Data Structures and Algorithms

(ESO207)

Lecture 7:

- **Data structure for Range-minima problem**

  Compact and fast

Data structures

AIM:

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**

$O(n \log n)$ time to build sorted array.

- Use **binary search** for answering query

$O(\log n)$ time per query

Data structures

AIM:

To organize a data in the memory

so that any query can be answered efficiently.

Important assumption:

No. of queries to be answered will be **many**.

Parameters of Efficiency

- Query time
- Space
- Preprocessing time

RANGE-MINIMA Problem

Range-Minima Problem

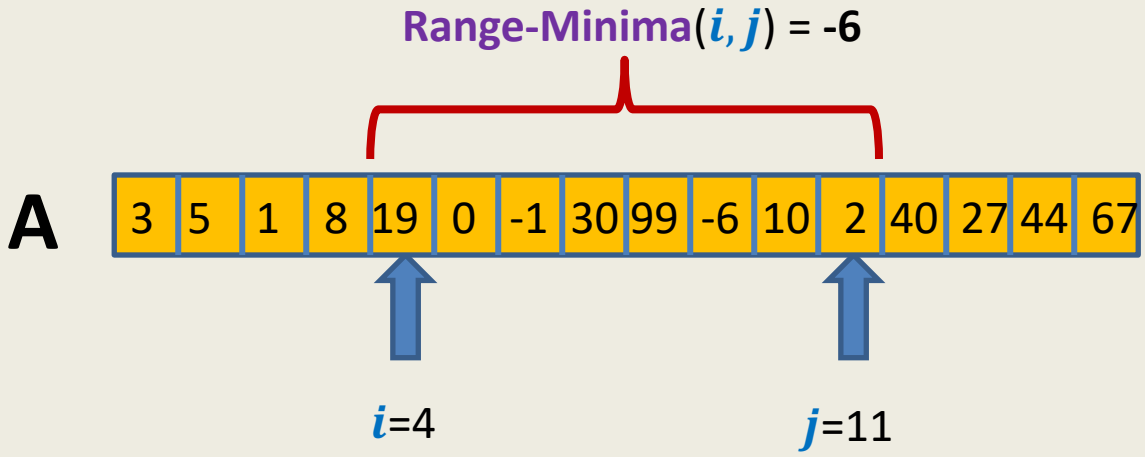
Given: an array **A** storing n numbers,

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**



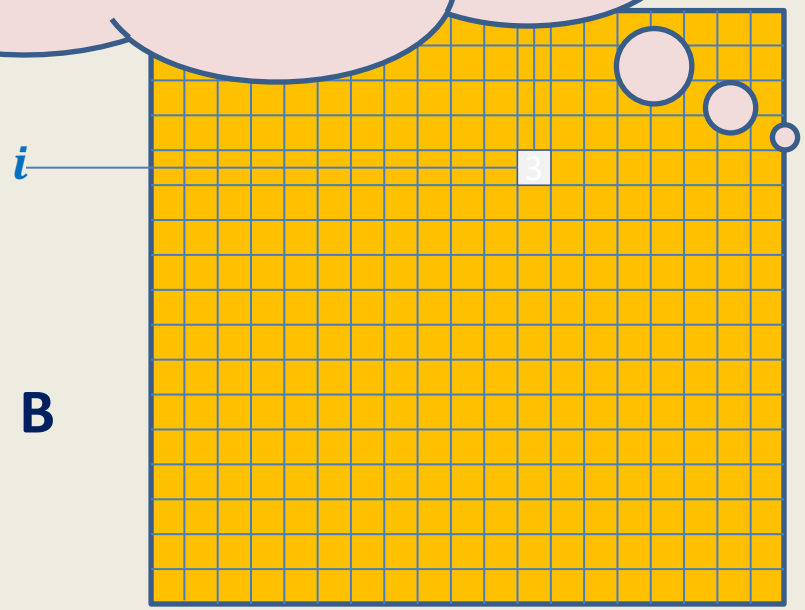
Range-Minima Problem

Solution 1

(brute force)

Size of **B** is **too large** to be kept in RAM.
So we shall have to keep most of it in the **Hard disk drive.**
Hence it will take a few **milliseconds per query**.

```
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 one query: O(n)

(a few **hours** for 10 million queries)

Space : O(n^2)

Impractical

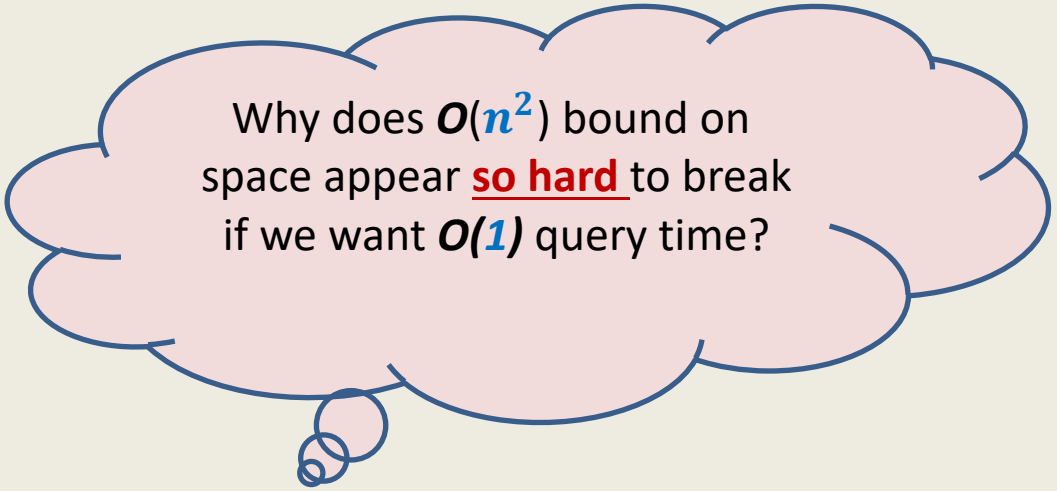
Range-Minima Problem

Query:

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

Aim :

- **compact** data structure
- **O(1) Query time** for any $1 \le i < j \le n$.



... 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 $O(n^2)$ queries,
so $O(n^2)$ 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

➔

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

A wrong inference

because it assumes that data structure for an index i will work in **total isolation** of others.

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** $j > i$.

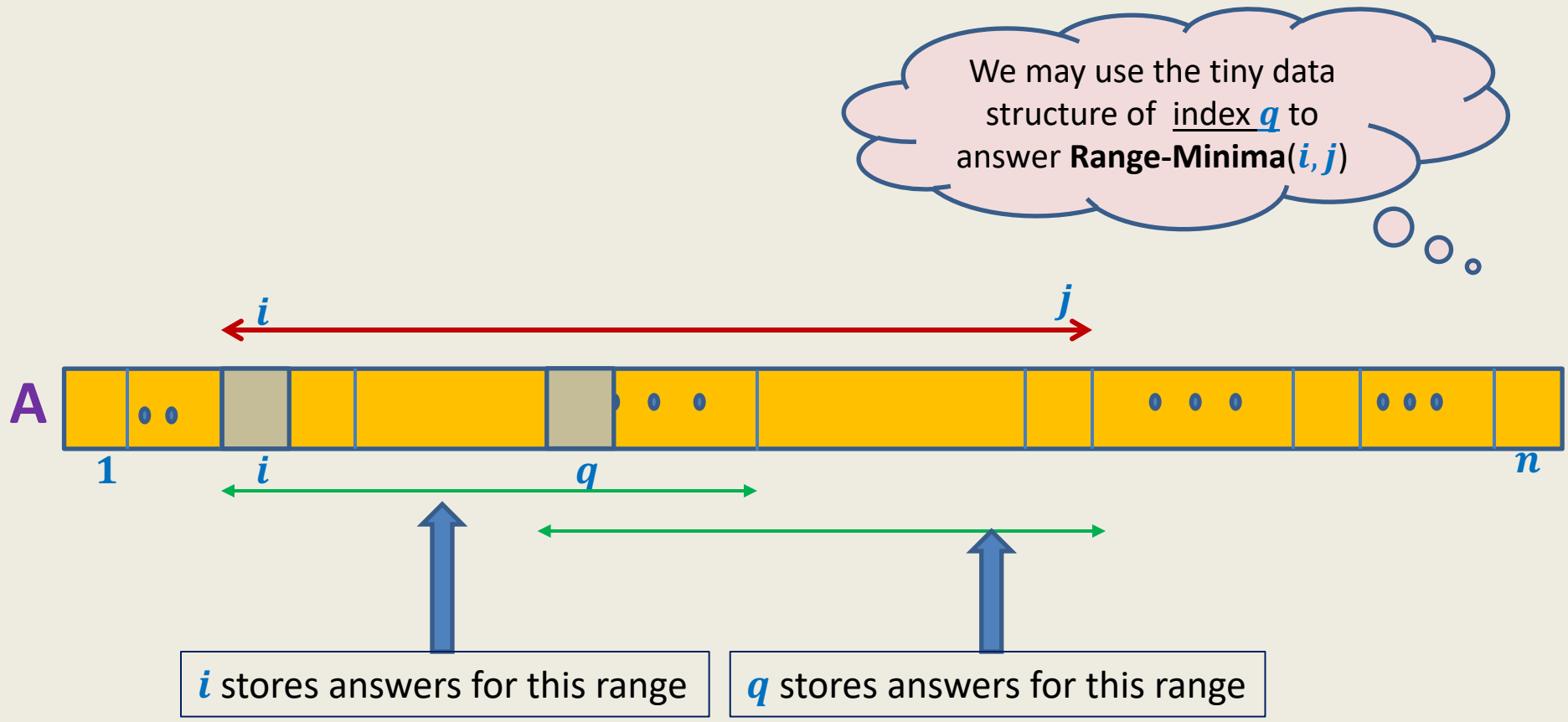
- For a query **Range-minima**(i, j),

  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 ?

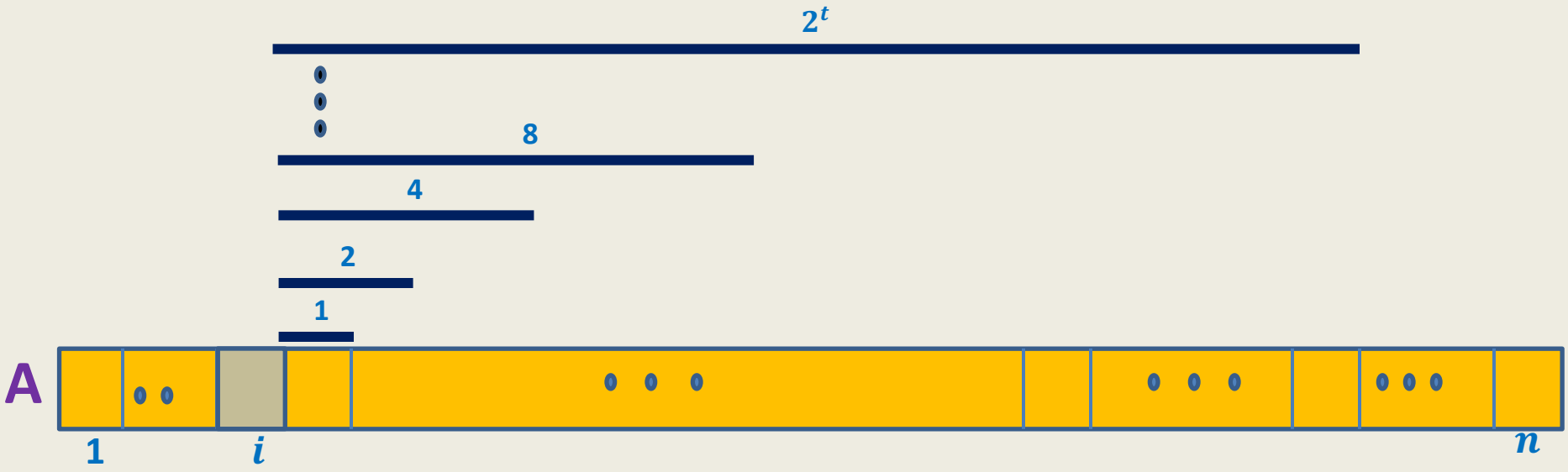
Range-minima problem:
Breaking the $O(n^2)$ barrier using collaboration



DETAILS OF TINY DATA STRUCTURES

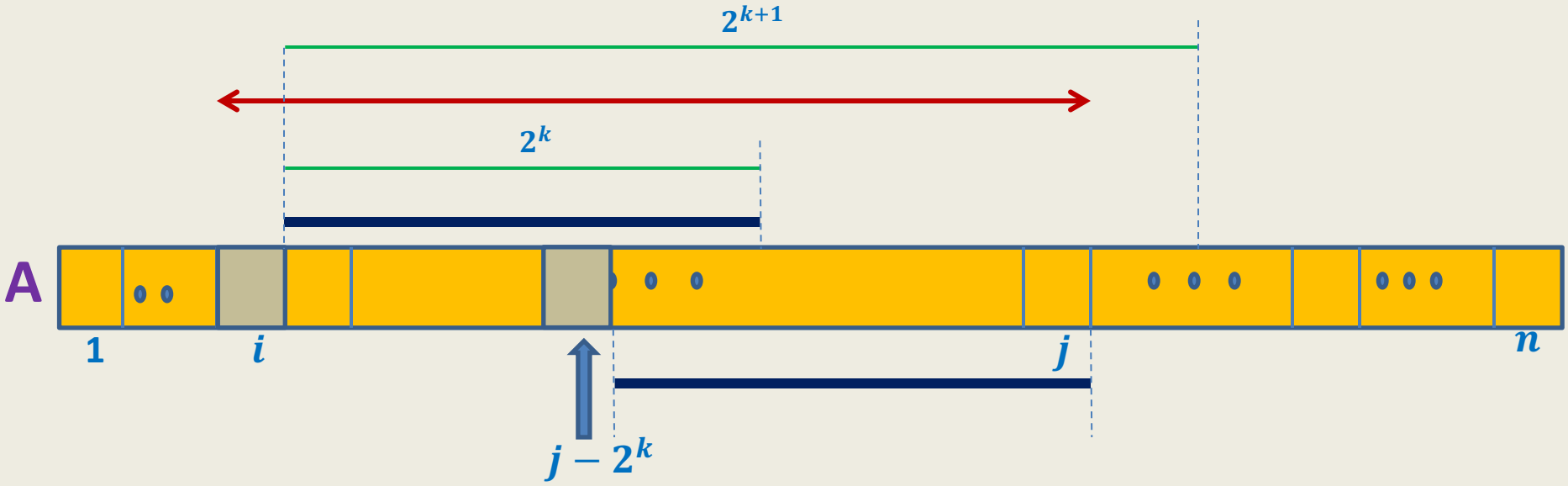
Range-minima problem :

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 $k \leq \log_2 n$

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 $2^k \leq m$.

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 $2^k \leq m$.

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** ← $j - i$;

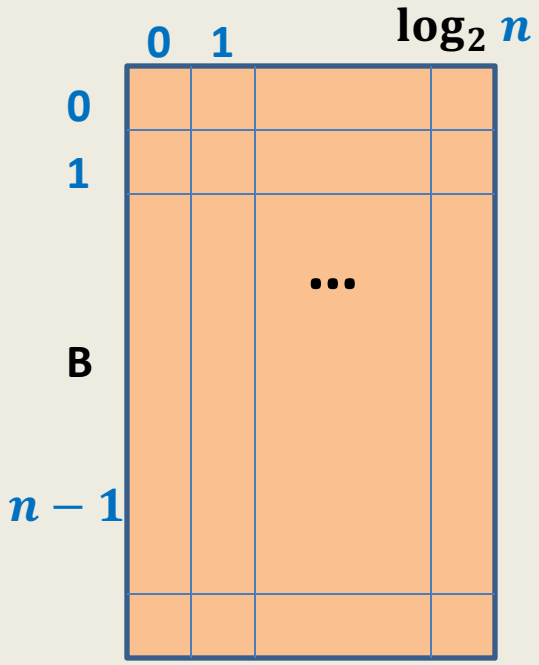
t ← **Power-of-2**[**L**];

k ← **Log**[**L**];

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

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

}



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$(n^2$ **log** $n)$: Trivial

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

Homework:

Design an **O**(n **log** n) time algorithm

to build the n **×** **log** n matrix **B** used in data structure of Range-Minima problem.

Hint: (Inspiration from iterative algorithm for Fibonacci numbers).

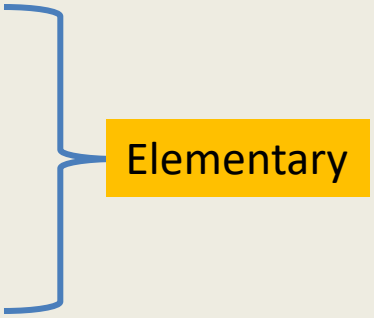
Spend some time before looking at the *more explicit hint* below. (it is just a click away)…You can do it…

To compute **B**[i][k], you need to know only two entries from column $**$.

Data structures

(To be discussed in the course)

- Arrays
- Linked Lists
- Stacks
- Queues



Tree Data Structures:

- Binary heap
- Binary Search Trees
- Augmented Data structures

Data Structures for integers:

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