Data Structures and Algorithms

(**CS210A**)

Semester I – **2014-15**

Lecture 6:

- A compact and fast data structure for **Range-minima problem**
- Proof of correctness of algorithm: **Examples**

The data structures

Purpose:

To organize a data in the memory so that any query can be answered efficiently.

Example:

Data: A set **S** of n numbers

Query: "Is a number x present in **S** ?"

A trivial solution: **sequential search**

$O(n)$ time per query

A Data structure solution:

- Sort **S**
- Use **binary search** for answering query

$O(\log n)$ time per query

The data structures

Purpose:

To organize a data in the memory so that any query can be answered efficiently.

Important assumption: Many queries will have to be answered.

Parameters :

- Query time
- Space
- Preprocessing time

RANGE-MINIMA Problem

An interesting example to realize the importance of data structures

Range-Minima Problem

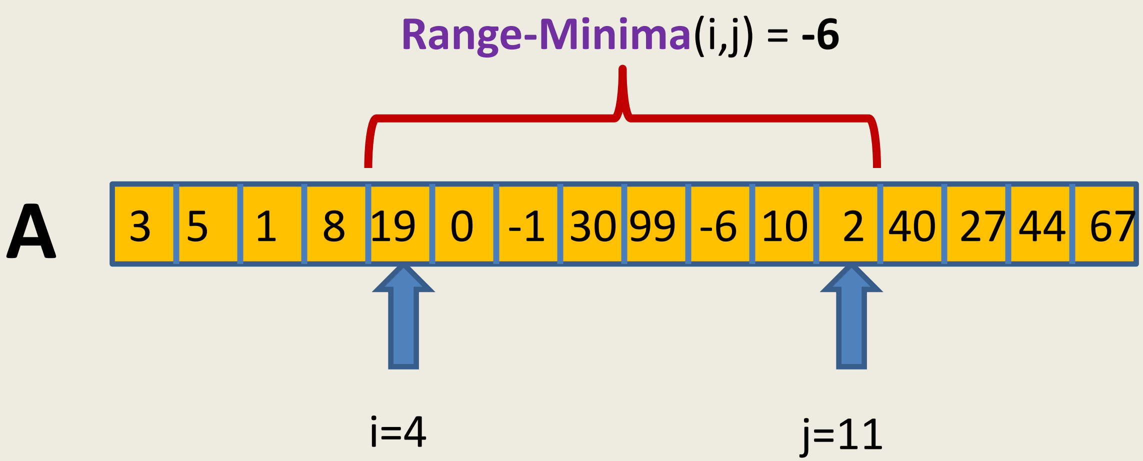
Given: an array **A** storing n numbers,

Aim: a data structure to answer a sequence of queries of the following type

Range-minima(i,j) : report the smallest element from **A**[i],…,**A**[j]

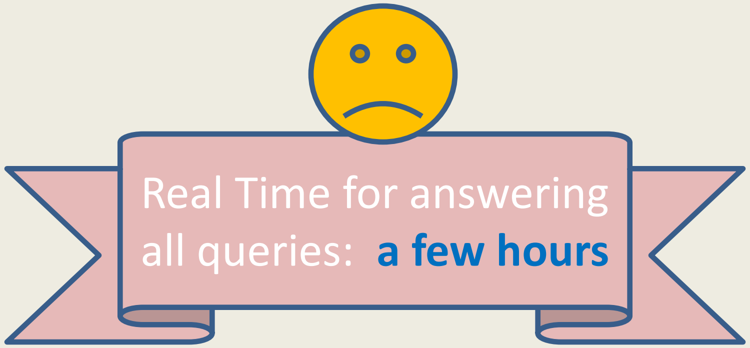
Let **A** store one **million** numbers

Let the number of queries be **10 millions**



Range-Minima Problem

Solution 1: Answer each query in a brute force manner using **A** itself.

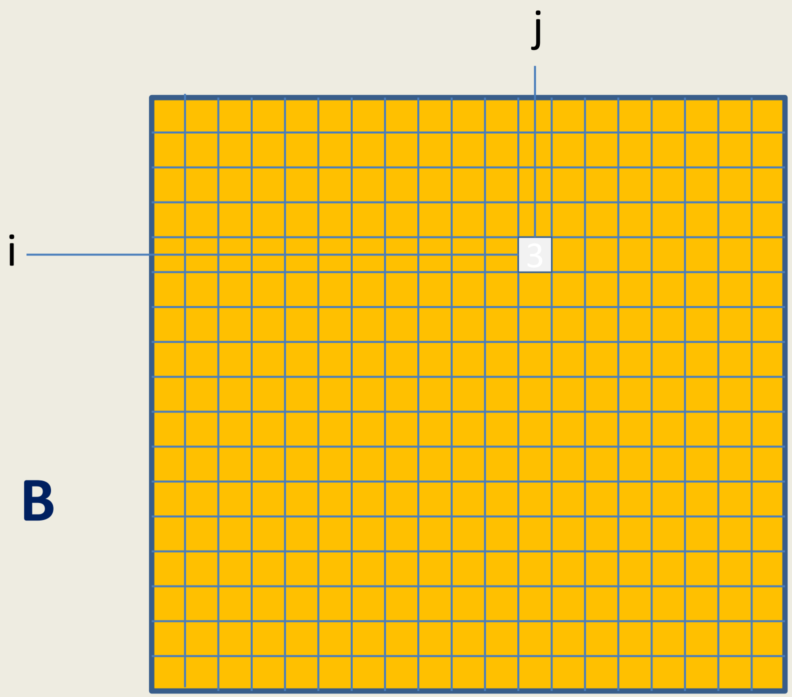


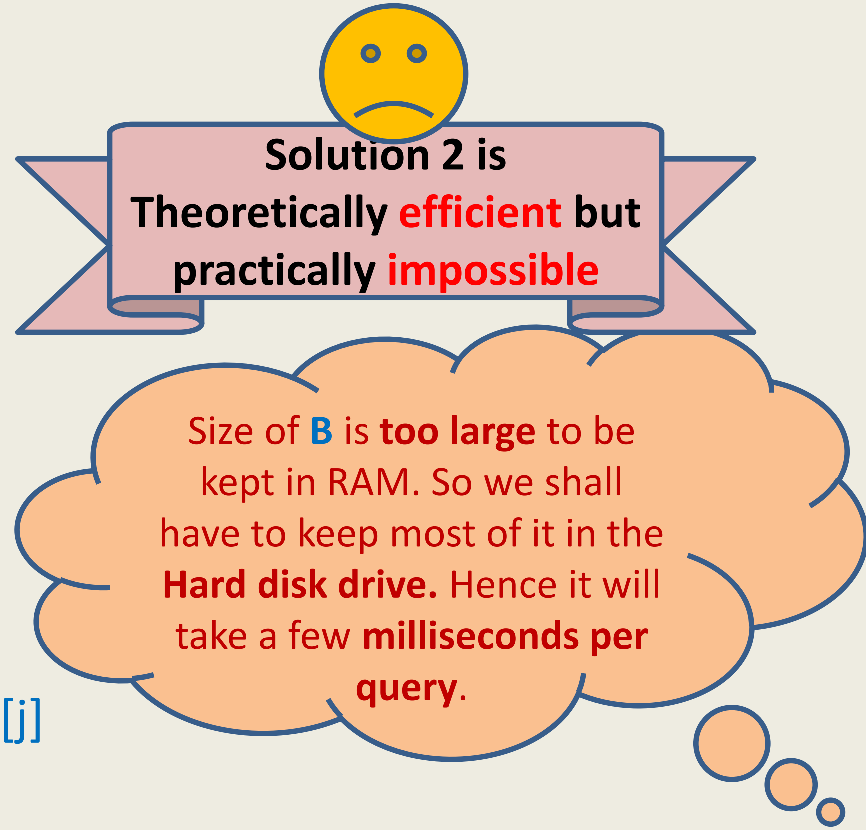

```
Range-minima-trivial(i,j)
{    temp ← i+1;
     min ← A[i];
     While(temp <= j)
     {    if (min > A[temp])
                min ← A[temp];
          temp← temp+1;
     }
     return min
}
```

Time complexity for answering a query: O(n) (equivalent to few **milliseconds)**

Range-Minima Problem

Solution 2: Compute and store answer for each possible query in a $n \times n$ matrix **B**.





B[i][j] stores the smallest element from **A**[i],…,**A**[j]

Space : O(n^2)

Range-minima Problem

Query:

Report_min(**A**,i,j) : report smallest element from {**A**[i],…,**A**[j]}

Aim :

To build a **compact** data structure which can answer Report_min(**A**,i,j) in **<u>O(1) time</u>** for any $1 \le i < j \le n$.

Why does ***O***$(\boldsymbol{n^2})$ bound on space appear **so hard** to break if we want ***O(1)*** query time?

… Because of **artificial hurdles**

Artificial hurdle

If we want to answer each query in O(1) time,

➔ we must store its answer explicitly.

➔ Since there are around $\boldsymbol{O}(n^2)$ queries, so $\boldsymbol{O}(n^2)$ space is needed.

Spend some time to find the origin of this hurdle….

Artificial hurdle

… If we fix the first parameter i for all queries, we need $O(n)$ space.

True Fact

because it assumes that data structure for an index i will work in isolation of others (**NO collaboration**)

So

for all i, we need $O(n^2)$ space.

A wrong inference

Collaboration (team effort) works in real life

Why not try collaboration for the given problem ?

Range-minima problem:
Breaking the $O(n^2)$ barrier using collaboration

An Overview:

- Keep n tiny data structures:

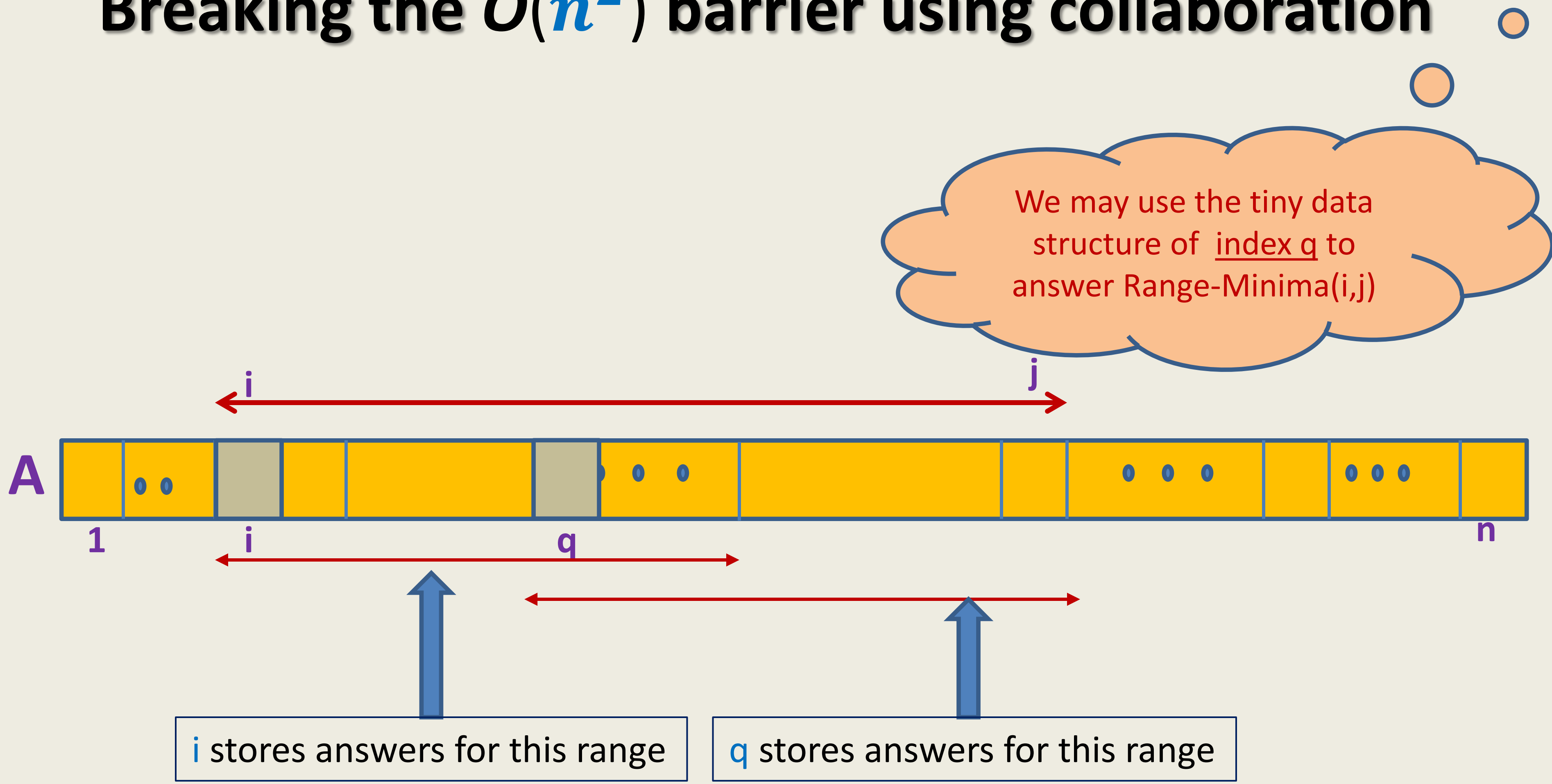
  Each index i stores minimum **only for a few** j>i.

- For a query **Range-minima**(i,j), if the answer is not stored in the tiny data structure of i,

  look up tiny data structure of some index q (chosen carefully).

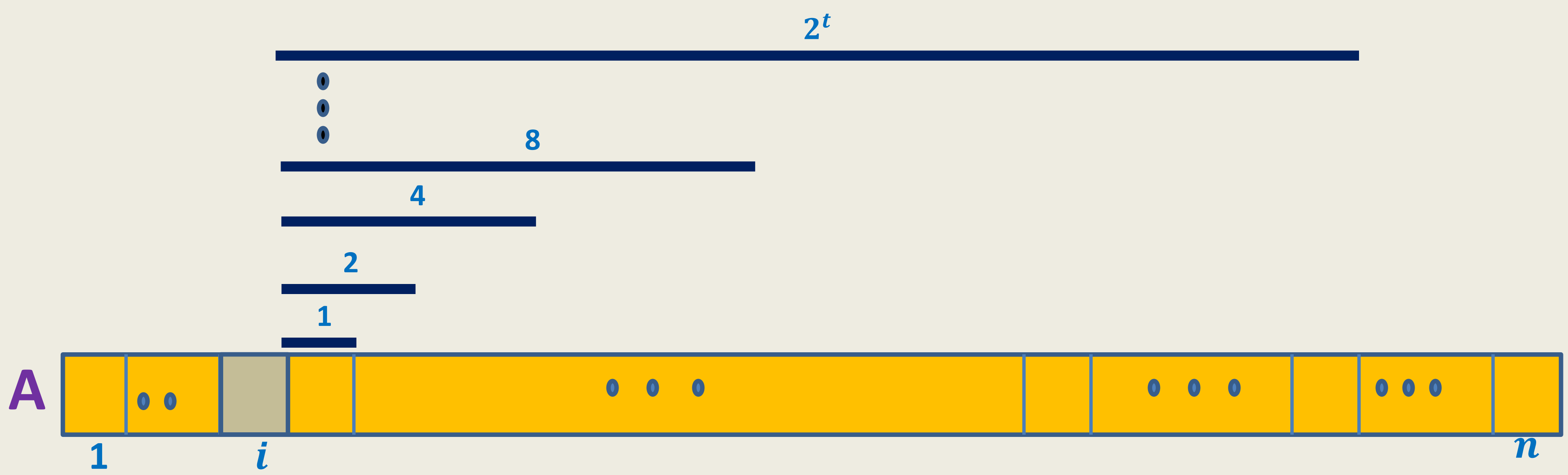
HOW DOES COLLABORATION WORK IN THIS PROBLEM ?

Range-minima problem:
Breaking the $O(n^2)$ barrier using collaboration



DETAILS OF TINY DATA STRUCTURES

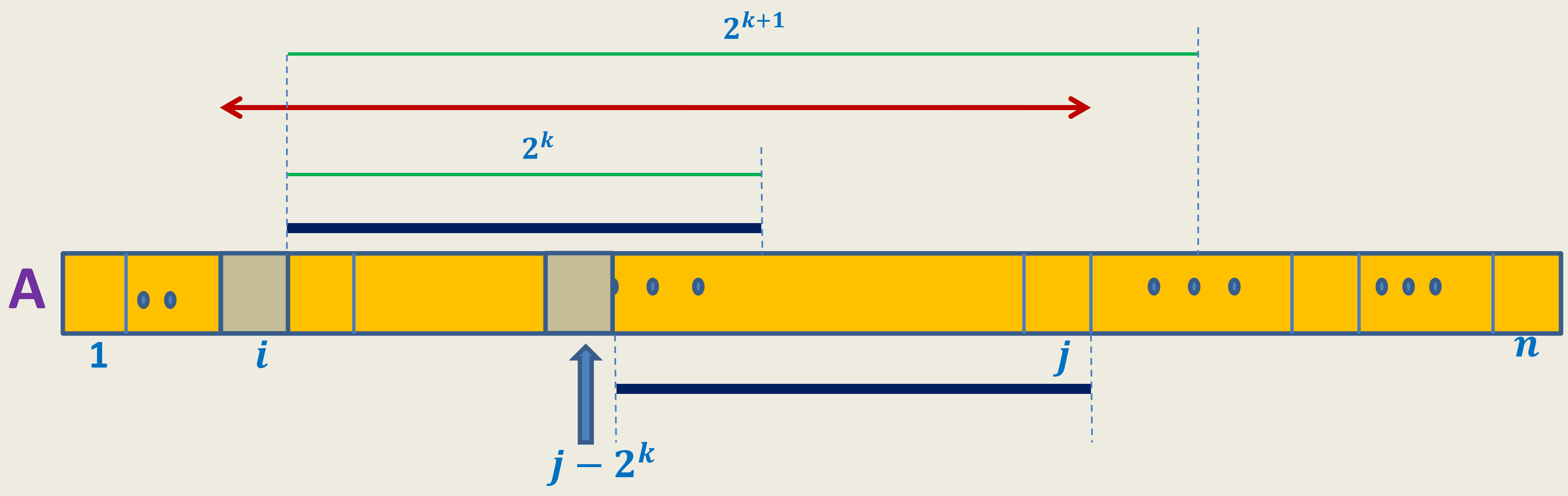
Range-minima problem :
Details of tiny data structure stored at each i



Tiny data structure of Index i stores minimum element for {**A**[i],…,**A**[i +2^k]} for each $k \le \log_2 n$

Answering Range-minima query for index i :

Collaboration works



We shall use two additional arrays

Definition :

Power-of-2[m] : the greatest number of the form 2^k such that $2^k \le m$.

Examples: **Power-of-2**[5] = 4,

Power-of-2[19**]=** 16**,**

Power-of-2[32**]=**32.

Definition :

Log[m] : the greatest integer k such that $2^k \le m$.

Examples: **Log**[5] = 2,

Log[19**]=** 4**,**

Log[32**]=**5.

Homework: Design **O**(n) time algorithm to compute arrays **Power-of-2**[] and **Log[]** of size n.

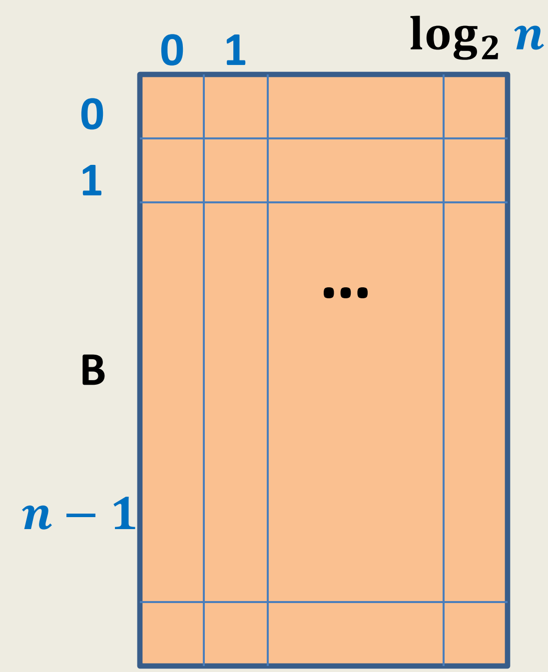
FINAL SOLUTION FOR
RANGE MINIMA PROBLEM

Range-Minima Problem:

Data structure with **O**(n **log** n) **space** and **O**(1) **query time**

Data Structure:

- n × **log** n matrix **B** where **B**[i][k] stores minimum of {**A**[i],**A**[i +1],…, **A**[i +2^k]}
- Array **Power-of-2[]**
- Array **Log[]**



Range-minima-(i,j)

{   **L** ← $j - i$;

    t ← **Power-of-2**[L];

    k ← **Log**[**L**];

    If (t = **L**) return **B**[i][k];

    else return min(**B**[i][k] , **B**[$j - t$][k];);

}

Range-Minima Problem:

Data structure with **O**$(n$ **log** $n)$ **space** and **O**(1) **query time**

Theorem: There is a data structure for range-minima problem that takes **O**$(n$ **log** $n)$ **space** and **O**(1) **query time.**

Preprocessing time:

O(n^2) : Trivial

O$(n$ **log** $n)$: Doable with little hints

Range Minima Problem: further extensions

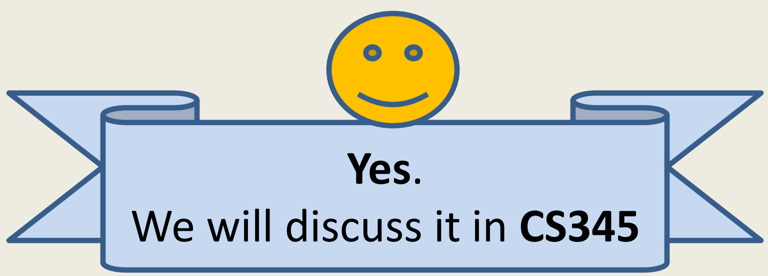
Dynamic Range Minima Problem:

- ❑ **O**(**log** n) update and query time.

Extension to **2-dimensions** ?

- ❑ **O**(**log** n) query time.

Question: Can we achieve **O**(n) space and **O**(1) query time ?



Data structures

(To be discussed in the course)

- Arrays
- Lists
- Stacks
- Queues

} Elementary

Tree Data Structures:

- ➢ Binary heap
- ➢ Binary Search Trees
- ➢ Augmented Data structures

Data Structures for integers:

- ➢ Hash Tables
- ➢ Searching in **O(log log** n) time

PROOF OF CORRECTNESS OF ALGORITHMS

GCD

```
GCD(a,b)     // a,b are two positive integers and a ≥ b
{
    while (b <> 0)
    {     t ← b;
          b ← a mod b ;
          a ← t
    }
    return a;
}
```

Question: What is the proof that **GCD**(a,b) returns greatest common divisor of a and b ?

Sum of first n positive integers

```
Sum(n)     // n is  a positive integers ≥ 1
{     sum ← 0;
          i ← 1;
      while (i ≤ n)
      {     sum ← sum + i;
            i ← i + 1;
      }
      return sum;
}
```

Question: What is the proof that **Sum**(n) returns the sum of first n positive integers ?

Sincerely attempt these questions before the next class …