Data Structures and Algorithms

(ESO207)

Lecture 31

- Magical applications of Binary trees -II

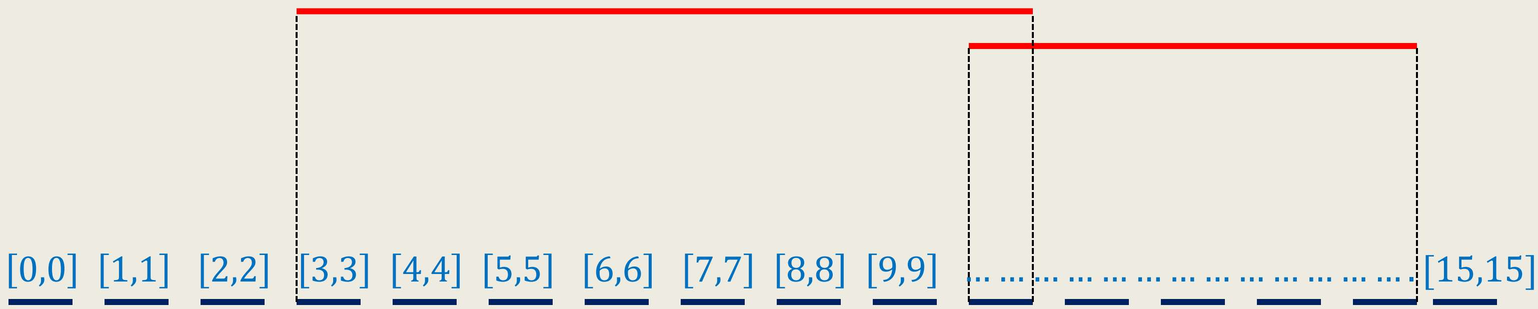
RECAP OF LAST LECTURE

Intervals

S = {[$\boldsymbol{i}$, $\boldsymbol{j}$], $0 \le i \le j < n$}

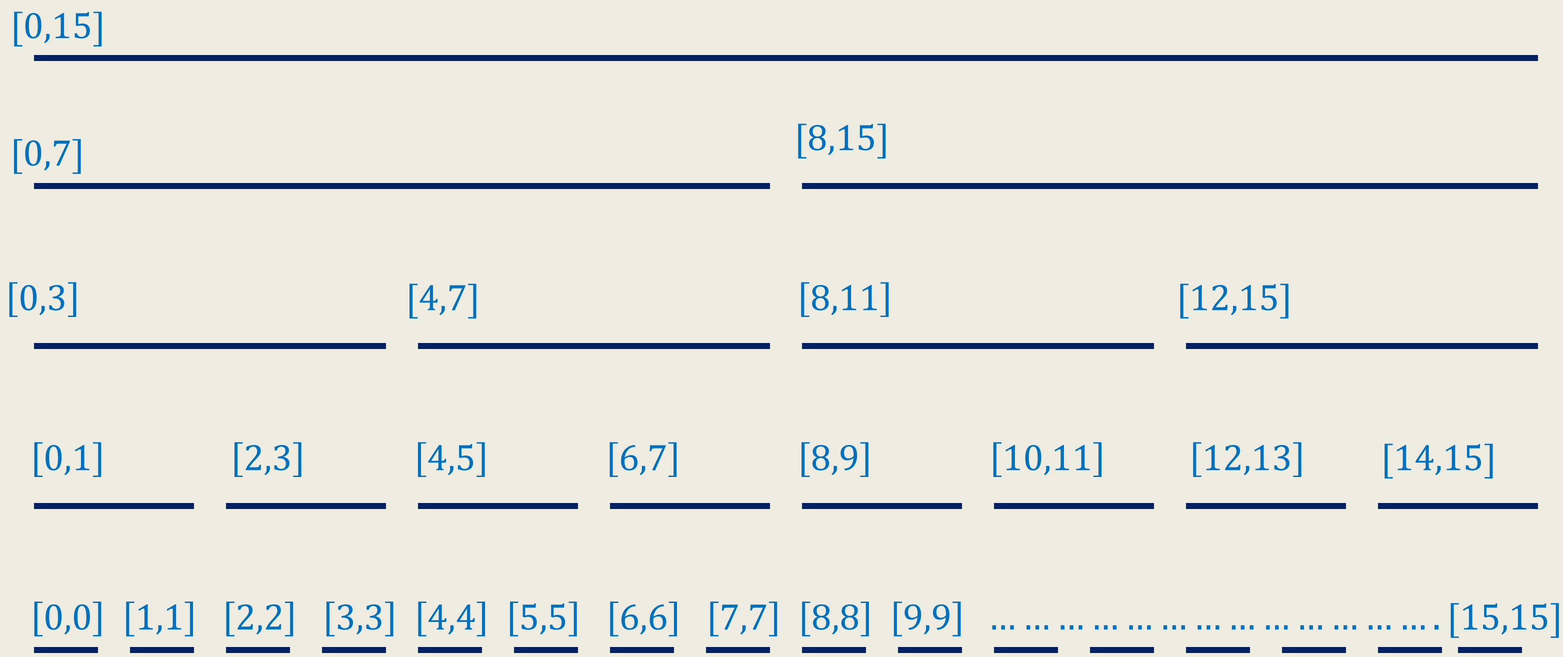
Question: Can we have a small set **X**⊂**S** of **intervals** s.t.

every interval in **S** can be expressed as a union of a few **intervals** from **X** ?

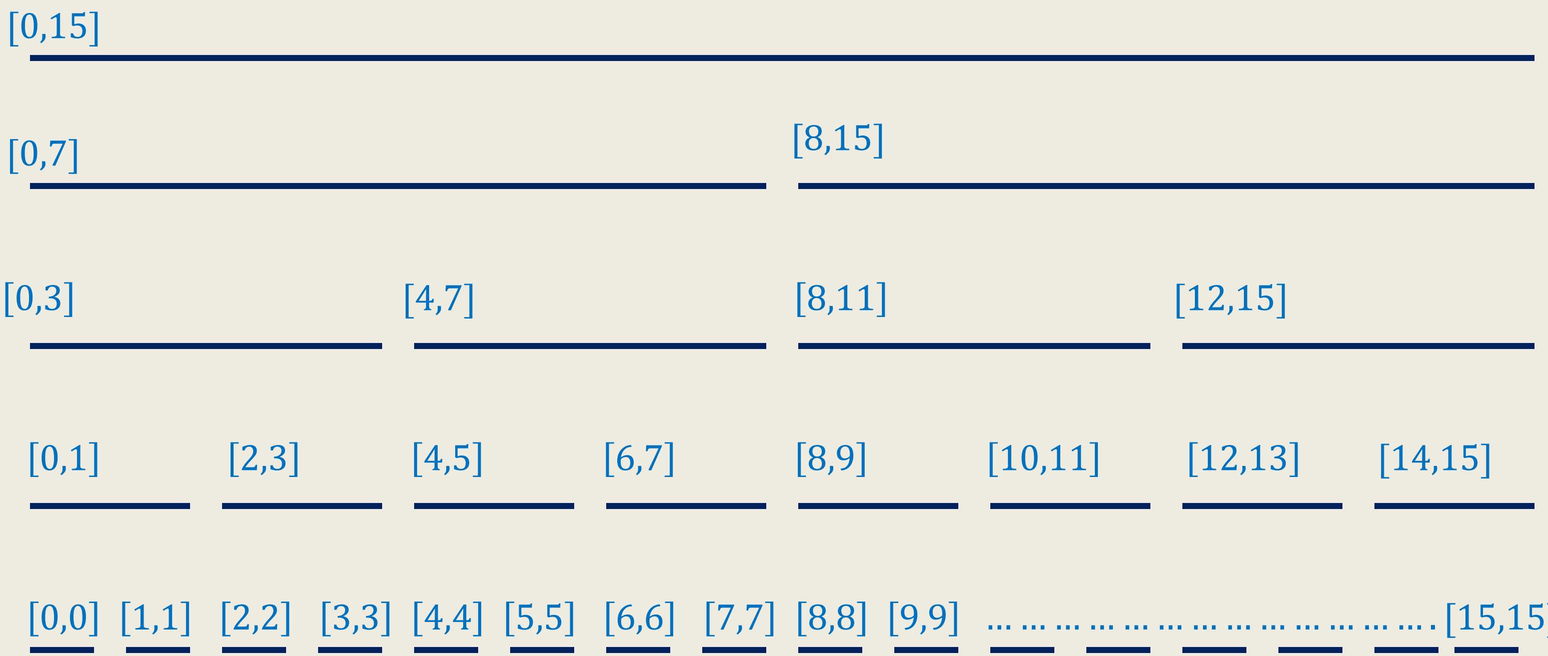


Answer: yes☺

Hierarchy of intervals

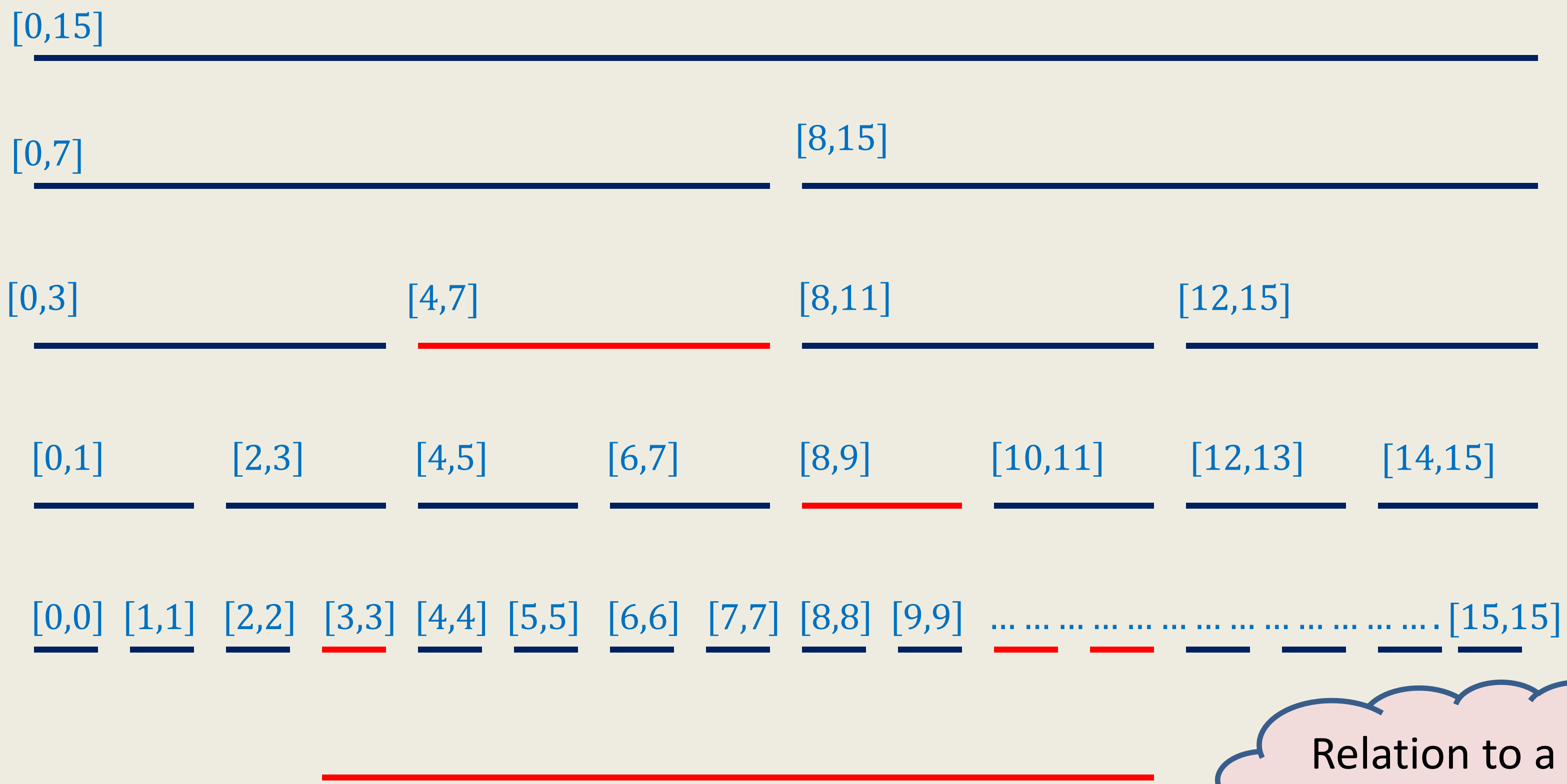


Hierarchy of intervals



Observation: There are $2n$ intervals such that any interval $[i, j]$ can be expressed as **union** of O(**log** n) basic intervals ☺

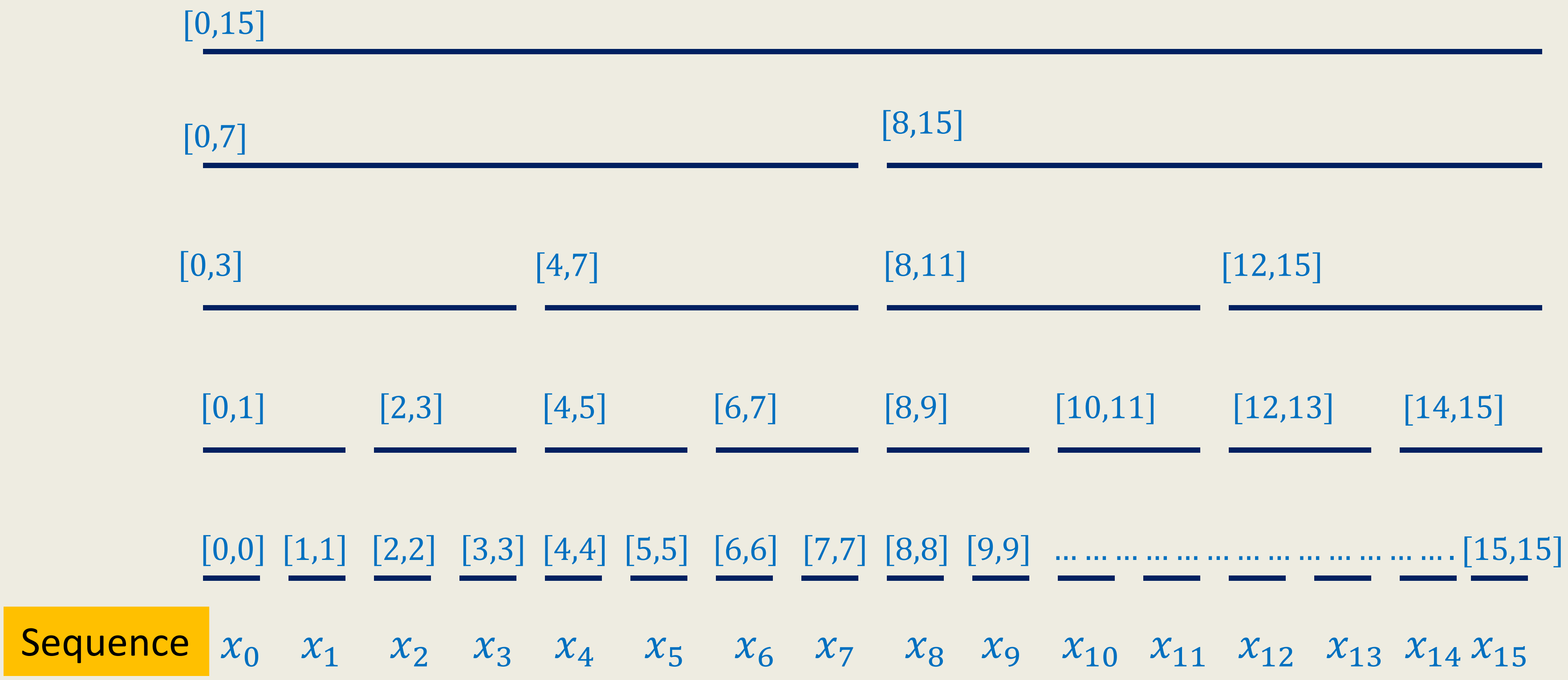
Hierarchy of intervals



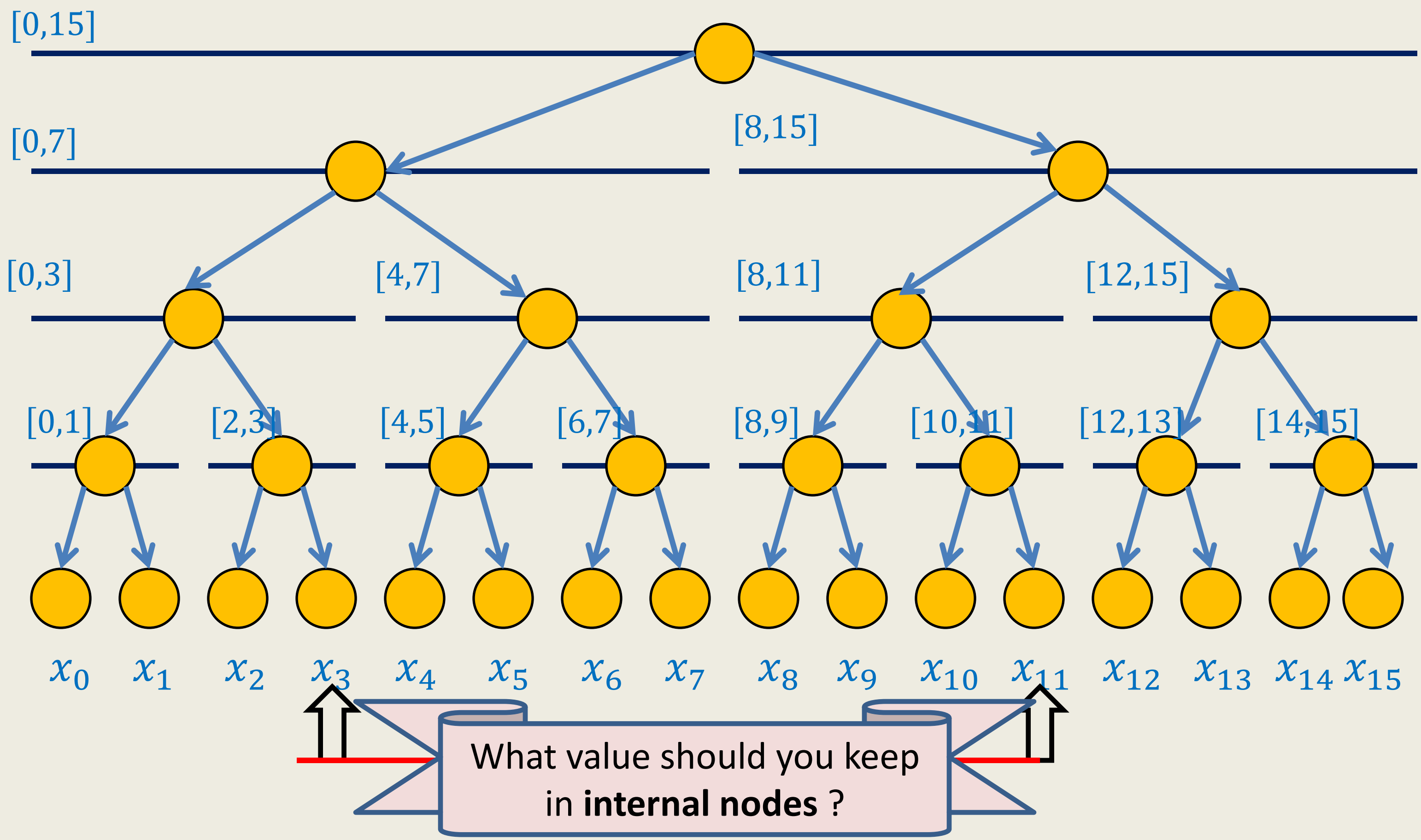
Observation: There are $2n$ intervals such that
any interval $[i, j]$ can be expressed as **union** of **O(log** n) basic intervals ☺

Relation to a **sequence** ?

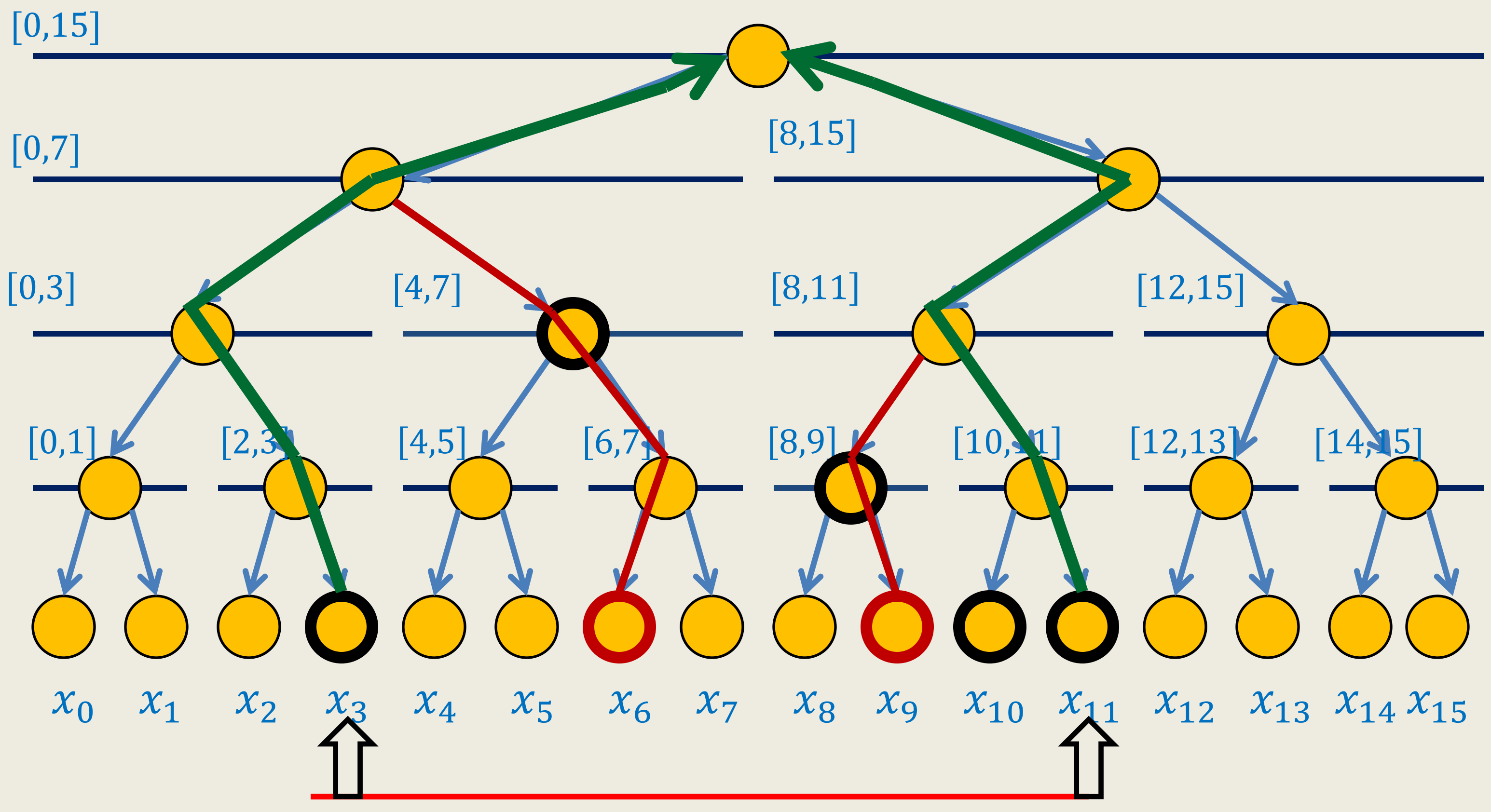
Which **data structure** emerges ?



A Binary tree



How to perform Operation on an interval ?



How to perform Operation on an interval ?

Problem 2

Dynamic Range-minima

Dynamic Range Minima Problem

Given an initial sequence **S** = $\prec x_0, \ldots, x_{n-1} \succ$ of numbers, maintain a compact data structure to perform the following operations efficiently for any $0 \le \boldsymbol{i} < \boldsymbol{j} < \boldsymbol{n}$.

- **ReportMin**($\boldsymbol{i}$, $\boldsymbol{j}$):

  Report the minimum element from $\{x_k$ | for each $\boldsymbol{i} \le \boldsymbol{k} \le \boldsymbol{j}\}$

- **Update**($\boldsymbol{i}$, **a**):

  a becomes the new value of x_i.

Example:

Let the initial sequence be **S** = $\prec$ 14, 12, 3, 49, 4, 21, 322, -40 $\succ$

ReportMin(1, 5) returns **3**

ReportMin(0, 3) returns **3**

Update(2, **19**) update **S** to $\prec$ 14, 12, **19**, 49, 4, 21, 322, -40 $\succ$

ReportMin(1, 5) returns **4**

ReportMin(0, 3) returns **12**

Dynamic Range Minima **Problem**

Given an initial sequence **S** = $\prec x_0, \ldots, x_{n-1} \succ$ of numbers, maintain a compact data structure to perform the following operations efficiently for any $0 \leq \boldsymbol{i} < \boldsymbol{j} < \boldsymbol{n}$.

- **ReportMin**($\boldsymbol{i}$, $\boldsymbol{j}$):

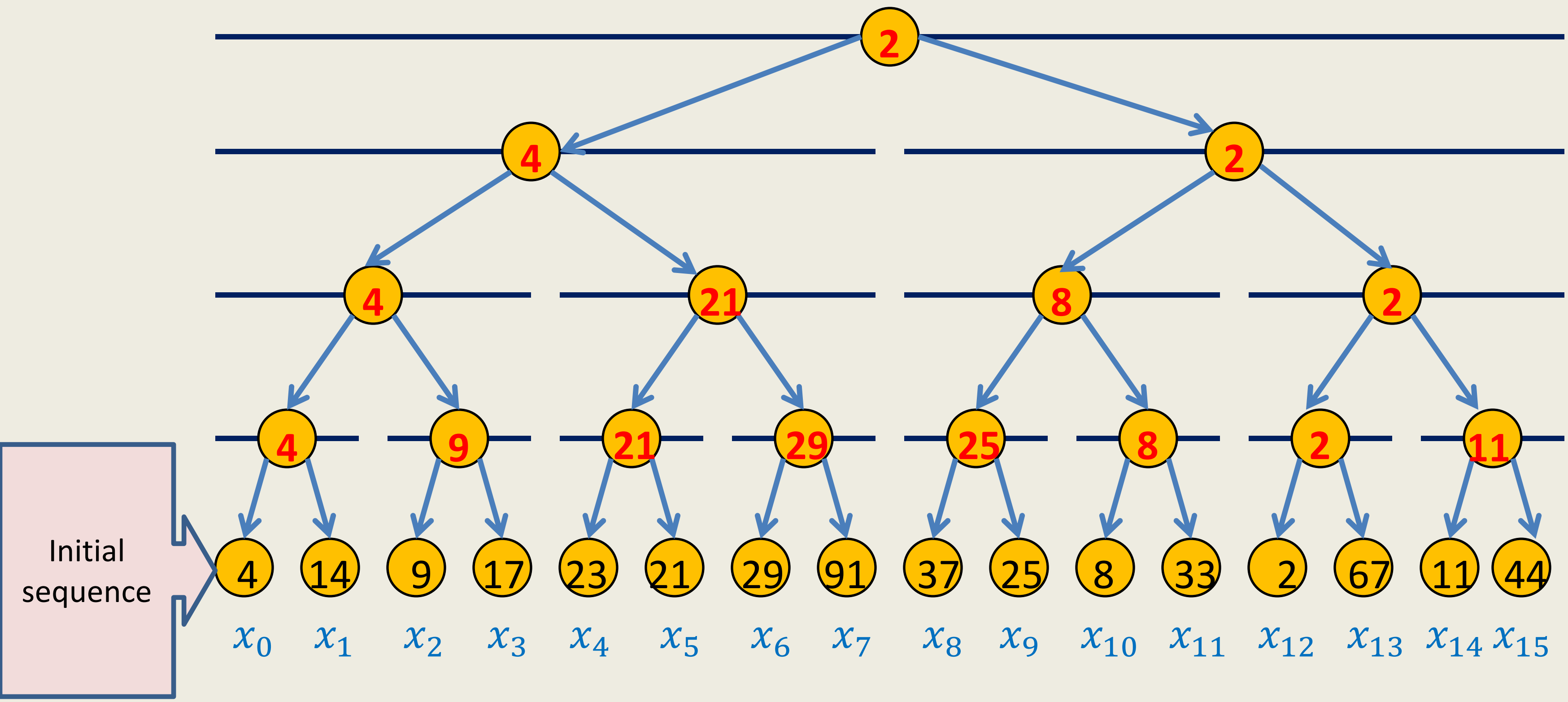
  Report the minimum element from $\{x_k$ | for each $\boldsymbol{i} \leq \boldsymbol{k} \leq \boldsymbol{j}\}$
- **Update**($\boldsymbol{i}$, **a**):

  a becomes the new value of x_i.

AIM:

- **O**($\boldsymbol{n}$) size data structure.
- **ReportMin**($\boldsymbol{i}$, $\boldsymbol{j}$) in **O**(log $\boldsymbol{n}$) time.
- **Update**($\boldsymbol{i}$, **a**) in **O**(log $\boldsymbol{n}$) time.

Data structure for dynamic range minima

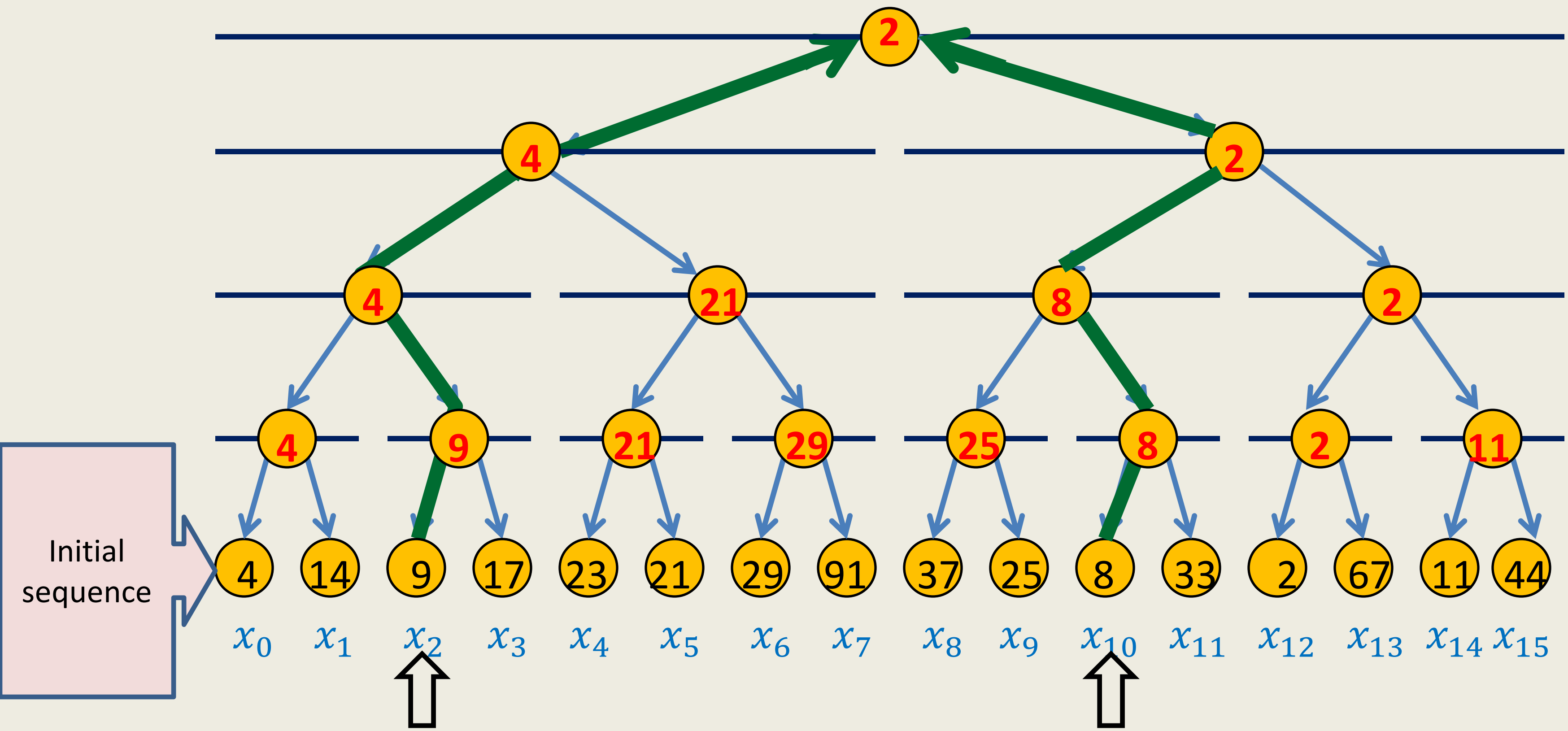


Question: What should be stored in an internal node **v** ?

Answer:

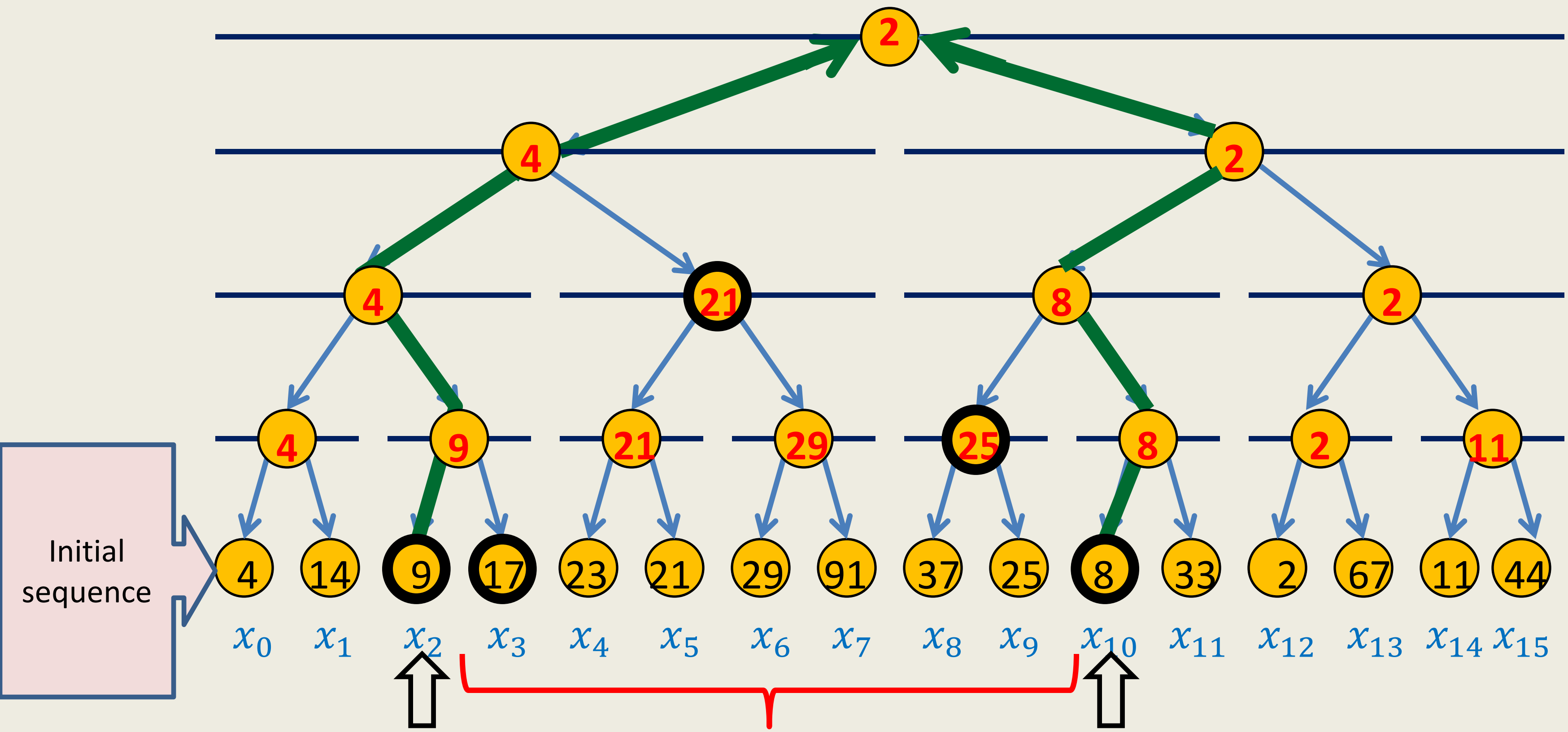
minimum value

Data structure for dynamic range minima



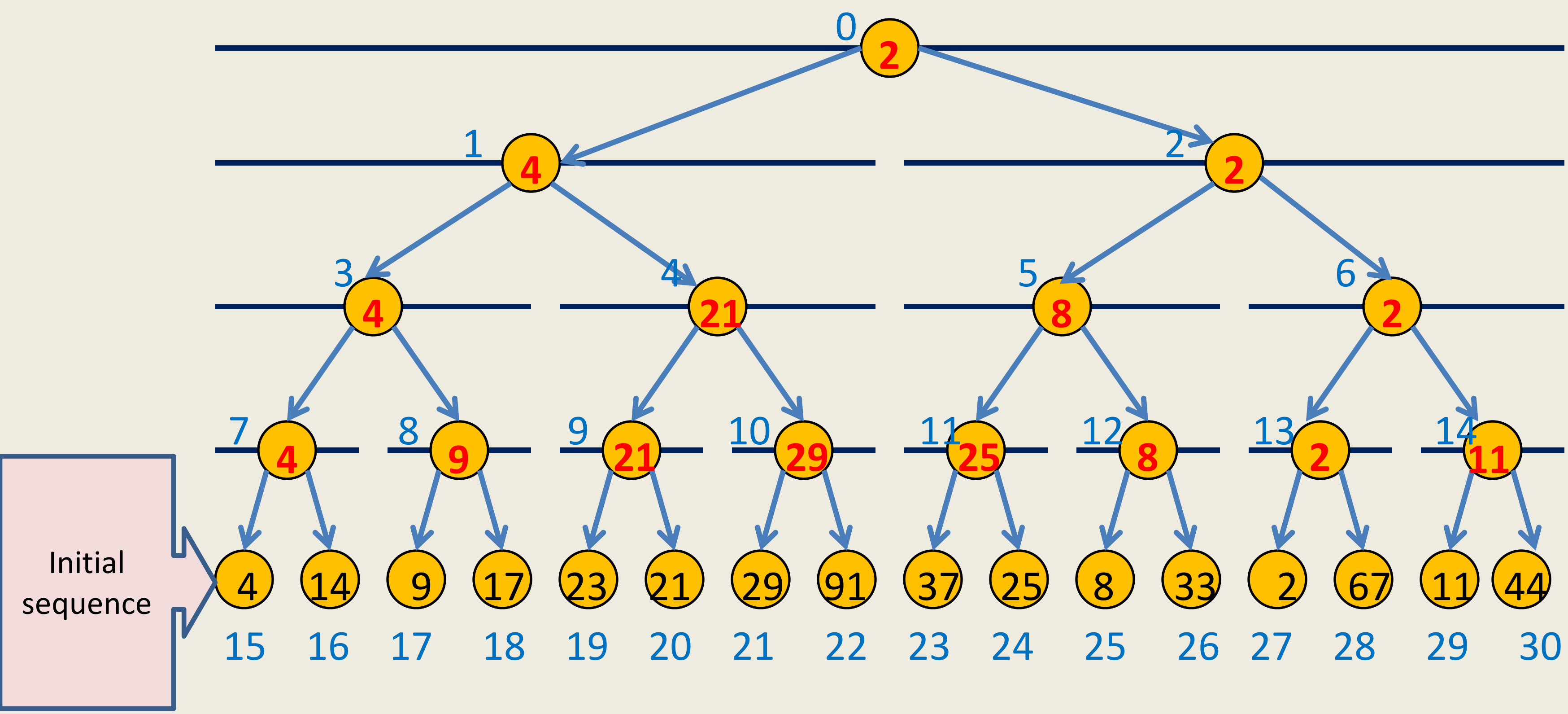
How to do Report-Min(2,10) ?

Data structure for dynamic range minima



How to do Report-Min(2,10) ?

Data structure for dynamic range minima



Data structure: An array **A** of size 2**n**-1.

Copy the sequence **S** = $\prec x_0, \ldots, x_{n-1} \succ$ into **A**$[n-1]$…**A**$[2n-2]$

Leaf node corresponding to x_i = **A**$[(n-1)+i]$

How to check if a node is left child or right child of its parent ?

(if index of the node is odd, then the node is left child, else the node is right child)

Update(i, a)

Update(i, a)

$i \leftarrow (n-1) + i$;

A[i**]**$\leftarrow a$;

$i \leftarrow \lfloor (i-1)/2 \rfloor$;

While(?? **)**

{

}

Homework

At the end of the lecture slides, an **incorrect** pseudocode is given for **Update(i, a)**.

It is followed by a **correct** one.

Ponder over it ☺

Report-Min(i,j)

```
Report-Min(i,j)
    i ← (n − 1) + i ;
    j ← (n − 1) + j ;
    min ← A(i);
    If (j > i)
    {       If (A(j)< min)   min ← A(j);
            While( ⌊(i − 1)/2⌋ <> ⌊(j − 1)/2⌋ )
            {
                If(    i%2=1 and A(i + 1)< min     )   min ←             ;
                If(    j%2=0 and A(j − 1)< min     )   min ←             ;
                i ←                  ;
                j ←                  ;
            }
    }
    return min;
```

Another interesting problem on sequences

Practice Problem

Given an initial sequence **S** = $\prec x_0, \ldots, x_{n-1} \succ$ of n numbers,
maintain a compact data structure to perform the following operations efficiently :

- **Report_min**($\boldsymbol{i}$, $\boldsymbol{j}$):

  Report the minimum element from $\{x_i, \ldots x\_j\}$.

- **Multi-Increment**($\boldsymbol{i}$, $\boldsymbol{j}$, **Δ**):

  Add **Δ** to each x_k for each $\boldsymbol{i} \leq \boldsymbol{k} \leq \boldsymbol{j}$

Example:

Let the initial sequence be **S** = $\prec$ 14, 12, 3, 12, 111, 51, 321, -40 $\succ$

Report_min(**1** , **4**):

returns 3

Multi-Increment(2,6,10):

S becomes $\prec$ 14, 12, **13**, **22**, **121**, **61**, **331**, -40 $\succ$

Report_min(**1** , **4**):

returns 12

An challenging problem on sequences

For summer vacation

(not for the exam)

$*$ Problem

Given an initial sequence **S** = $\prec x_0, \ldots, x_{n-1} \succ$ of n numbers,
maintain a compact data structure to perform the following operations efficiently :

- **Report_min**($\boldsymbol{i}$, $\boldsymbol{j}$):

  Report the minimum element from $\{x_i, \ldots x\_j\}$.

- **Multi-Increment**($\boldsymbol{i}$, $\boldsymbol{j}$, **Δ**):

  Add **Δ** to each x_k for each $\boldsymbol{i} \le \boldsymbol{k} \le \boldsymbol{j}$

- **Rotate**($\boldsymbol{i}$, $\boldsymbol{j}$):

  $x_i \leftrightarrow x_j$, $x_{i+1} \leftrightarrow x_{j-1}$, ….

Example:

Let the initial sequence be **S** = $\prec$ 14, 12, 23, 19, 111, 51, 321, -40 $\succ$

After **Rotate**(1,6), **S** becomes

$\prec$ 14, **321**, **51**, **111**, **19**, **23**, **12**, -40 $\succ$

The solution of this problem will require additional tools. But all these tools have already been discussed in the course.

Problem 4

A data structure for sets

Sets under operations

Given: a collection of n singleton sets $\{0\}$, $\{1\}$, $\{2\}$, … $\{n-1\}$

Aim: a compact data structure to perform

- **Union**(i, j):

  Unite the two sets containing i and j.

- **Same_sets**(i, j):

  Determine if i and j belong to the same set.

Trivial Solution 1

Keep an array **Label[]** such that

Label[i]=**Label**[j] if and only if i and j belong to the same set.

➔ **Same_sets**(i, j):

check if **Label**[i]=**Label**[j] ?

O(1) time

➔ **Union**(i, j):

O(n) time

For each $0 \le k < n$

if (**Label**[k]= **Label**[i]) **Label**[k] ← **Label**[j])

Sets under operations

Given: a collection of n singleton sets $\{0\}$, $\{1\}$, $\{2\}$, … $\{n-1\}$

Aim: a compact data structure to perform

- **Union**(i, j):

  Unite the two sets containing i and j.

- **Same_sets**(i, j):

  Determine if i and j belong to the same set.

Trivial Solution 2

Treat the problem as a graph problem: ??

Connected component

- **V** = $\{0, \ldots, n-1\}$, **E** = empty set initially.
- A set ⇔
- Keep array **Label[]** such that **Label**[i]=**Label**[j] iff i and j belong to the same component.

➔

Union(i, j) :

O(n) time

add an edge (i, j) and

recompute connected components using **BFS/DFS**.

Sets under operations

Given: a collection of n singleton sets $\{0\}$, $\{1\}$, $\{2\}$, … $\{n-1\}$

Aim: a compact data structure to perform

- **Union**(i, j):

  Unite the two sets containing i and j.

- **Same_sets**(i, j):

  Determine if i and j belong to the same set.

Efficient solution:

- A data structure which supports each operation in **O(log n)** time.

- **An additional heuristic**

  ➔ time complexity of an operation :

We shall discuss it in the next lecture.

Homework

For **Dynamic Range-minima** problem

Update(i, a)

Update(i, a)

$i \leftarrow (n-1) + i$;

A[i]← a ;

$i \leftarrow \lfloor (i-1)/2 \rfloor$;

While($i \geq 0$ **)**

{

If(a< **A[**i**]**) **A[**i**]←** a**;**

$i \leftarrow \lfloor (i-1)/2 \rfloor$;

}

There is an **error** in the above pseudocode. Try spotting it.

Update(i, a)

Update(i, a)

$i \leftarrow (n-1) + i$;

A[i]$\leftarrow$ a ;

$i \leftarrow \lfloor (i-1)/2 \rfloor$;

While($i \geq 0$)

{

If(A[$2i+1$]< A[$2i+2$])

A[i]$\leftarrow$ A[$2i+1$]

else

A[i]$\leftarrow$ A[$2i+2$] ;

$i \leftarrow \lfloor (i-1)/2 \rfloor$;

}

Here is the correct pseudocode.