STDISCM: Threads & Task Partitioning

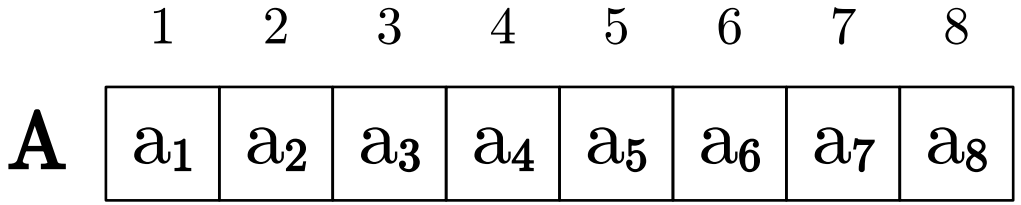
Task Partitioning

How should we divide the task into subtasks?

How should we distribute the subtasks to the threads?

Task: Find a prime number

Task: Given an array of numbers **A**, determine if there is a prime number in **A**. If there is, return the prime number and its location/index.



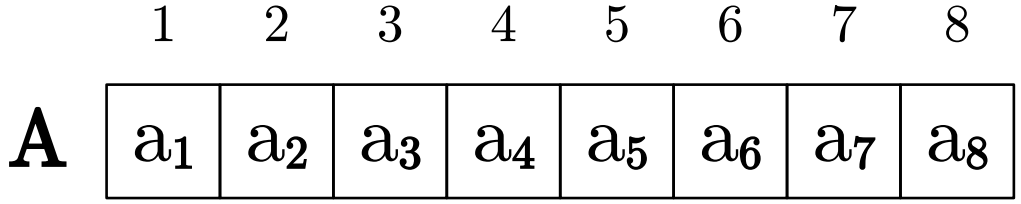
Task:

a_1 prime?
a_2 prime?
a_3 prime?
a_4 prime?
a_5 prime?
a_6 prime?
a_7 prime?
a_8 prime?

Worst Case: done in 8 steps

Task: Find a prime number

Task: Given an array of numbers **A**, determine if there is a prime number in **A**. If there is, return the prime number and its location/index.



Subtask 1:

a_1 prime?
a_2 prime?
a_3 prime?
a_4 prime?

Subtask 2:

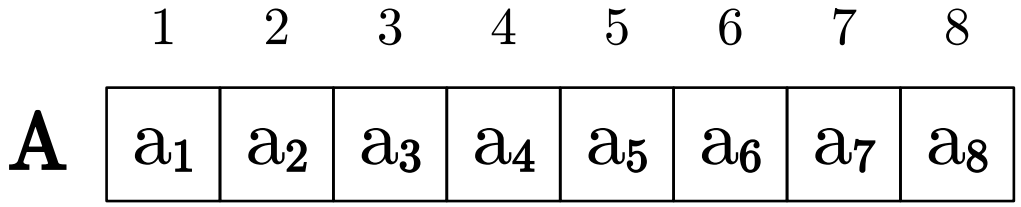
a_5 prime?
a_6 prime?
a_7 prime?
a_8 prime?

Subtask 1 and Subtask 2 run at the same time (in parallel)

Worst Case: done in 4 steps

Task: Find a prime number

Task: Given an array of numbers **A**, determine if there is a prime number in **A**. If there is, return the prime number and its location/index.



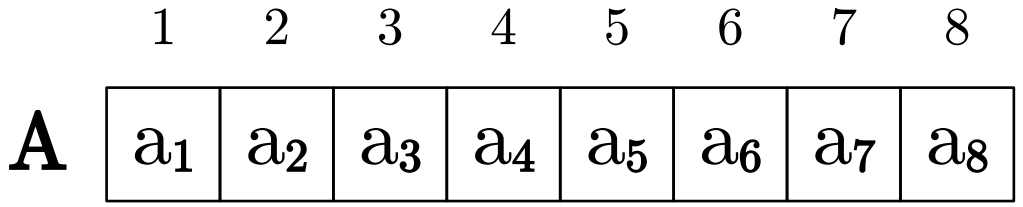
Subtask 1:	Subtask 2:	Subtask 3:	Subtask 4:
a_1 prime?	a_3 prime?	a_5 prime?	a_7 prime?
a_2 prime?	a_4 prime?	a_6 prime?	a_8 prime?

Subtasks 1, 2, 3, and 4 run at the same time (in parallel)

Worst Case: done in 2 steps

Task: Find a prime number

Task: Given an array of numbers **A**, determine if there is a prime number in **A**. If there is, return the prime number and its location/index.



Subtask 1: a_1 prime?

Subtask 2: a_2 prime?

Subtask 3: a_3 prime?

Subtask 4: a_4 prime?

Subtask 5: a_5 prime?

Subtask 6: a_6 prime?

Subtask 7: a_7 prime?

Subtask 8: a_8 prime?

Subtasks 1-8 run at the same time (in parallel)

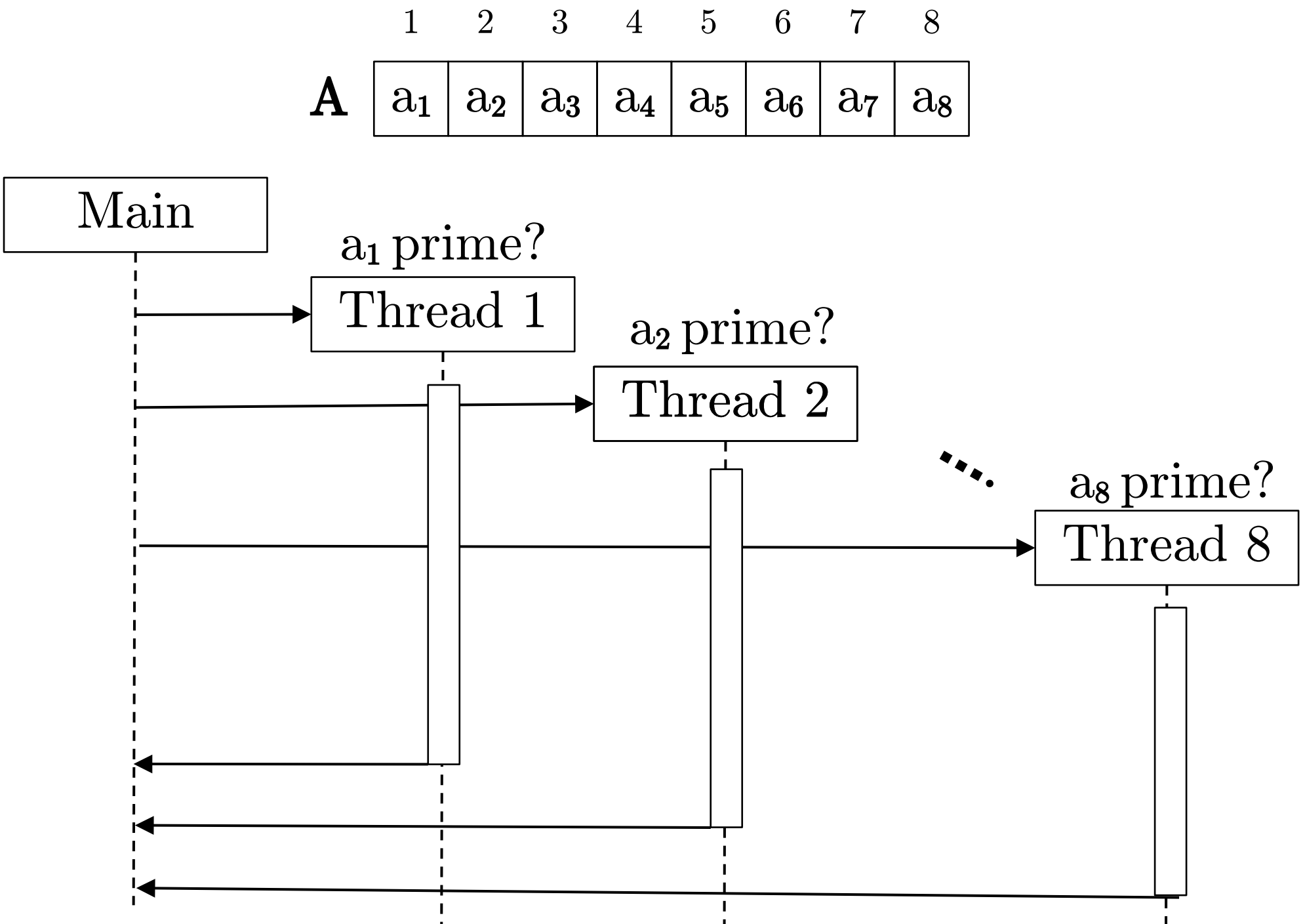
Worst Case: done in 1 steps

Tasking Partitioning: Finding a Prime

Partitioning:
Subtask – Finding a prime number for a smaller (sub) array of numbers

Task Distribution: One subtask per one thread/worker.

Tasking Partitioning: Finding a Prime



Tasking Partitioning: Finding a Prime

Partitioning:
Subtask – Finding a prime number for a smaller (sub) array of numbers

Task Distribution: One subtask per one thread/worker.

Question:

One of the smaller subtask involves finding a prime prime number for an array of size 1. This substask is basically a primality test/check for a single number.

Can this subtask be broken down further into smaller Subtasks that can be parallelize?

Quick Answer: Yes!

Primality Checking

```
bool isPrime(int n)
{
  bool is_prime = true;
  int  i        = 0;

  if ( n == 0 || n == 1 ){ return false; }

  for (i = 2; i < n-1; i++)
  {
    if (n % i == 0)
    {
      is_prime = false;
      break;
    }
  }
  return is_prime;
}
```

Primality Checking

```
bool isPrime(int n)
{
  bool is_prime = true;
  int  i        = 0;

  if ( n == 0 || n == 1 ){ return false; }

  for (i = 2; i < n/2; i++)
  {
    if (n % i == 0)
    {
      is_prime = false;
      break;
    }
  }
  return is_prime;
}
```

Primality Checking

```
bool isPrime(int n)
{
  bool is_prime = true;
  int  sqrt_n   = 1;
  int  i        = 0;

  if ( n == 0 || n == 1 ){ return false; }

  sqrt_n = ceil(sqrt(n));
  for (i = 2; i <= sqrt_n; i++)
  {
    if (n % i == 0)
    {
      is_prime = false;
      break;
    }
  }
  return is_prime;
}
```

Primality Checking

```
100 =  1 * 100
100 =  2 * 50
100 =  4 * 25
100 = 10 * 10
(Note: 11 * 11 = 121 > 100)

10000 =   1 * 10000
10000 =   2 * 5000
10000 =   4 * 2500
10000 =  10 * 1000
10000 =  16 * 625
10000 =  20 * 500
10000 =  25 * 400
10000 =  40 * 250
10000 =  50 * 200
10000 =  80 * 125
10000 = 100 * 100
(Note: 101 * 101 = 10201 > 10000)
```

Primality Checking Parallelized

Given the (naive) algorithm above, the primality of the number ***n***, is check by checking if any of the numbers from ***2*** to ***sqrt(n)*** divides ***n***. If none of those numbers divides **n**, then n is prime.

Checking if any number from ***2*** to ***sqrt(n)*** divides ***n*** can can parallelize.

n % 2 == 0?
n % 3 == 0?
n % 4 == 0?
...
n % sqrt(n) == 0?

Tasking Partitioning: Finding a Prime