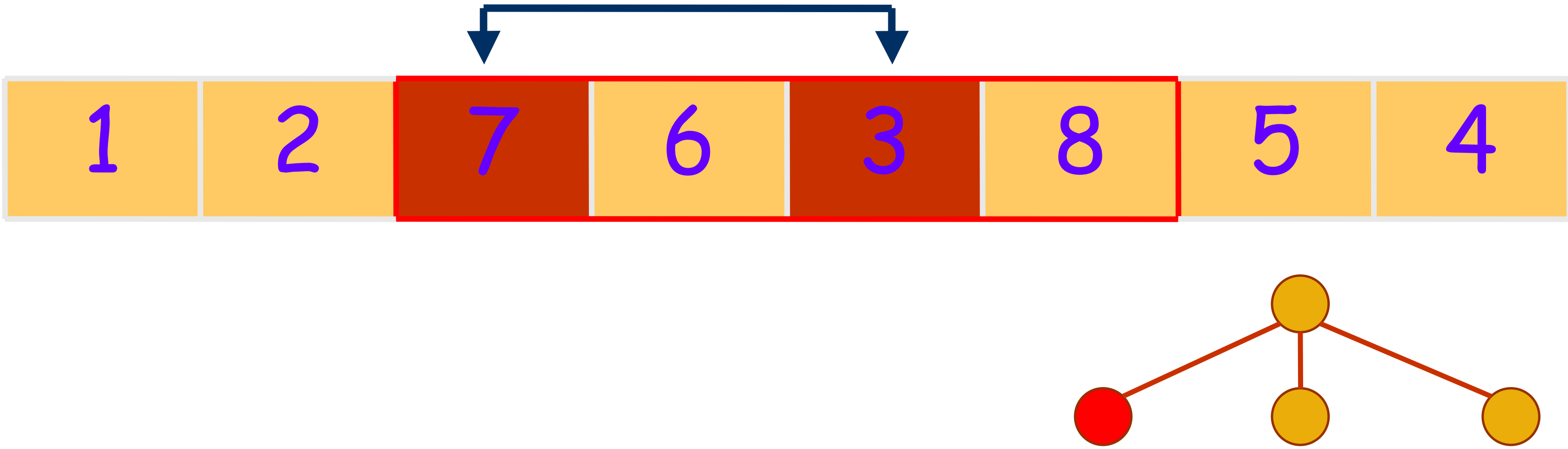
1	2	7	6	3	8	5	4



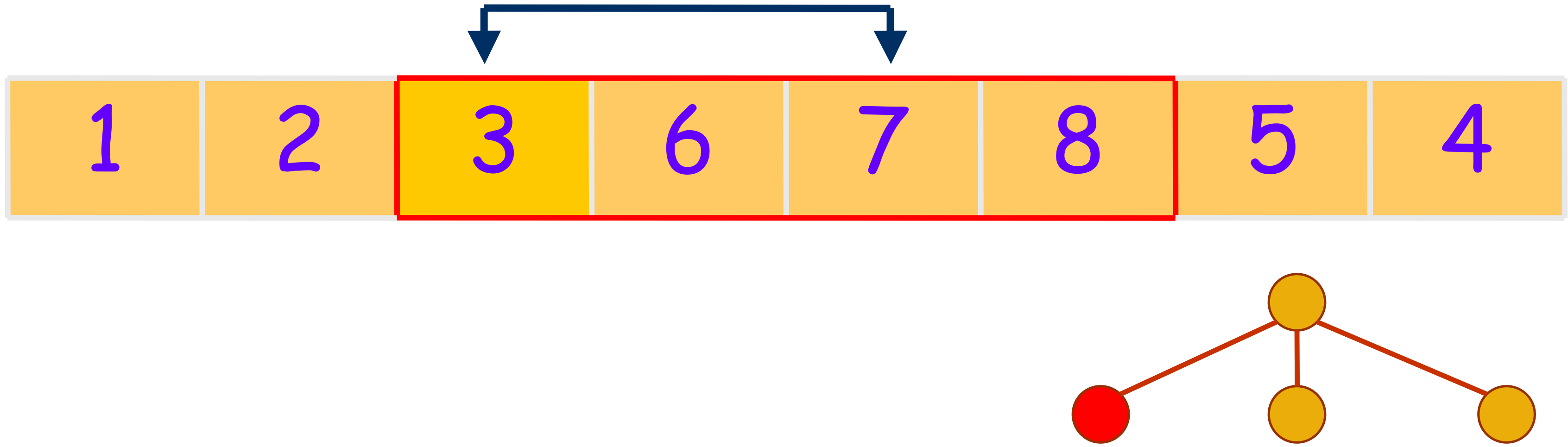
Question1



Question1

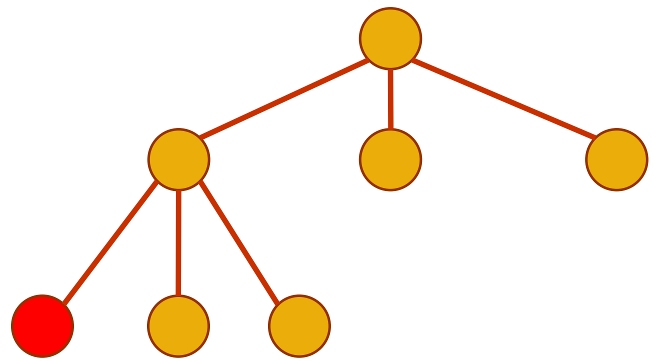


Question1

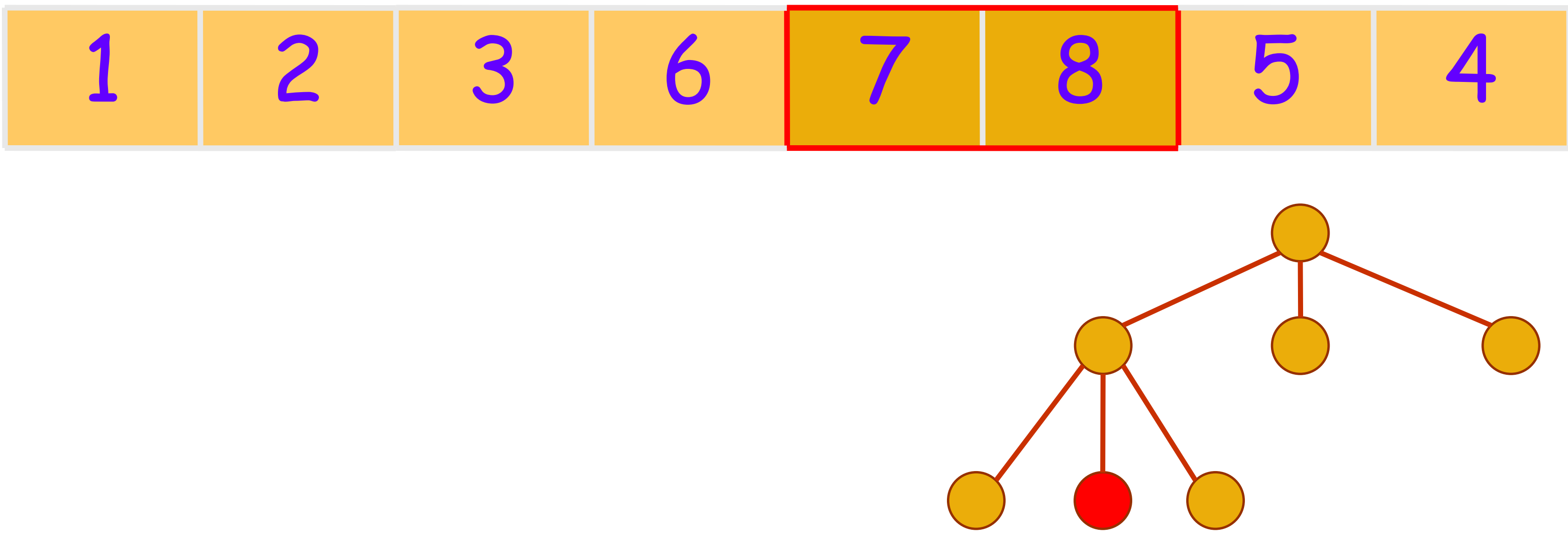


Question1

1	2	3	6	7	8	5	4

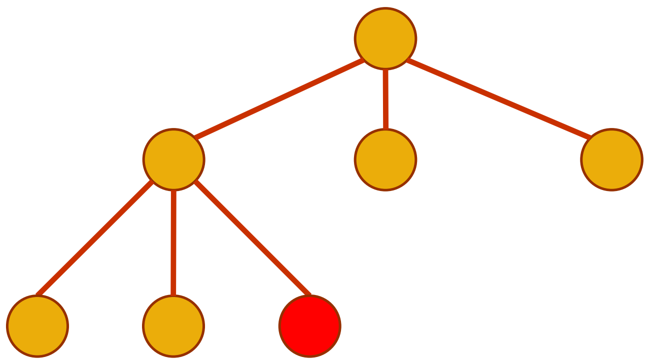


Question1



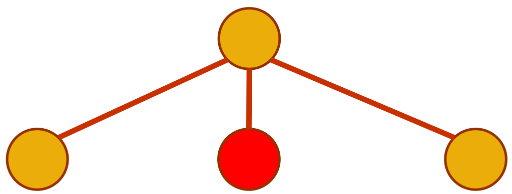
Question1

1	2	3	6	7	8	5	4

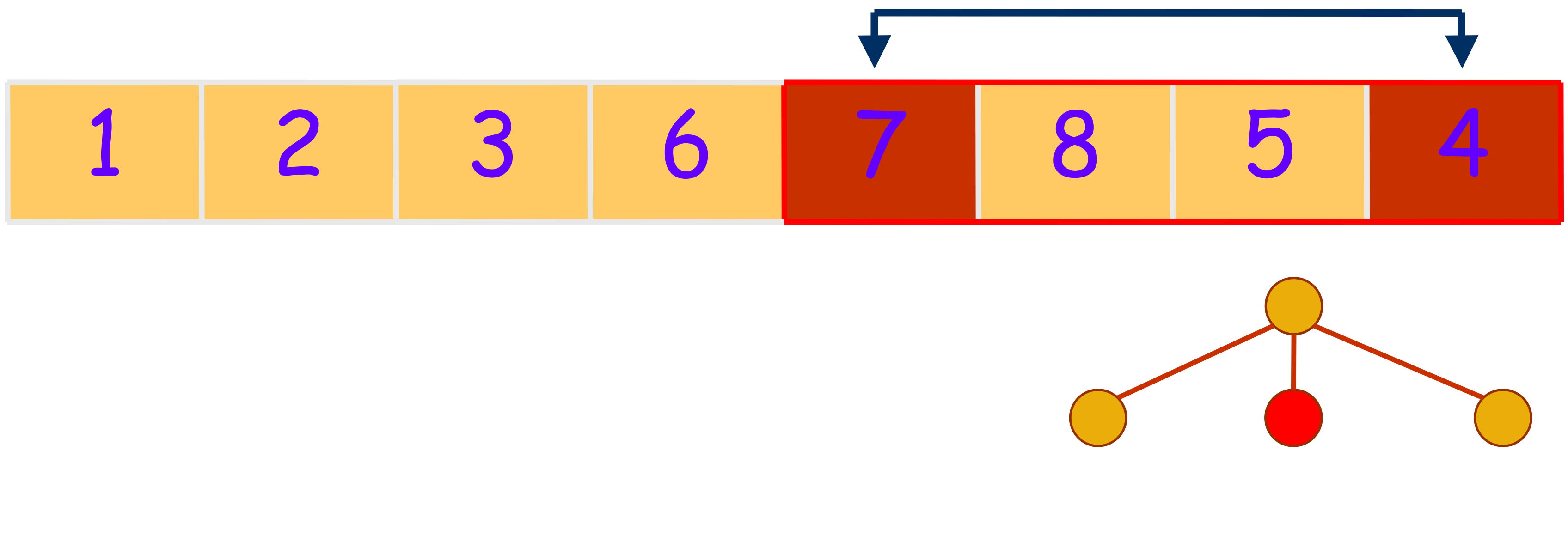


Question1

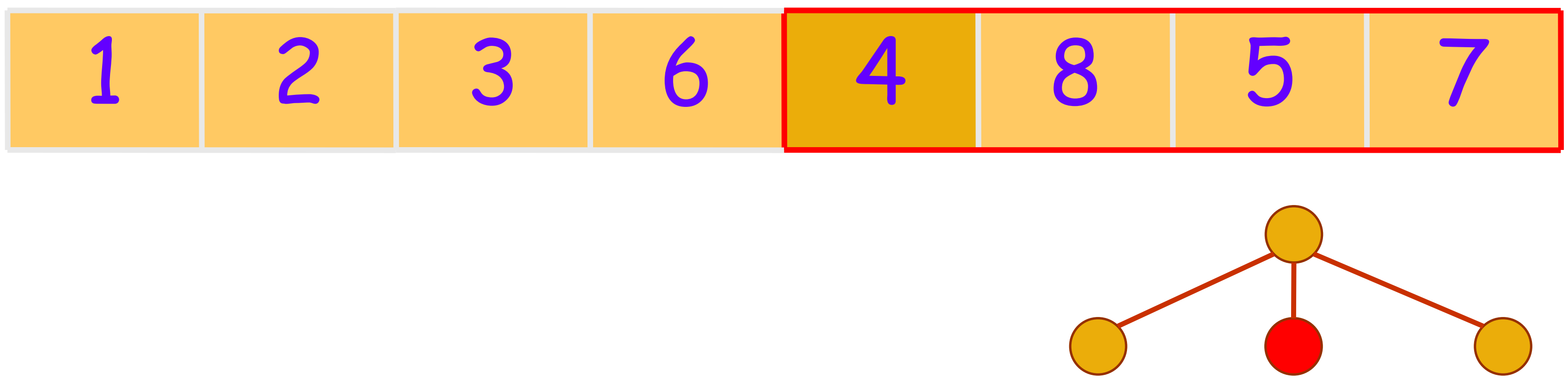
1	2	3	6	7	8	5	4



Question1

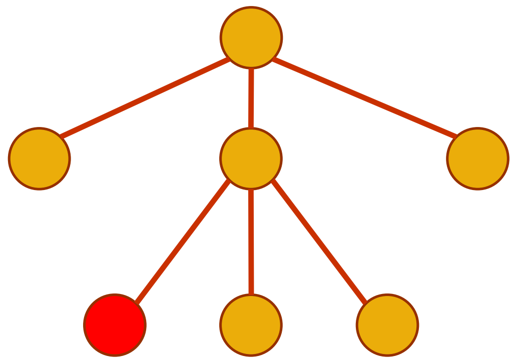


Question1

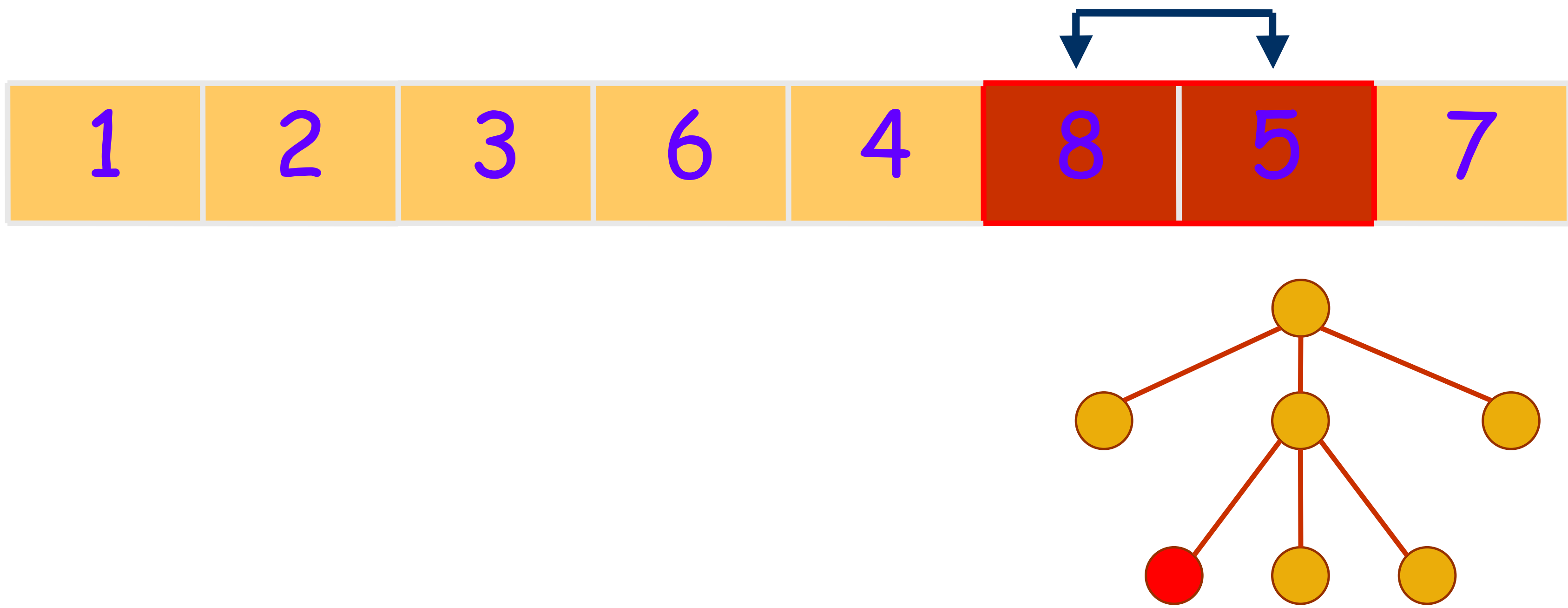


Question1

1	2	3	6	4	8	5	7

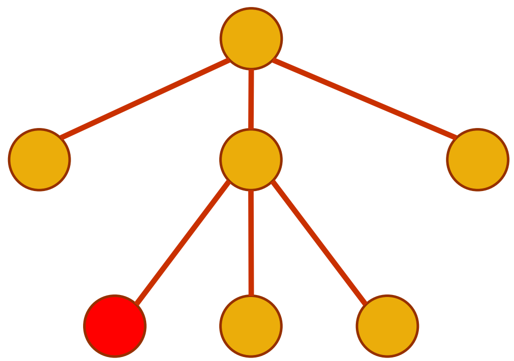


Question1



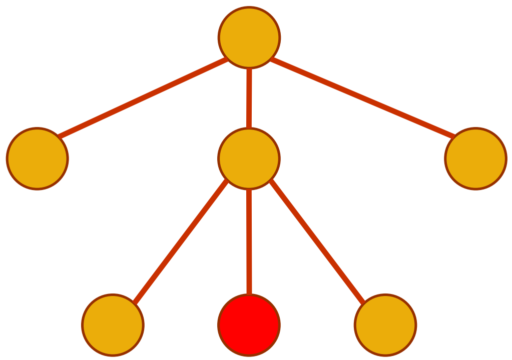
Question1

1	2	3	6	4	5	8	7

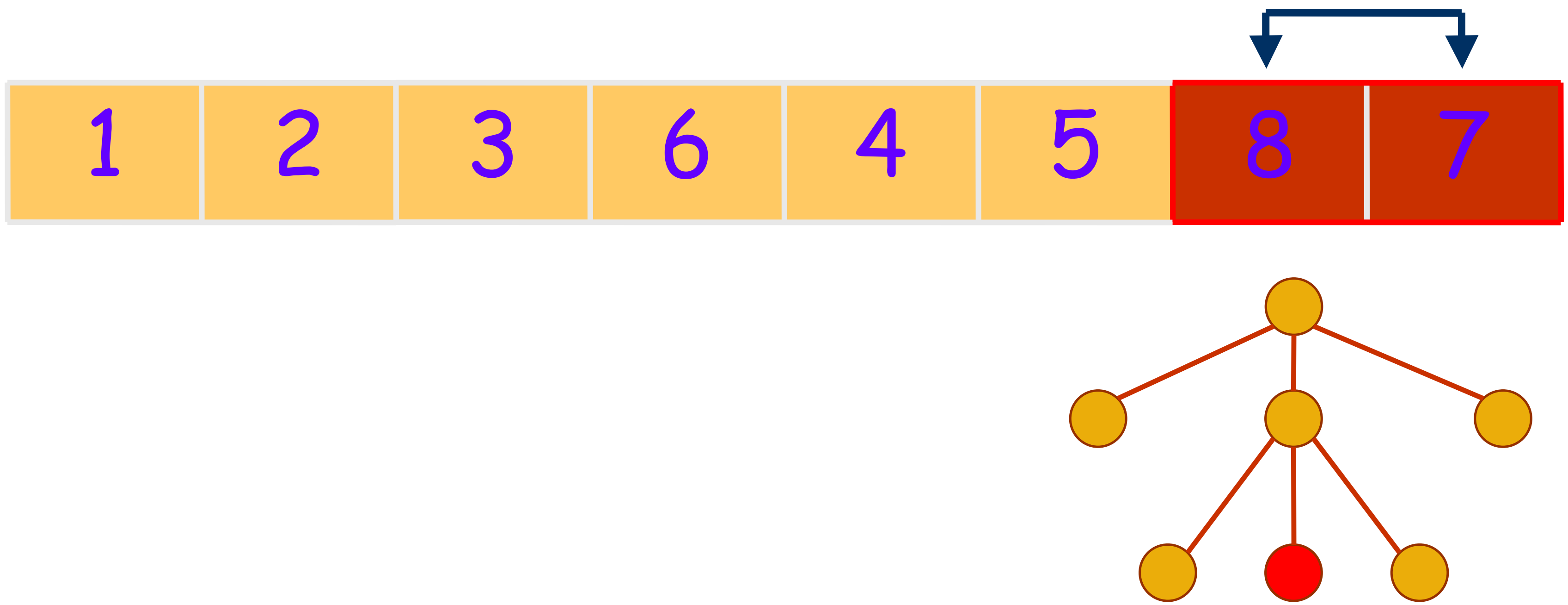


Question1

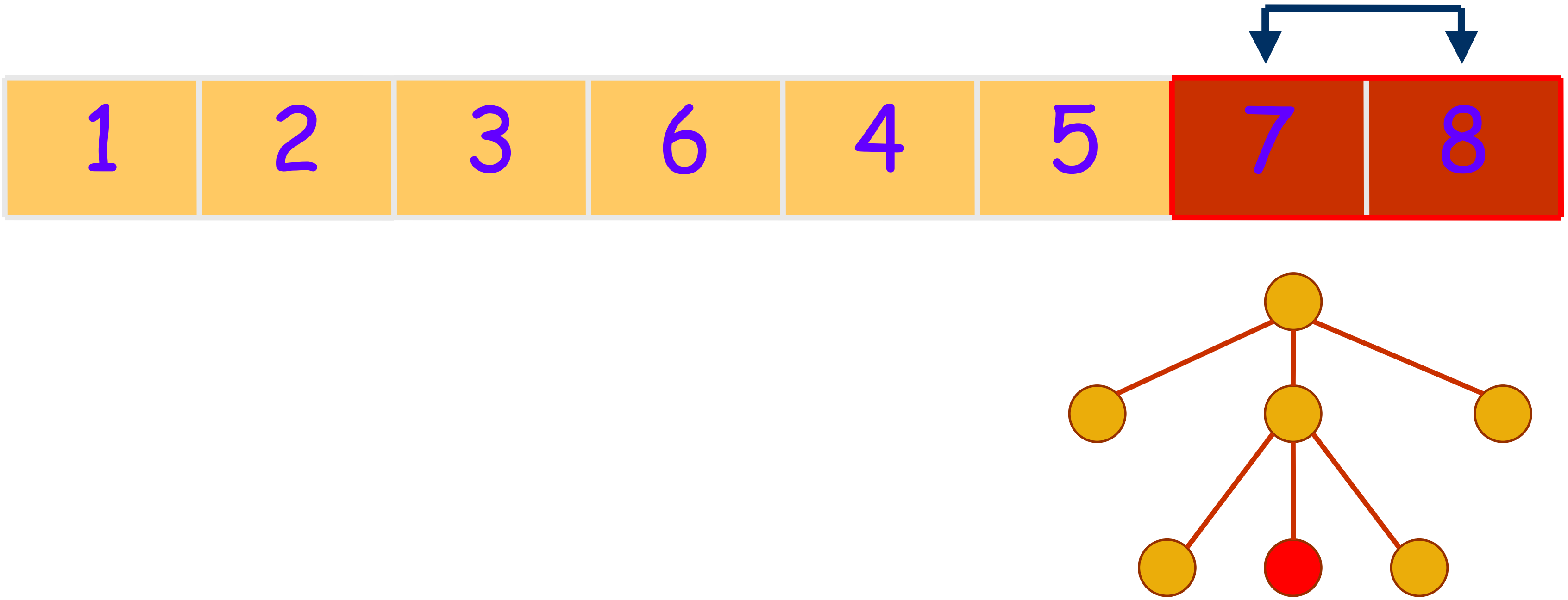
1	2	3	6	4	5	8	7



Question1

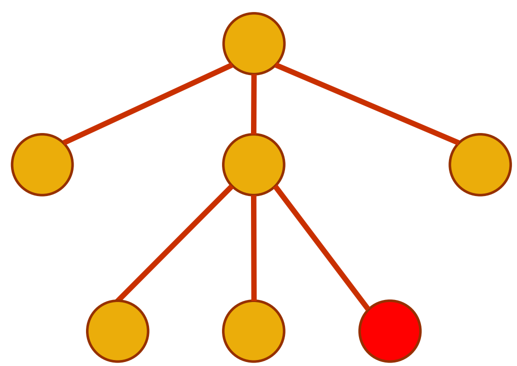


Question1



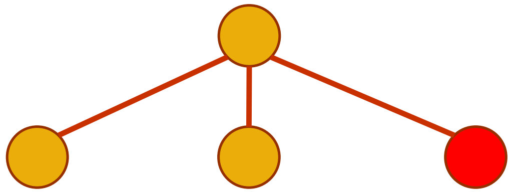
Question1

1	2	3	6	4	5	7	8



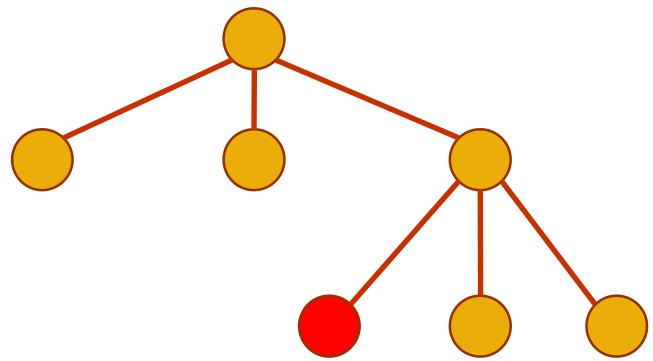
Question1

1	2	3	6	4	5	7	8

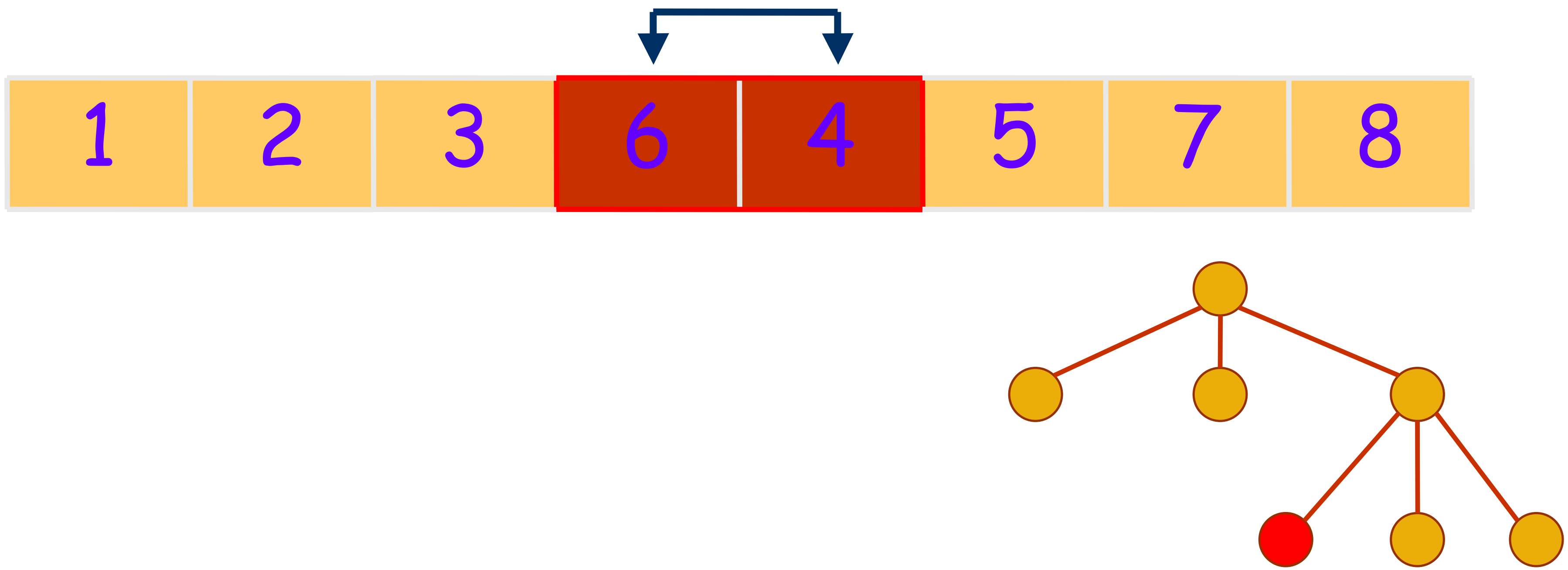


Question1

1	2	3	6	4	5	7	8

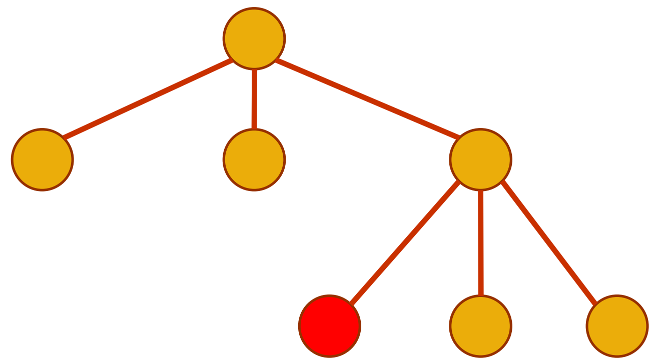


Question1



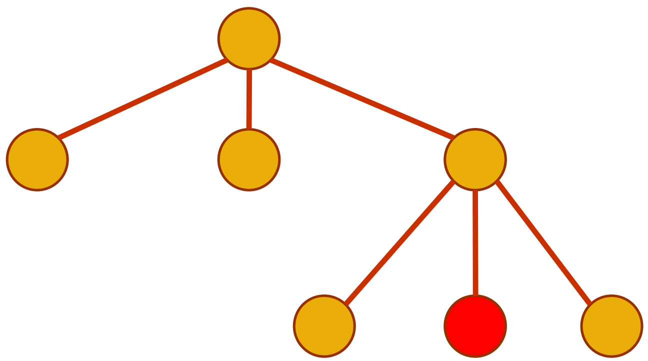
Question1

1	2	3	4	6	5	7	8

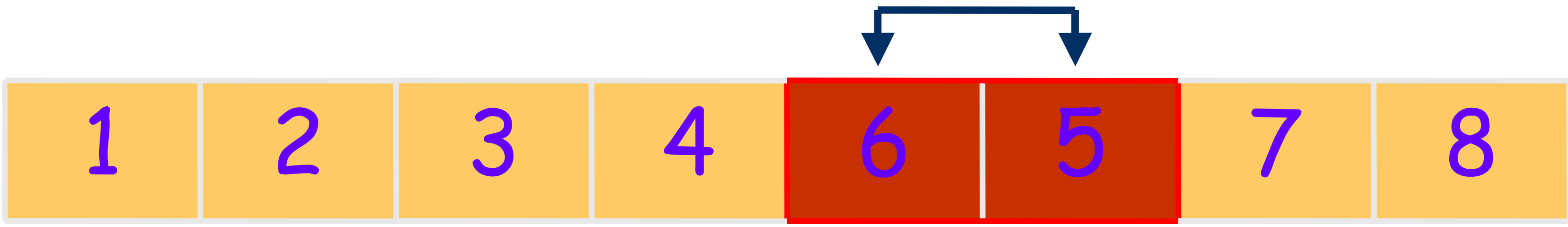


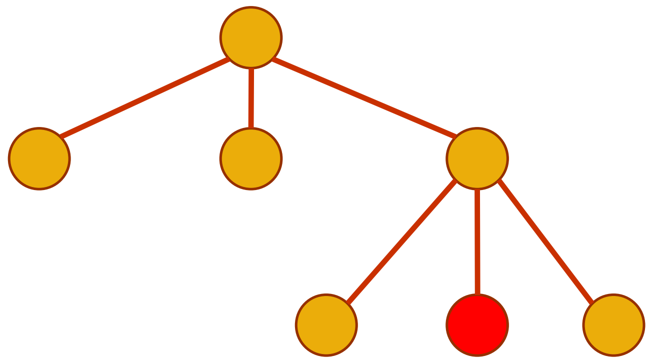
Question1

1	2	3	4	6	5	7	8

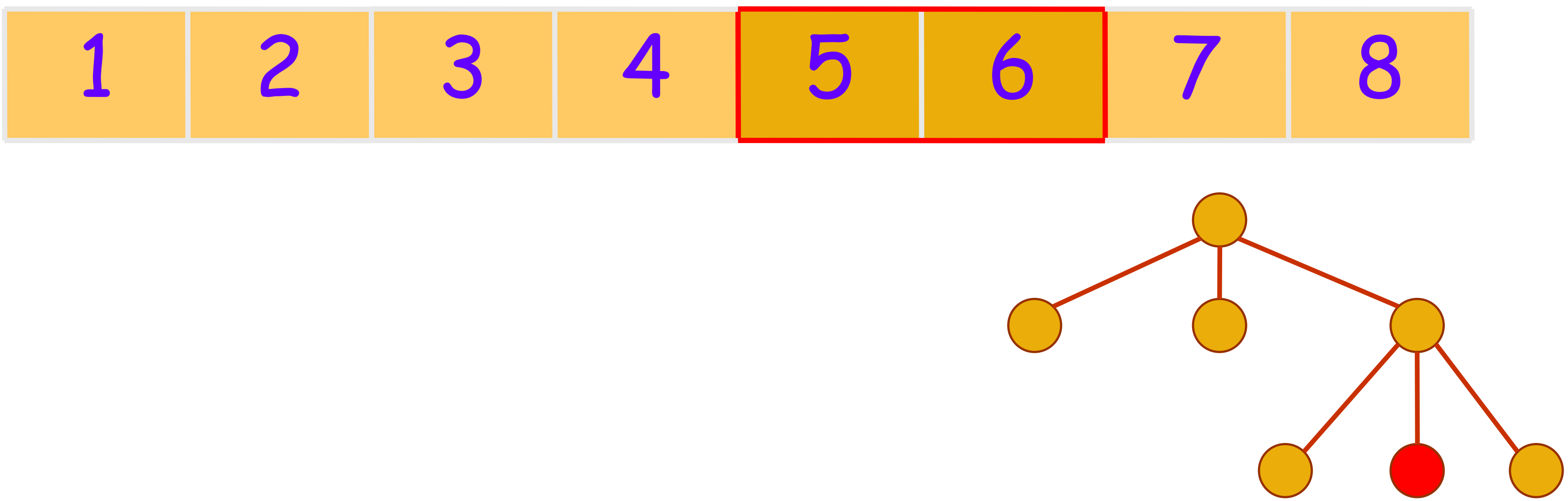


Question1



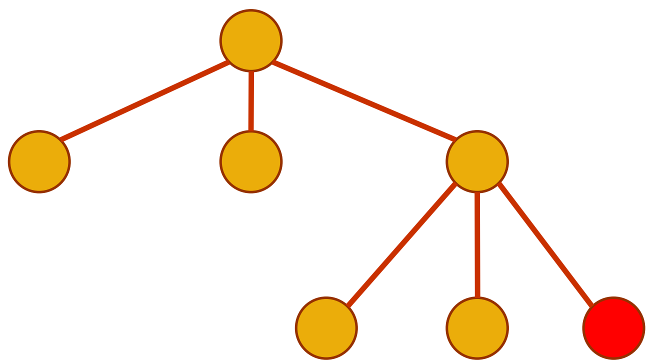


Question1



Question1

1	2	3	4	5	6	7	8



Question1

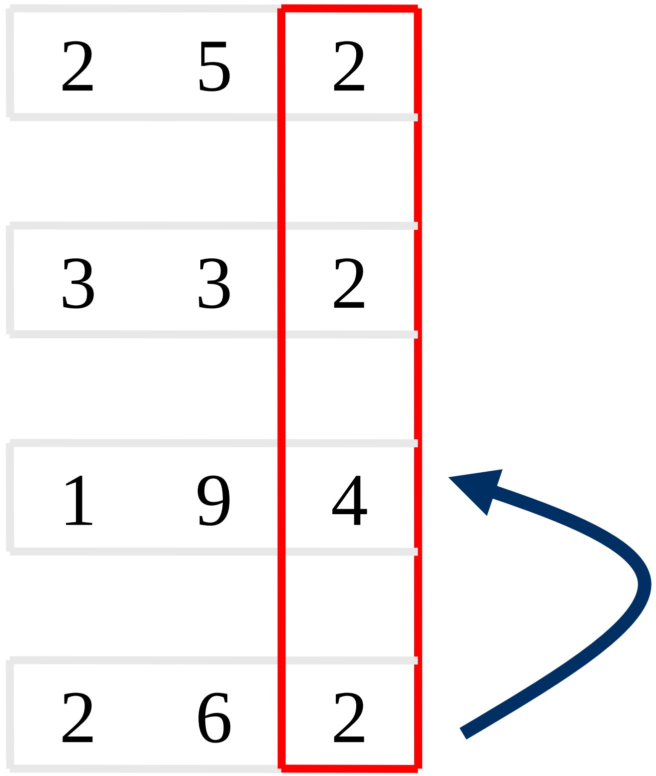
1	2	3	4	5	6	7	8

Question1

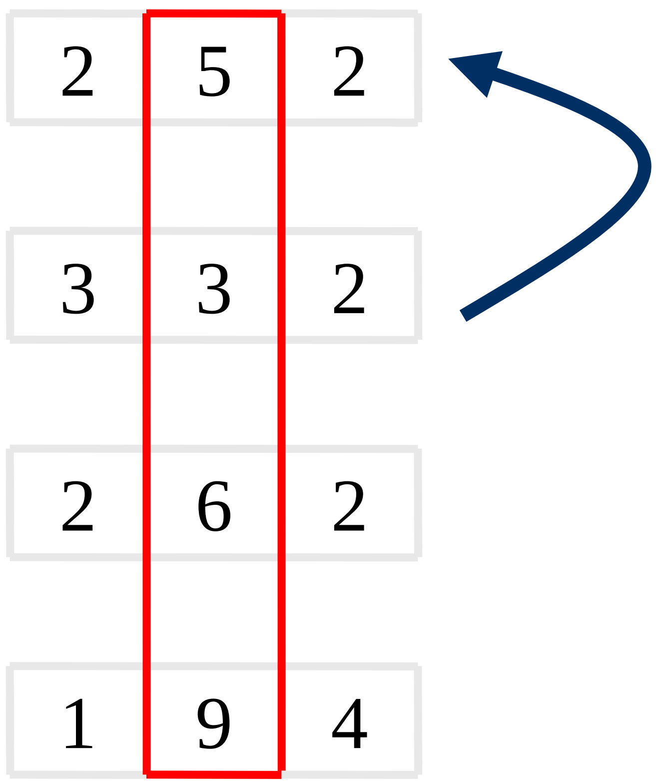
- In (b), only need to give the recurrence
- In (c), use Master Theorem to bound the time complexity

Question2

- What's RadixSort?
- An example:



Question2



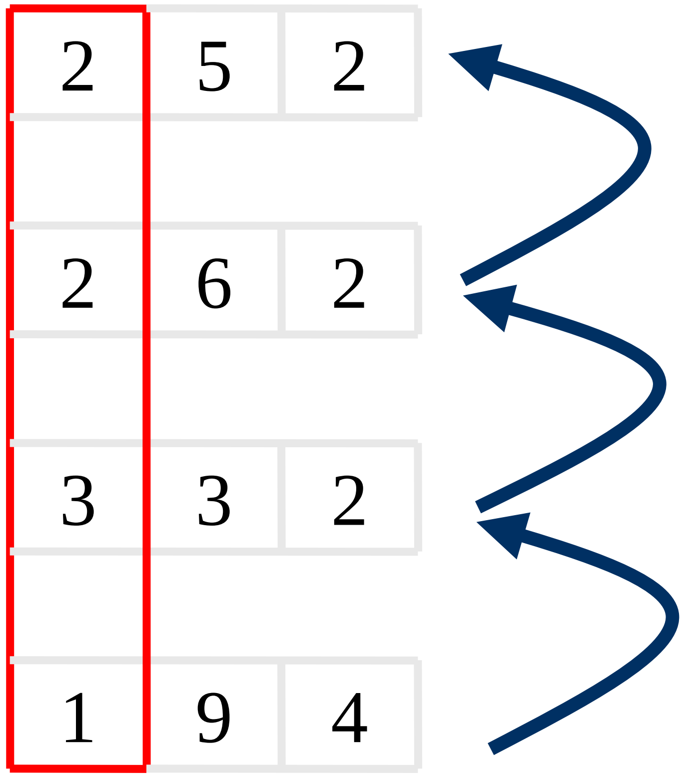
Question2

3	3	2
2	5	2
2	6	2
1	9	4

Question2

2	5	2
3	3	2
2	6	2
1	9	4

Question2



Question2

1	9	4

2	5	2

2	6	2

3	3	2

Question2

Illustrate the operation of RadixSort on the following list of English words:

COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB, BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX

Show clearly how RadixSort works on those data

Question3

- Given an array $A[1..k]$ of k strings, with each string representing an integer. Totally there are n digits in these strings.

Show how to sort them in $O(n)$ time

Question3

Example:

If the input array is

{ "235", "8", "17","652",
"490", "231562955",
"940", "2"},

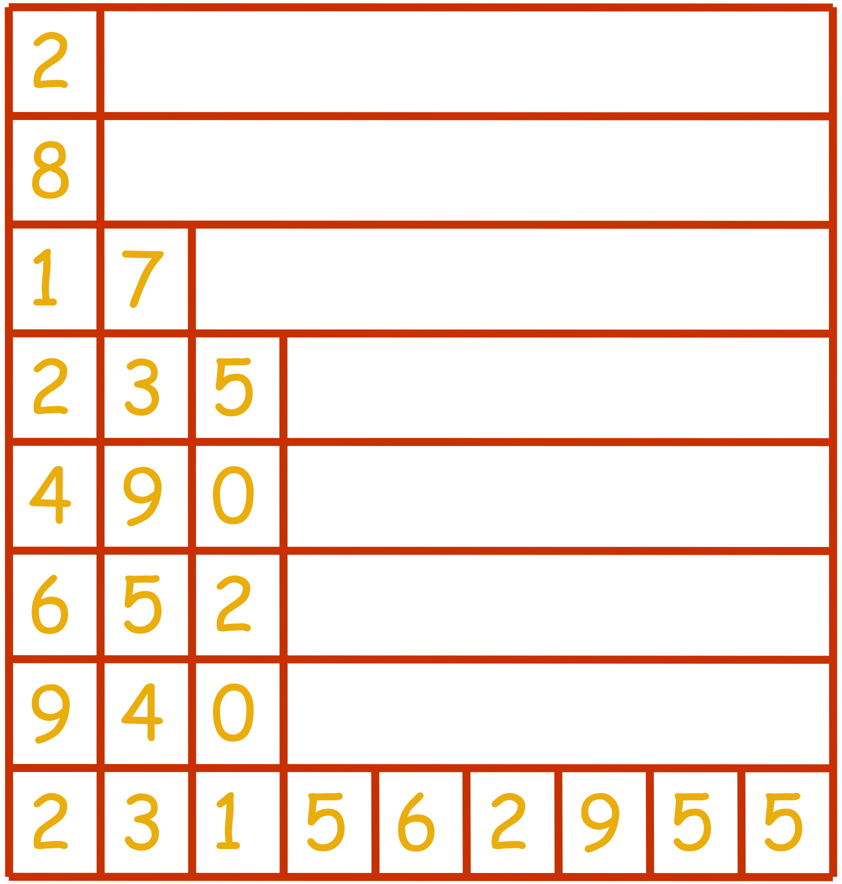
then n = 25

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

Question3

After sorting, we should obtain

{ "2", "8", "17", "235", "490", "652", "940", "231562955"}



Question3

- A simple RadixSort is not capable of solving this problem

- Why?

Question4

1	3	5	8
2	5	6	13
4	7	12	14
6	9	15	19

What's so special about this matrix?

Question4

Young Tableau is an $m \times n$ matrix with

(i) each row is sorted

(ii) each column is sorted order

Some entries = ∞

→ Indicate an empty entry

Young Tableau can hold $\leq mn$ finite #s

Question4

Young Tableau

1	4	5	9
3	5	6	13
4	6	12	14
7	9	10	16

Is this a Young Tableau?

Question4

Young Tableau

1	4	5	9
3	5	6	13
4	6	12	14
7	9	10	16

No !

Question4

(a) Draw three 4×4 Young tableaux containing exactly the elements $\{9, 16, 3, 2, 4, 8, 5, 14, 12\}$.

(b) Given an $m \times n$ Young tableau Y

-- Show Y is empty if $Y[1, 1] = \infty$

-- Show Y is full if $Y[m, n] < \infty$

Question4

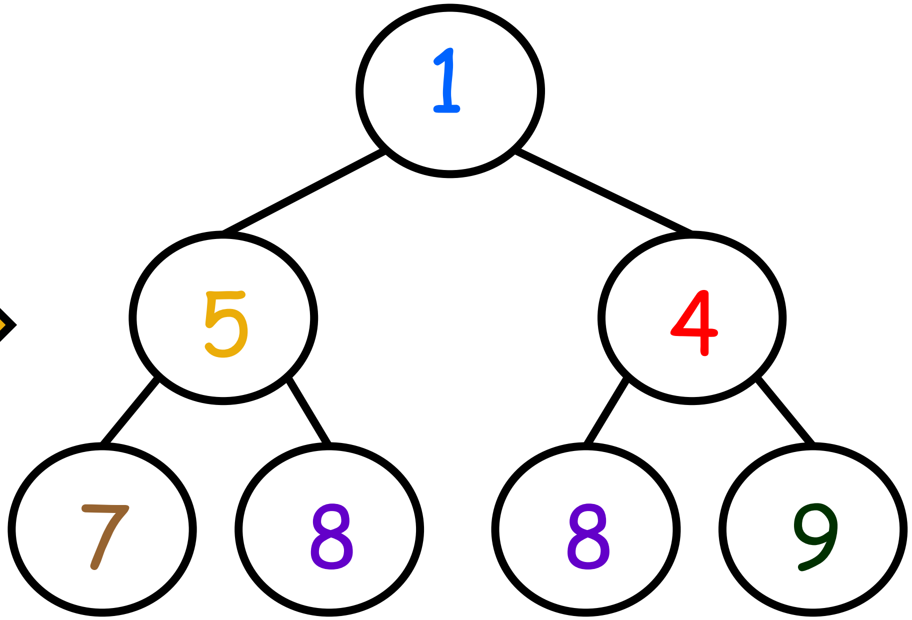
(c) Show how to do Extract-Min on a Young tableau in $O(m + n)$ time
Hint: Extract-Min in a heap
Show that your algorithm is correct.

(d) Show how to use an $n \times n$ Young tableau to sort n^2 numbers in $O(n^3)$ time

Question4

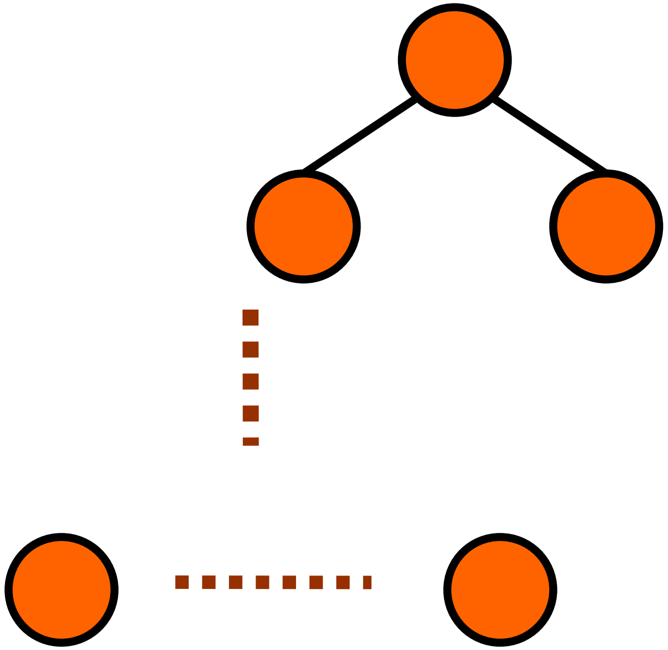
Young Tableau is similar to a HEAP

1	4	9
5	8	11
7	12	

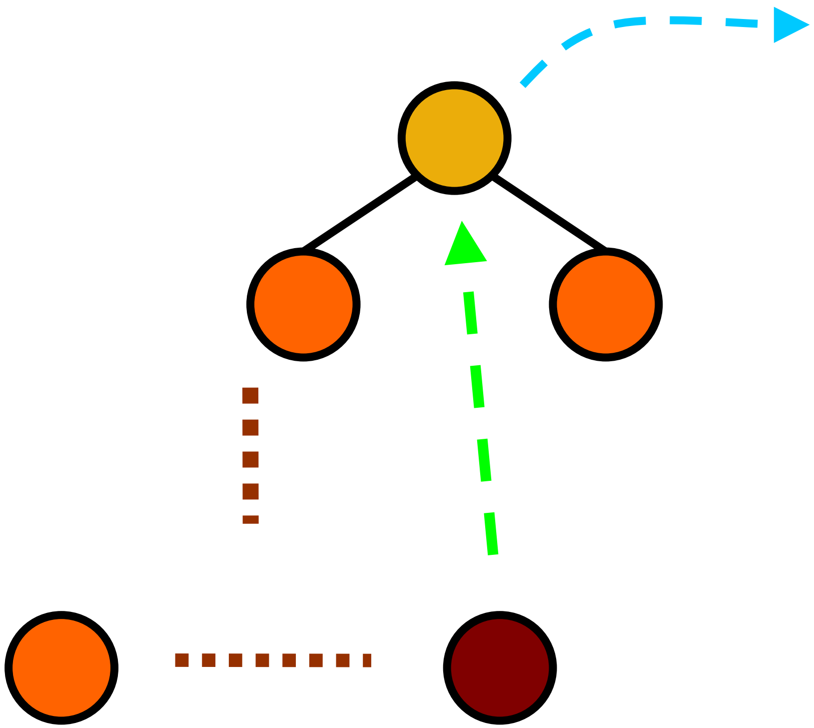


Question4

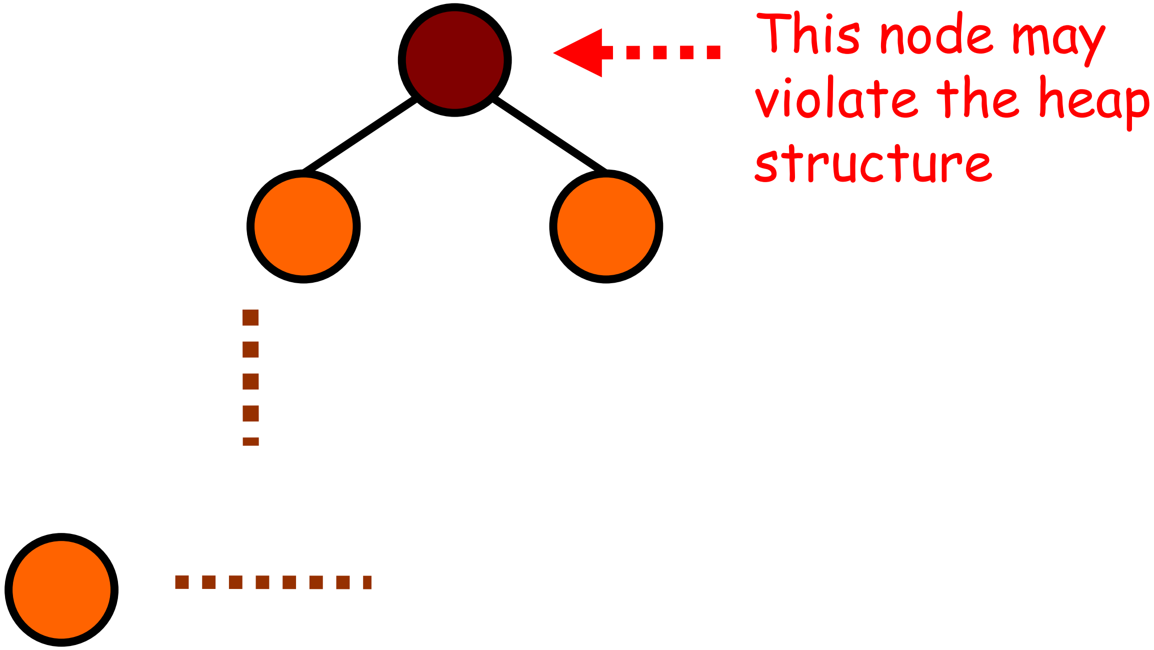
Recalling how we do Extract-Min in Heap



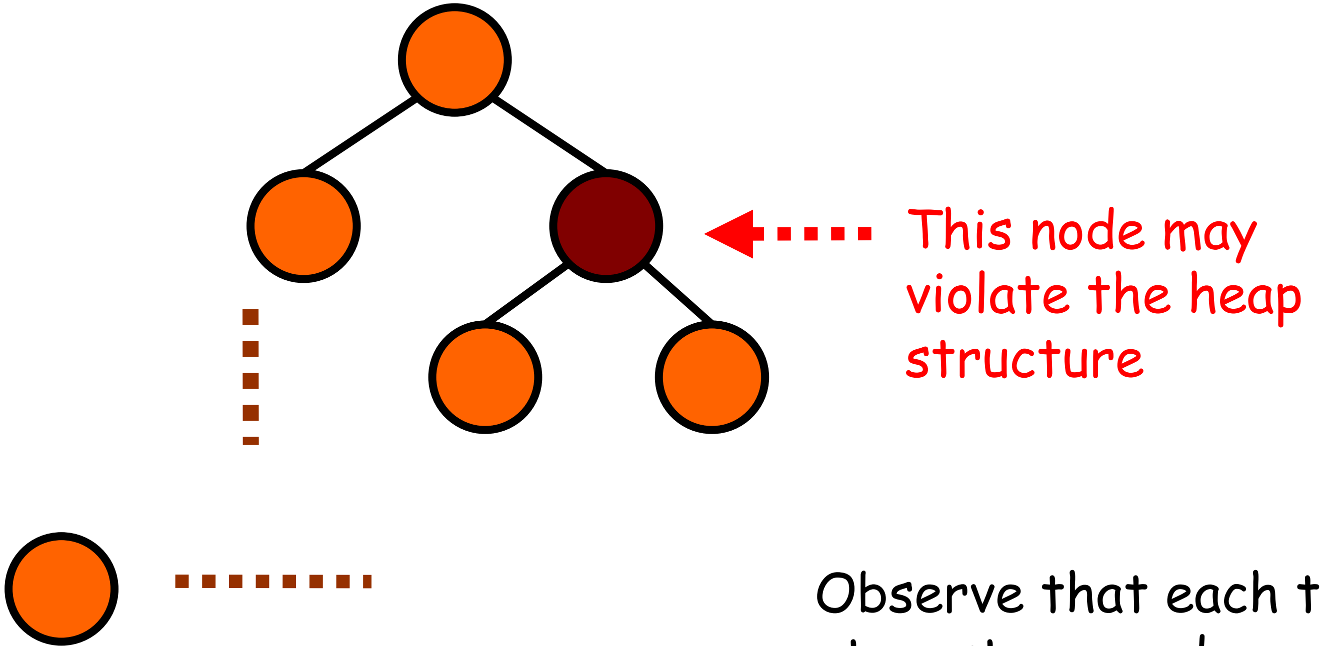
Question4



Question4

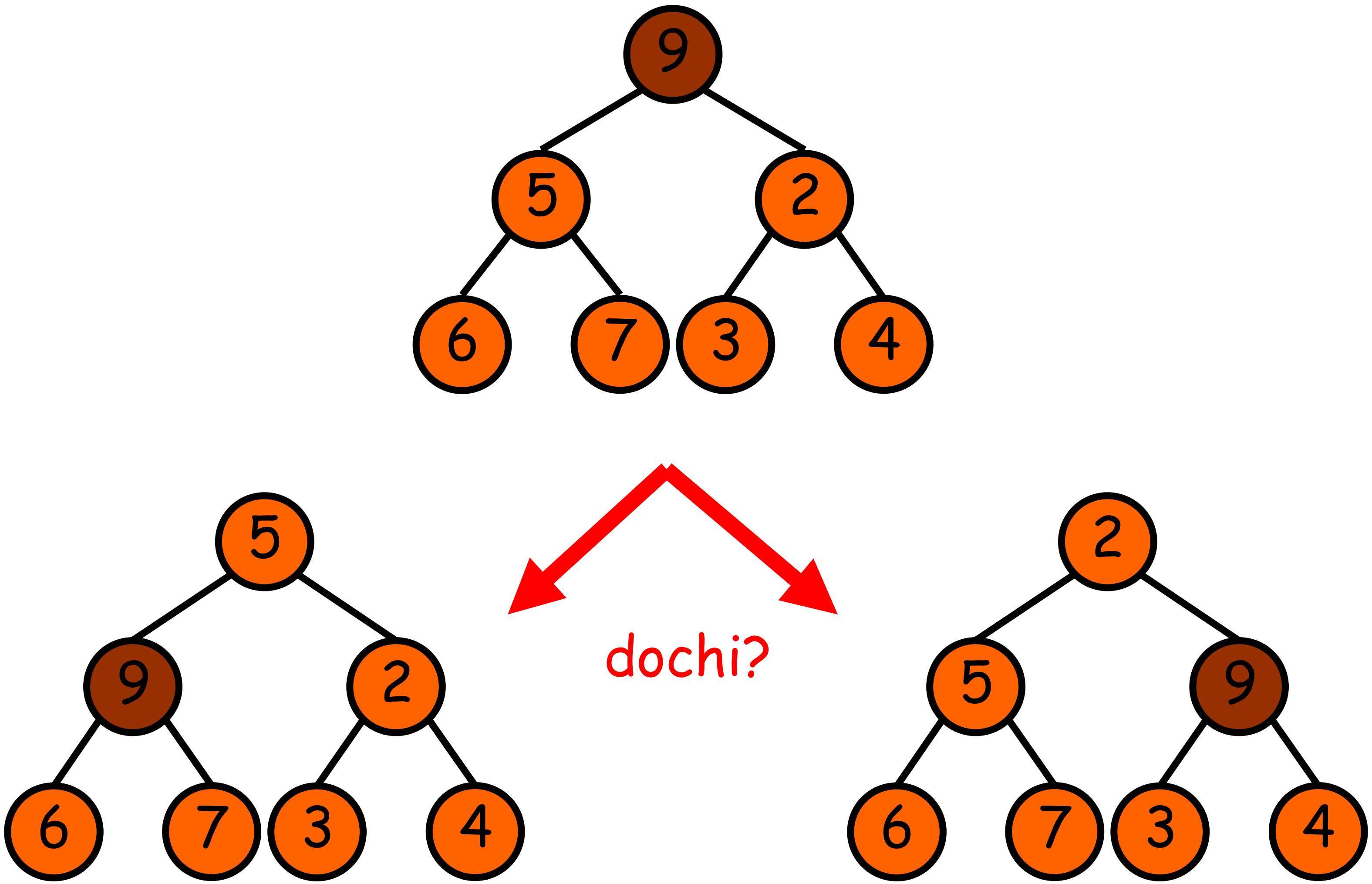


Question4

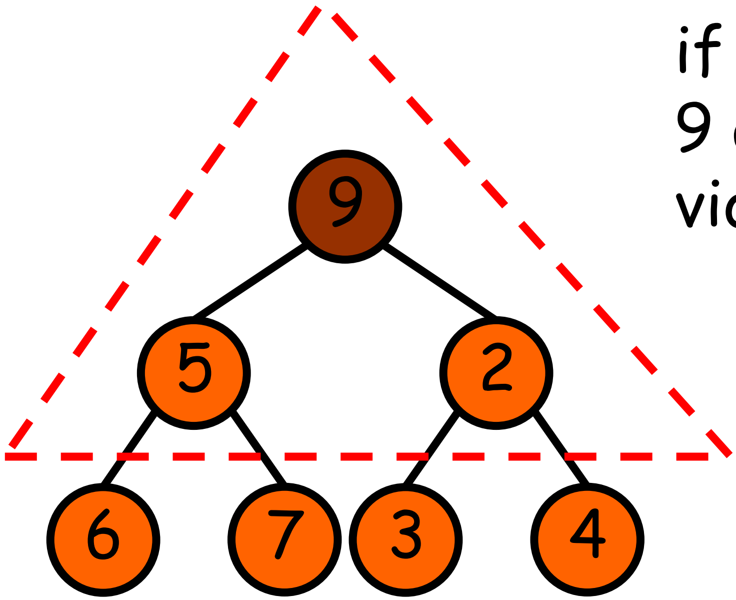


Observe that each time at most one node may violate the structure

Question4

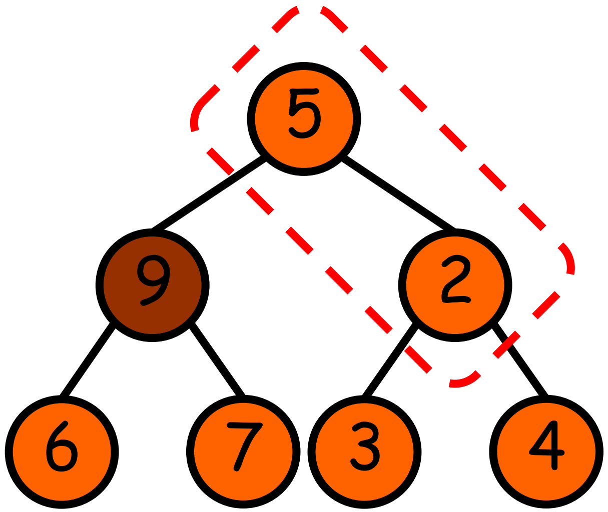


Question4



if we choose to swap 9 and 5, another violation occurs.

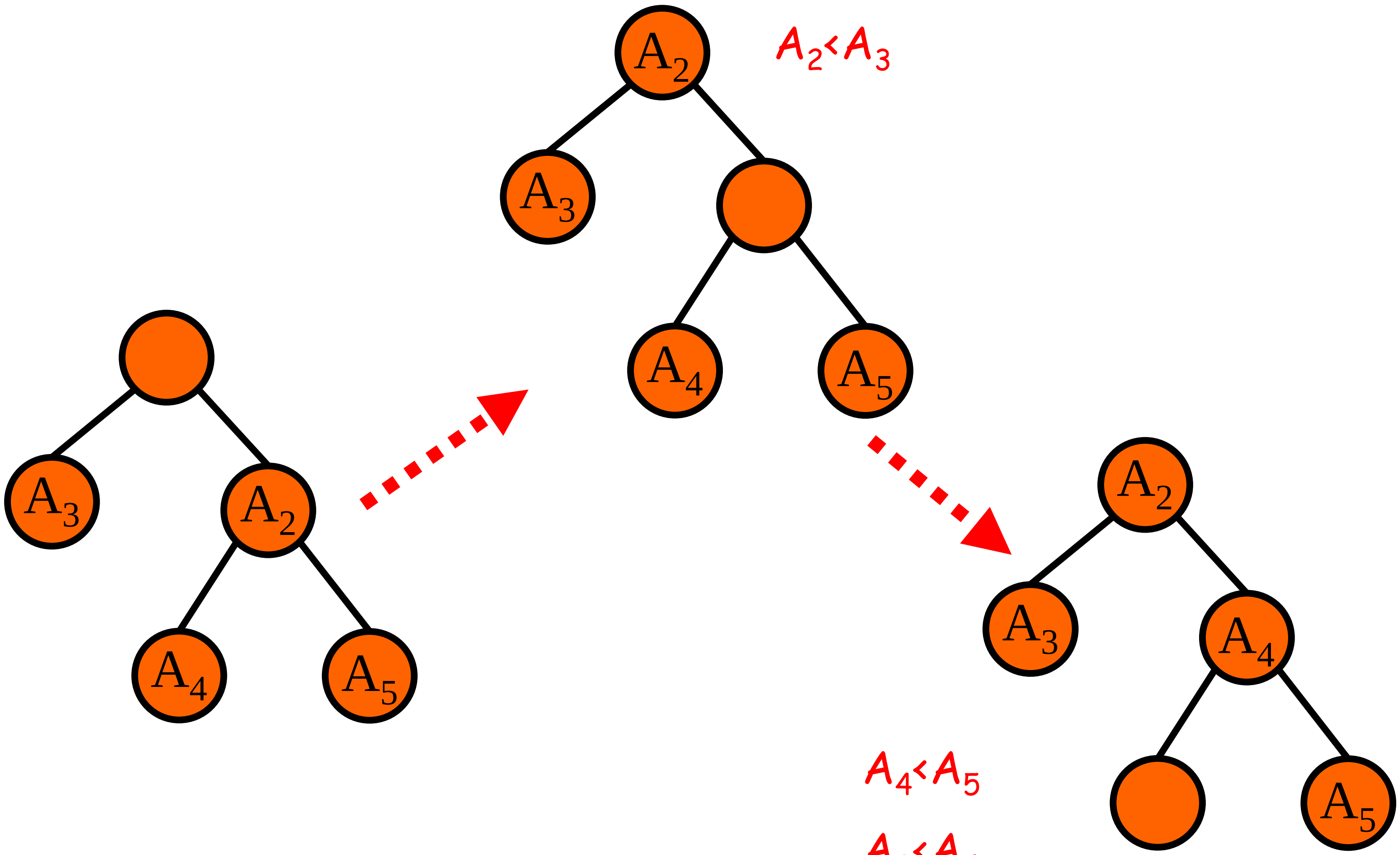
Question4



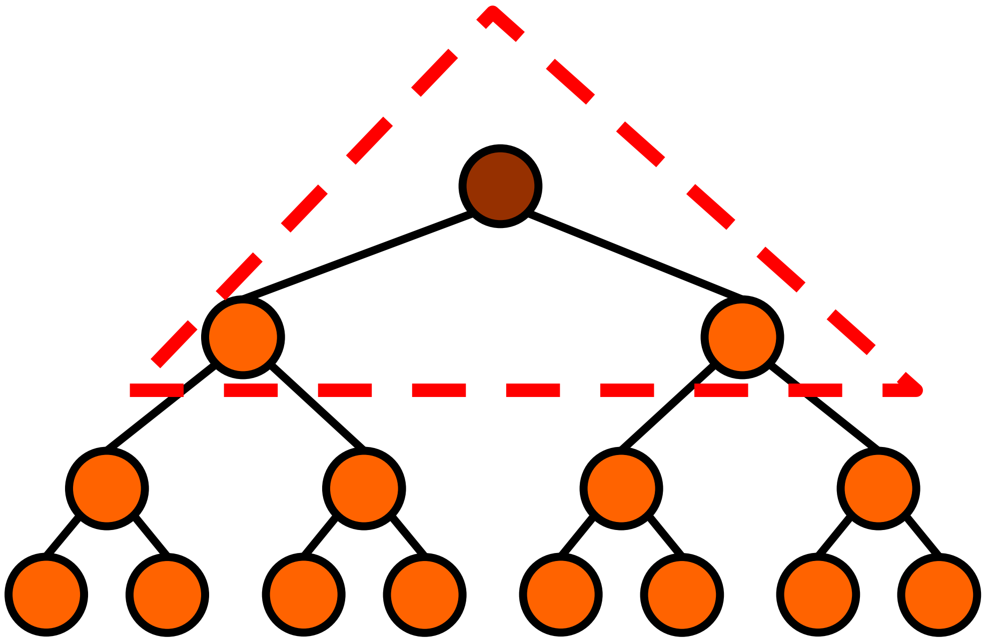
New violation !

So we choose to swap the minimum among nearby nodes

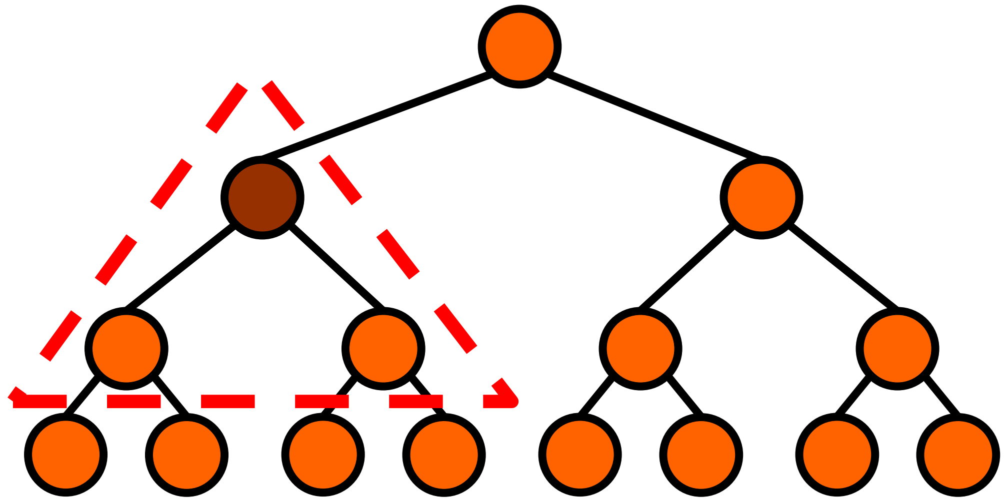
Question4



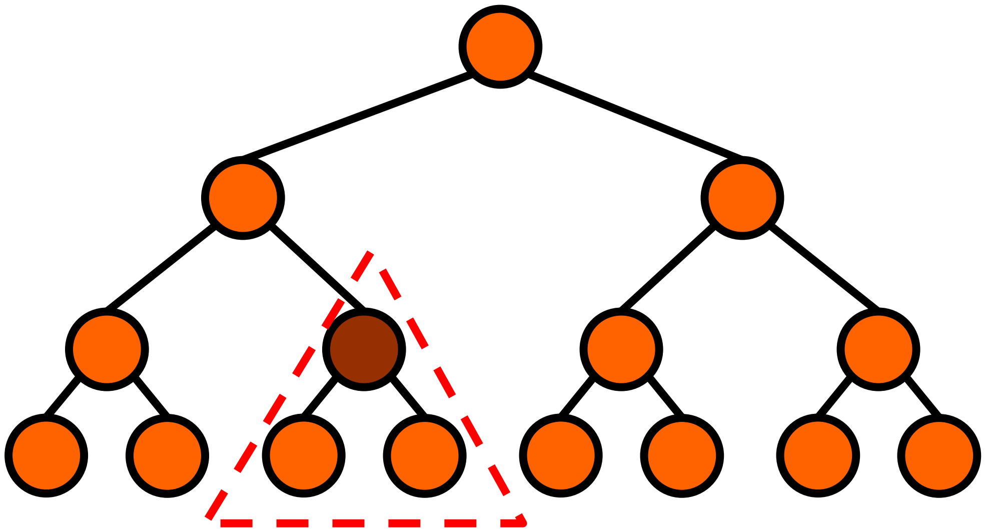
Question4



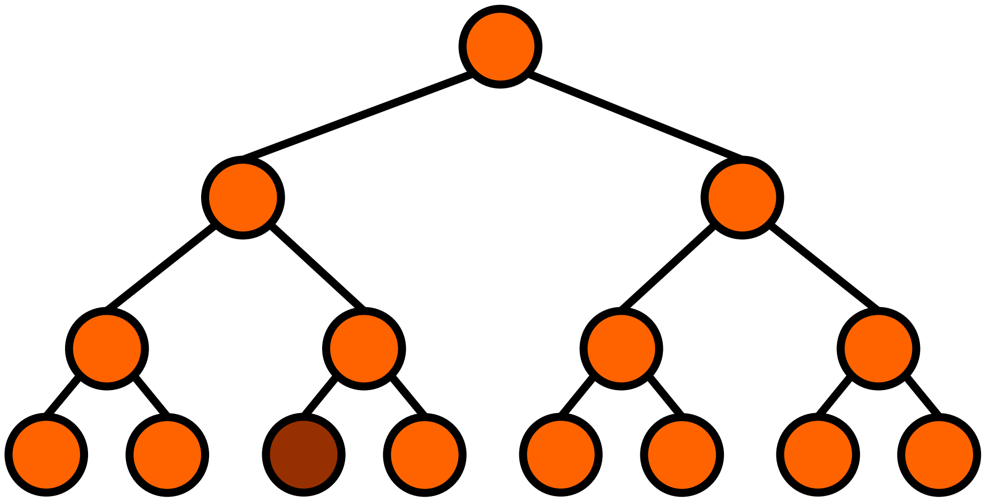
Question4



Question4



Question4



Question4

(d) You need to clearly show why your algorithm's time complexity is $O(n^3)$

Question5 (Bonus)

Finding a number in $O(m+n)$ time inside a Young tableau

Question5 (Bonus)

HINT 1:

1	4	5	9
3	5	6	13
4	6	12	14
7	9	10	16

This is a boundary that the ↖ direction contains all elements that is less than 8

Question5 (Bonus)

HINT 2:

1	4	5	9
3	5	6	13
4	6	12	14
7	9	10	16

5 is the greatest element in the red square, 12 is the greatest element in the pink square