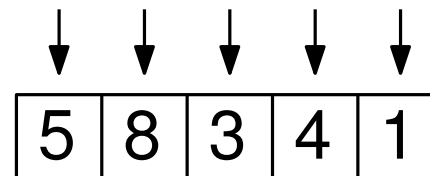
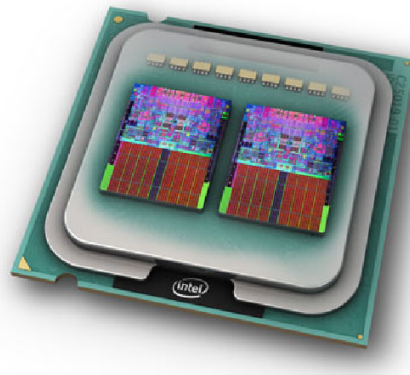




ICS 643: Advanced Parallel Algorithms

Prof. Nodari Sitchinava



Lecture 1: Introduction

Welcome to ICS 643

Course Webpage:

<https://www2.hawaii.edu/~nodari/teaching/s22-643/>

But don't take my word for it...

From students' Course Evaluations:

“Brushing up on run time analysis, proof of correctness (induction), mergesort, and linked lists/trees could be handy.”

ICS 443, Fall 2017

“Also do NOT cheat in this class. Nodari means it when he says that he doesn't take cheating lightly and unfortunately class mates of mine have been reported to the office of judicial affairs for academic dishonesty this semester.”

ICS 311, Spring 2020

“Go to office hours when possible AFTER reviewing the HW and ask for clarifications.”

ICS 311, Spring 2020

Take this course, if...

Take this course, if...

- You enjoyed ICS 311

Take this course, if...

- You enjoyed ICS 311
- You want to improve your algorithm design and analysis skills
 - Good algorithmic skills make better programmers

Take this course, if...

- You enjoyed ICS 311
- You want to improve your algorithm design and analysis skills
 - Good algorithmic skills make better programmers
- Be one of the few people who know how to solve problems by properly taking advantage of available parallelism in modern and future systems

This course is probably not for you if...

This course is probably not for you if...

- You hated ICS 311

This course is probably not for you if...

- You hated ICS 311
- You just want to pass to earn 3 units of credit

This course is probably not for you if...

- You hated ICS 311
- You just want to pass to earn 3 units of credit
- If you are looking for a parallel programming course, take ICS 432/632 instead

Design an algorithm...

m: A finite sequence of well-defined (simple) instructions to complete a task.

Design an algorithm...

m: A finite sequence of well-defined (simple) instructions to complete a task.

```
procedure DIRECTIONS(A, B)  
  if A == ICS and B == Waikiki then  
    Head east for 374 ft  
    Turn right, proceed for 341 ft  
    Turn left, proceed for 0.6 mi  
    Turn right, proceed for 0.1 mi  
    Turn right, proceed for 0.1 mi  
    Continue for 1.3 mi  
    Turn left, proceed for 0.2 mi  
    Turn left, proceed for 0.2 mi  
    Continue straight for 0.2 mi
```

Design an algorithm...

m: A finite sequence of well-defined (simple) instructions to complete a task.

procedure DIRECTIONS(*A*, *B*)

if *A* == *ICS* **and** *B* == *Waikiki* **then**

Head east for 374 ft

Turn right, proceed for 341 ft

Turn left, proceed for 0.6 mi

Turn right, proceed for 0.1 mi

Turn right, proceed for 0.1 mi

Continue for 1.3 mi

Turn left, proceed for 0.2 mi

Turn left, proceed for 0.2 mi

Continue straight for 0.2 mi

▼ Take Dole St to Waialae Ave

4 min (0.9 mi)

↑ Head east on Pope Rd toward East-West Rd
374 ft

➤ Turn right onto East-West Rd
341 ft

↶ Turn left at the 1st cross street onto Dole St
0.6 mi

➤ Turn right onto St Louis Dr
0.1 mi

▼ Take Kapiolani Blvd to Kalakaua Ave

9 min (1.9 mi)

➤ Turn right onto Waialae Ave
0.1 mi

↑ Continue onto Kapiolani Blvd
1.3 mi

↶ Turn left onto McCully St
0.2 mi

↶ Use any lane to turn left onto Ala Moana Blvd/Kalakaua Ave
[Continue to follow Kalakaua Ave](#)

0.2 mi

↑ Continue straight to stay on Kalakaua Ave
0.2 mi

Design an algorithm...

m: A finite sequence of well-defined (simple) instructions to complete a task.

procedure DIRECTIONS(*A*, *B*)

if *A* == *ICS* **and** *B* == *Waikiki* **then**

Head east for 374 ft

Turn right, proceed for 341 ft

Turn left, proceed for 0.6 mi

Turn right, proceed for 0.1 mi

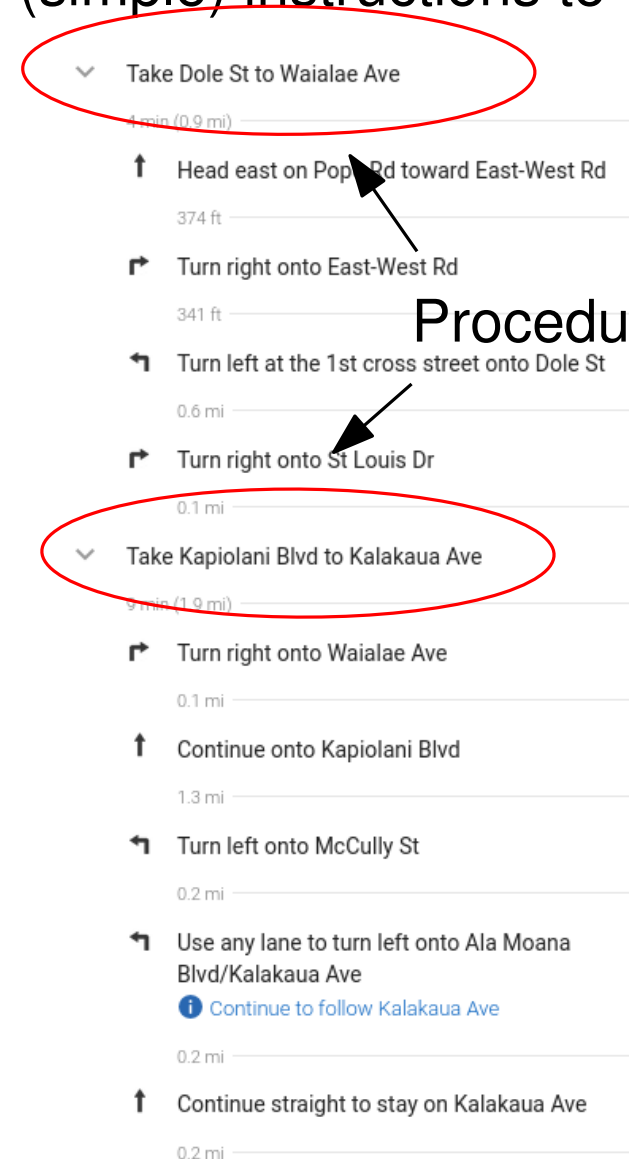
Turn right, proceed for 0.1 mi

Continue for 1.3 mi

Turn left, proceed for 0.2 mi

Turn left, proceed for 0.2 mi

Continue straight for 0.2 mi



Procedure calls

Design an algorithm...

m: A finite sequence of well-defined (simple) instructions to complete a task.

procedure D

if $A == IC$

Head e

Turn rig

Turn let

Turn rig

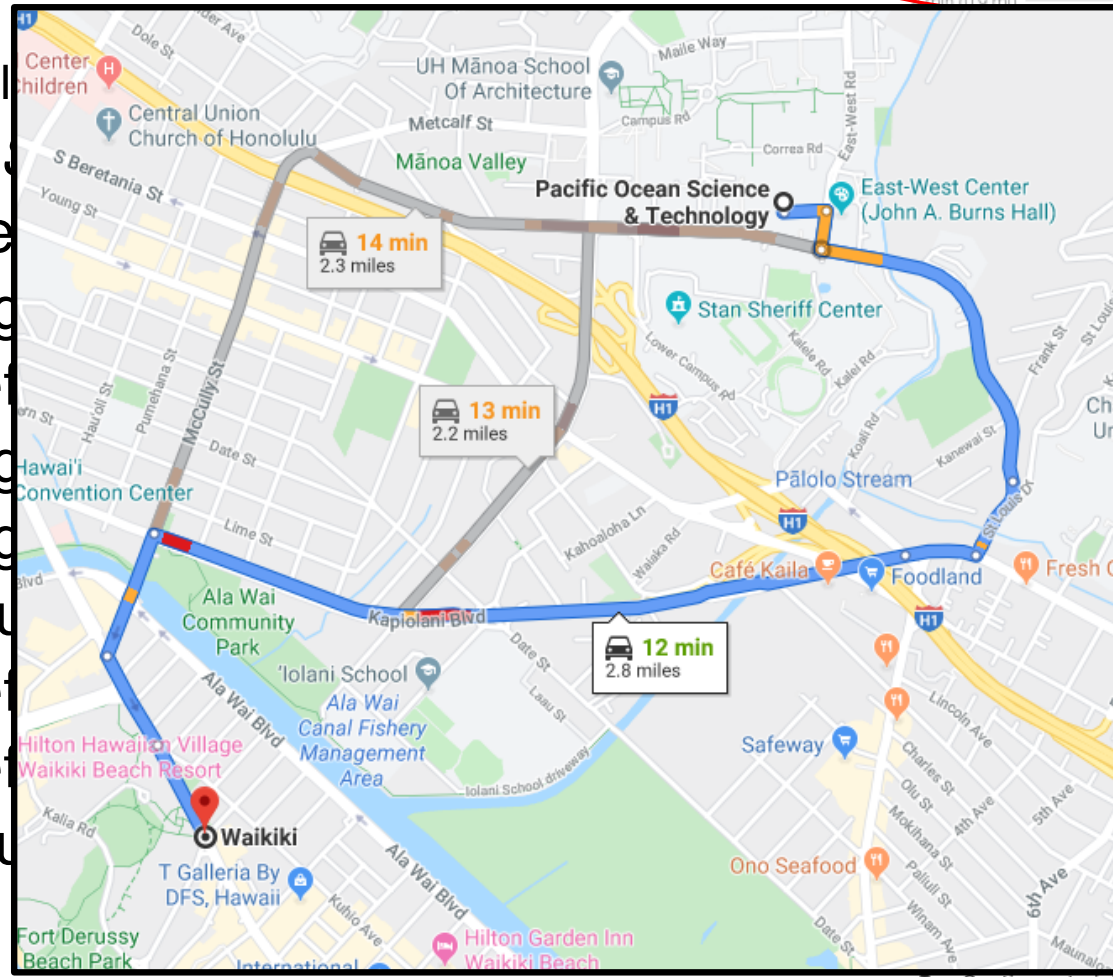
Turn rig

Continu

Turn let

Turn let

Continu



Take Dole St to Waialae Ave

Procedure calls

Continue straight to stay on Kalakaua Ave

0.2 mi

Design an algorithm...

When asked to “design an algorithm”, you should:

Design an algorithm...

When asked to “design an algorithm”, you should:

1. Describe/explain your algorithm in plain English
 - Draw pictures to aid the explanation
2. Write down the pseudocode
3. Prove its correctness
4. Analyze its running time

Parallel Algorithms

Understanding Runtime

```
procedure FOO()
```

```
  a = 7
```

```
  b = 20
```

```
  c = 14
```

```
  d = 27
```

```
  e = 15
```

Understanding Runtime

procedure FOO()

→ a = 7
b = 20
c = 14
d = 27
e = 15

Understanding Runtime

procedure FOO()

a = 7

→ b = 20

c = 14

d = 27

e = 15

Understanding Runtime

procedure FOO()

 a = 7

 b = 20

→ c = 14

 d = 27

 e = 15

Understanding Runtime

procedure FOO()

a = 7

b = 20

c = 14

→ d = 27

e = 15

Understanding Runtime

procedure FOO()

a = 7

b = 20

c = 14

d = 27

→ e = 15

Understanding Runtime

```
procedure FOO()
```

```
  a = 7
```

```
  b = 20
```

```
  c = 14
```

```
  d = 27
```

```
  e = 15
```

Runtime = 5 steps

Understanding Runtime

procedure FOO()

a = 7

b = 20

c = 14

d = 27

e = 15

Runtime = 5 steps

procedure PARALLELFOO()

a = 7

spawn {
 b = 20
 c = 14

}

d = 27

e = 15

sync

Understanding Runtime

procedure FOO()

a = 7

b = 20

c = 14

d = 27

e = 15

Runtime = 5 steps

procedure PARALLELFOO()

a = 7

spawn 

b = 20

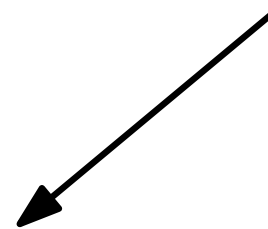
c = 14



d = 27

e = 15

sync



Understanding Runtime

procedure FOO()

a = 7

b = 20

c = 14

d = 27

e = 15

Runtime = 5 steps

procedure PARALLELFOO()



a = 7

spawn



b = 20

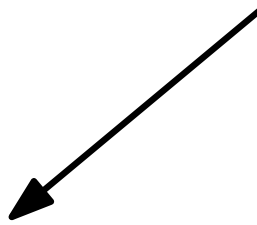
c = 14



d = 27

e = 15

sync



Understanding Runtime

procedure FOO()

a = 7

b = 20

c = 14

d = 27

e = 15

Runtime = 5 steps

procedure PARALLELFOO()

a = 7



spawn



b = 20

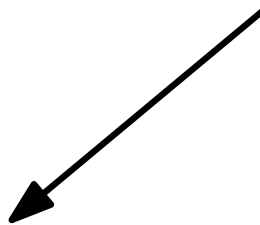
c = 14



d = 27

e = 15

sync



Understanding Runtime

procedure FOO()

a = 7

b = 20

c = 14

d = 27

e = 15

Runtime = 5 steps

procedure PARALLELFOO()

a = 7

spawn

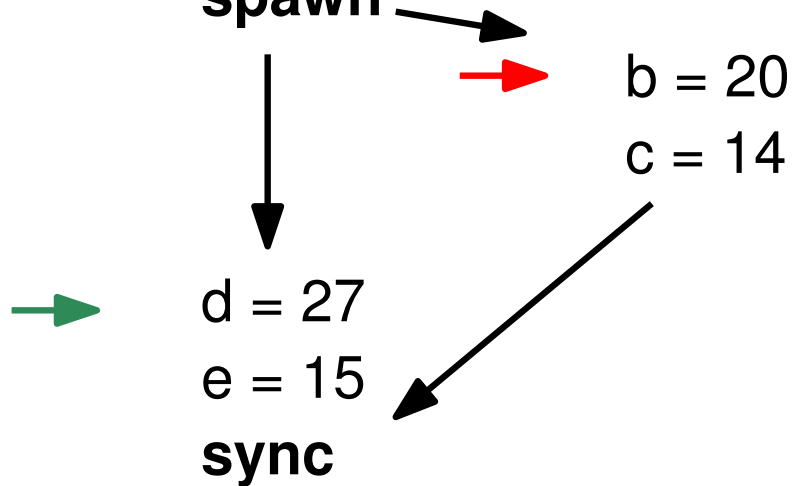
b = 20

c = 14

d = 27

e = 15

sync



Understanding Runtime

procedure FOO()

a = 7

b = 20

c = 14

d = 27

e = 15

Runtime = 5 steps

procedure PARALLELFOO()

a = 7

spawn



b = 20

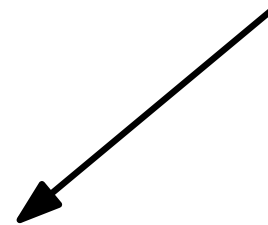
c = 14



d = 27

e = 15

sync



Understanding Runtime

procedure FOO()

a = 7

b = 20

c = 14

d = 27

e = 15

Runtime = 5 steps

procedure PARALLELFOO()

a = 7

spawn



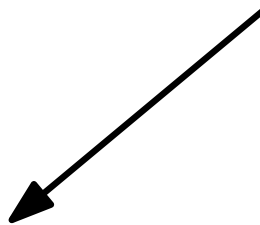
d = 27

e = 15

sync

b = 20

c = 14



Understanding Runtime

procedure FOO()

a = 7

b = 20

c = 14

d = 27

e = 15

Runtime = 5 steps

procedure PARALLELFOO()

a = 7

spawn →



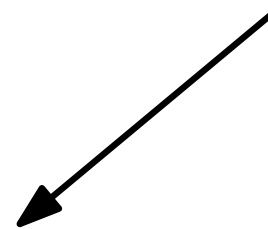
d = 27

e = 15

sync

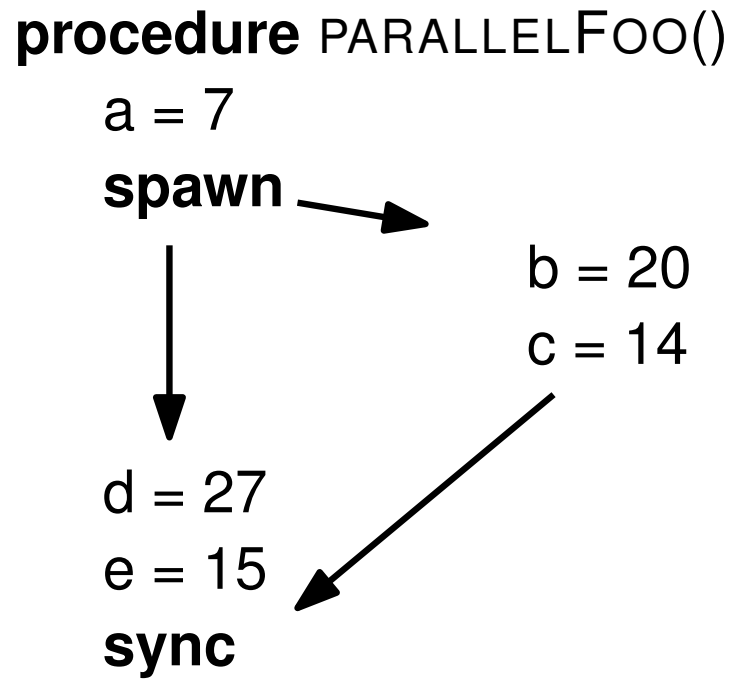
b = 20

c = 14

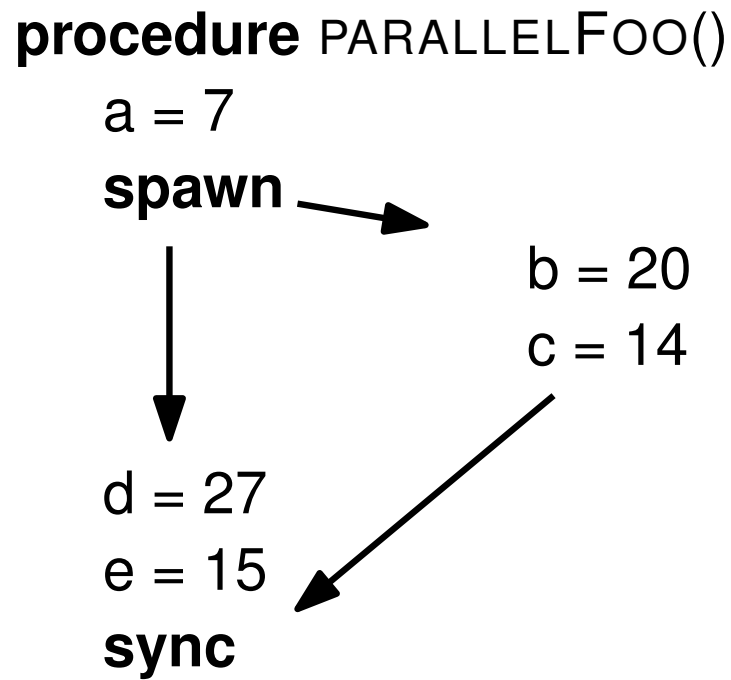


Runtime = 5 steps

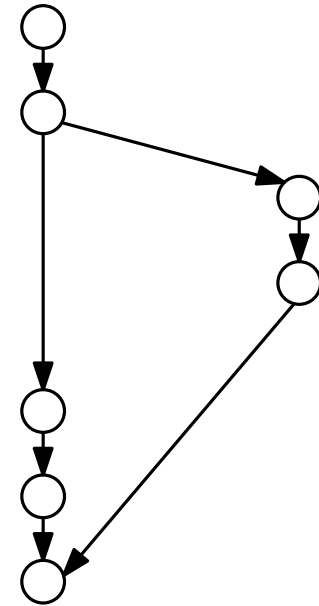
Parallel Execution as a DAG



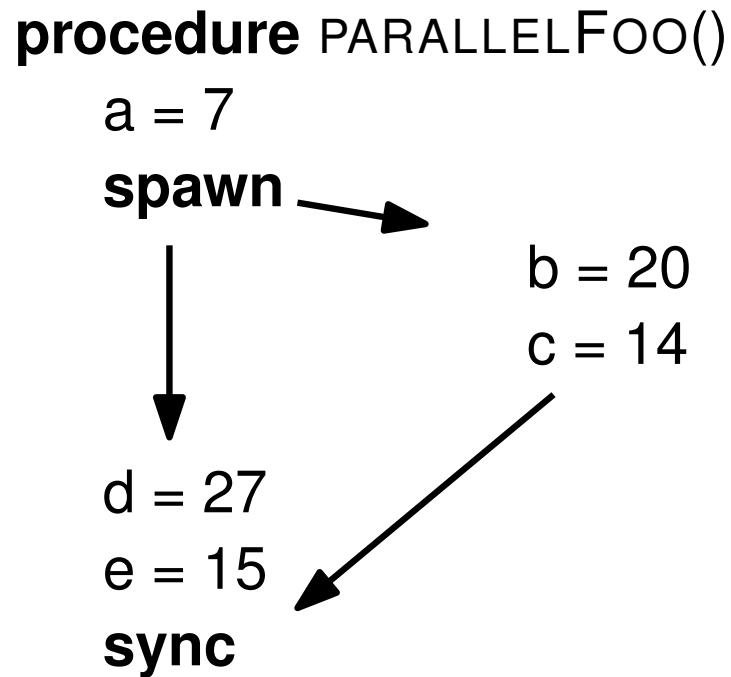
Parallel Execution as a DAG



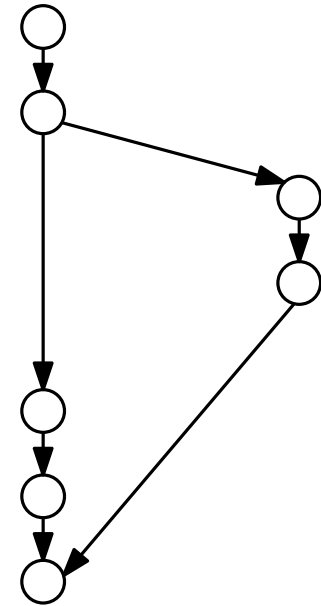
Dependency graph



Parallel Execution as a DAG

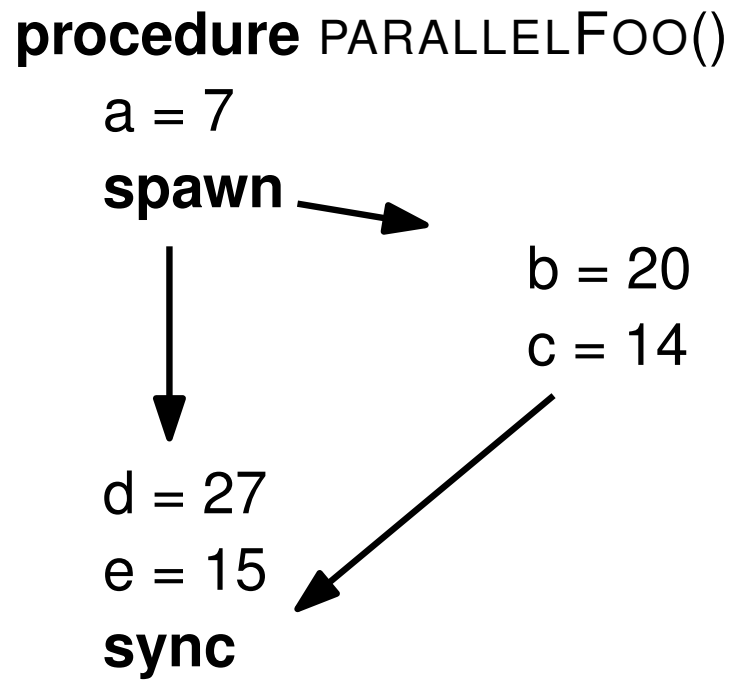


Dependency graph



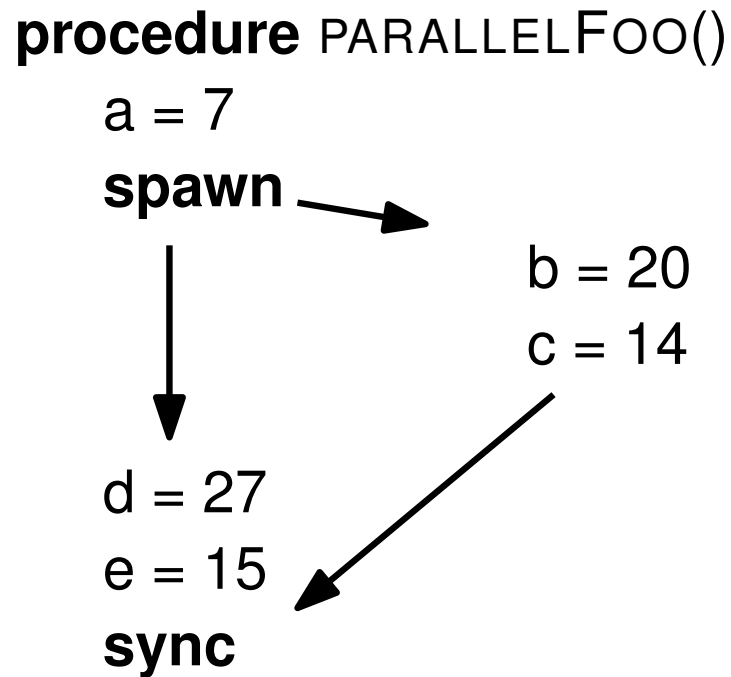
Parallel runtime = longest path length in the dependency graph

Understanding Parallel Runtime



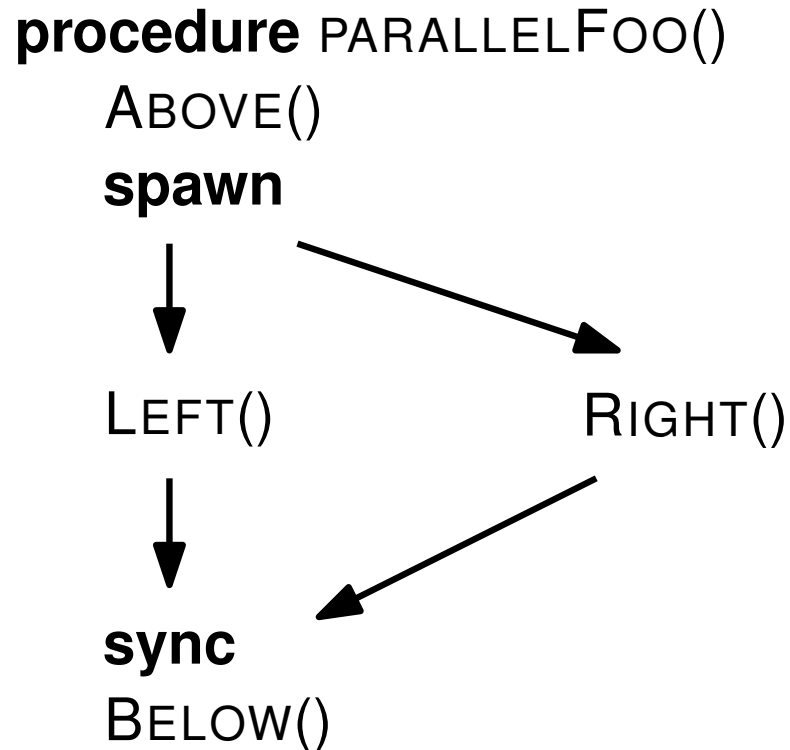
Understanding Parallel Runtime

procedure PARALLELFOO()
 a = 7
 spawn →
 ↓
 d = 27
 e = 15
 sync ←
 b = 20
 c = 14



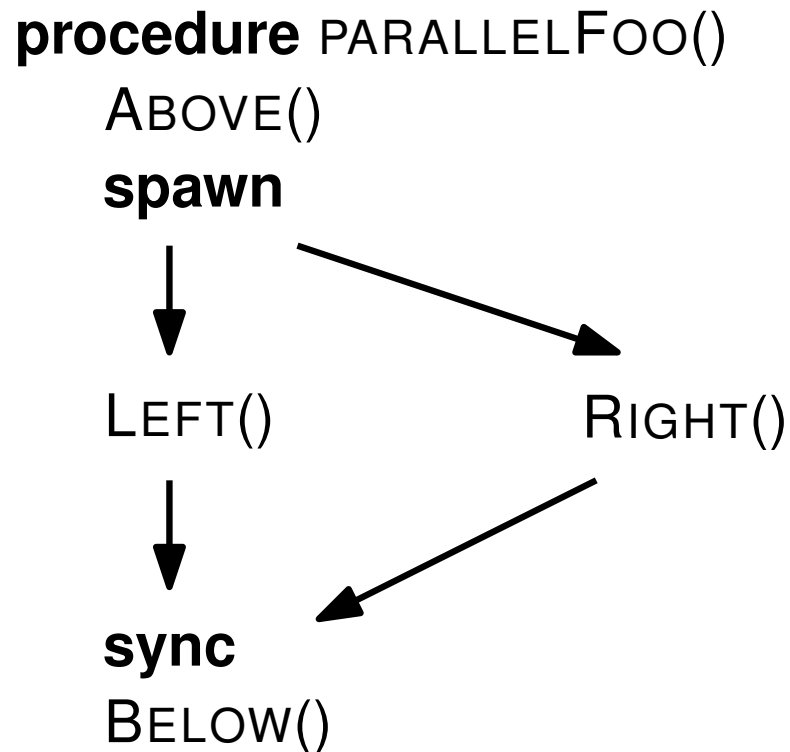
procedure ABOVE()
 a = 7
procedure LEFT()
 d = 27
 e = 15
procedure RIGHT()
 b = 20
 c = 14
procedure BELOW()

Understanding Parallel Runtime



procedure ABOVE()
 a = 7
procedure LEFT()
 d = 27
 e = 15
procedure RIGHT()
 b = 20
 c = 14
procedure BELOW()

Understanding Parallel Runtime

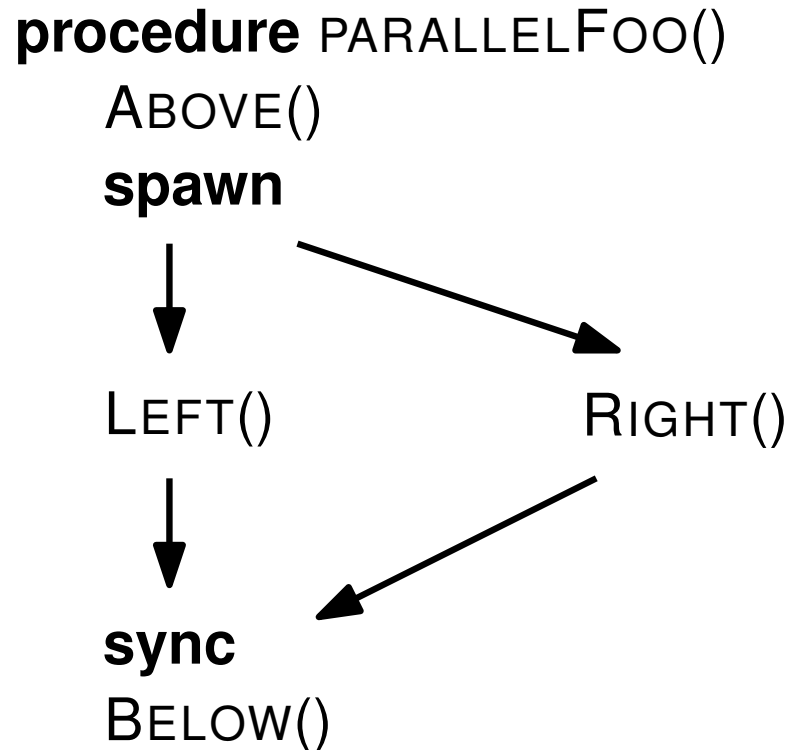


procedure ABOVE()
 a = 7
procedure LEFT()
 d = 27
 e = 15
procedure RIGHT()
 b = 20
 c = 14
procedure BELOW()

Parallel Runtime

$$\begin{aligned} & T(\text{ABOVE}) + 1 + \max(T(\text{LEFT}), T(\text{RIGHT})) + 1 + T(\text{BELOW}) \\ &= T(\text{ABOVE}) + \max \left\{ \begin{array}{l} T(\text{LEFT}) \\ T(\text{RIGHT}) \end{array} \right\} + T(\text{BELOW}) + O(1) \end{aligned}$$

Understanding Parallel Runtime



procedure ABOVE()
 a = 7
procedure LEFT()
 d = 27
 e = 15
procedure RIGHT()
 b = 20
 c = 14
procedure BELOW()

Parallel Runtime

$$\begin{aligned} & T(\text{ABOVE}) + 1 + \max(T(\text{LEFT}), T(\text{RIGHT})) + 1 + T(\text{BELOW}) \\ &= T(\text{ABOVE}) + \max \left\{ \begin{array}{l} T(\text{LEFT}) \\ T(\text{RIGHT}) \end{array} \right\} + T(\text{BELOW}) + O(1) \\ &= T(\text{ABOVE}) + \max \left\{ \begin{array}{l} T(\text{LEFT}) \\ T(\text{RIGHT}) \end{array} \right\} + T(\text{BELOW}) \end{aligned}$$

Proper Pseudocode

```
procedure PARALLELFOO()  
  ABOVE()  
  spawn  
    ↓  
    LEFT()  
    ↓  
    sync  
    BELOW()  
    ↗  
  RIGHT()
```

```
graph TD; spawn[spawn] --> LEFT[LEFT()]; spawn --> RIGHT[RIGHT()]; LEFT --> sync[sync]; sync --> BELOW[BELOW()]; RIGHT --> sync;
```

Proper Pseudocode

```
procedure PARALLELFOO()  
  ABOVE()  
  spawn  
    ↓  
  LEFT()  
    ↓  
  sync  
  BELOW()
```

```
graph TD; A[spawn] --> B[LEFT()]; A --> C[RIGHT()]; B --> D[sync]; D --> E[BELOW()];
```

```
procedure PARALLELFOO()  
  ABOVE()  
  spawn RIGHT()  
  LEFT()  
  sync  
  BELOW()
```

Proper Pseudocode

```
procedure PARALLELFOO()  
  ABOVE()  
  spawn  
    ↓  
  LEFT()  
    ↓  
  sync  
  BELOW()
```

```
graph TD; A[spawn] --> B[LEFT()]; A --> C[RIGHT()]; B --> D[sync]; D --> E[BELOW()];
```

```
procedure PARALLELFOO()  
  ABOVE()  
  spawn RIGHT()  
  LEFT()  
  sync  
  BELOW()
```

```
procedure PARALLELFOO()  
  ABOVE()  
  in parallel do  
    RIGHT()  
    LEFT()  
  BELOW()
```

Taking a step further

```
procedure PARALLELFOO()  
  ABOVE()  
  in parallel do  
    RIGHT()  
    LEFT()  
  BELOW()
```

Taking a step further

```
procedure PARALLELFOO()  
  ABOVE()  
  in parallel do  
    RIGHT()  
    LEFT()  
  BELOW()
```

```
procedure PARALLELFOO()  
  ABOVE()  
  in parallel do  
    RIGHT()  
    MIDDLE()  
    LEFT()  
  BELOW()
```

Taking a step further

procedure PARALLELFOO()

ABOVE()

in parallel do

RIGHT()

LEFT()

BELOW()

procedure PARALLELFOO()

ABOVE()

in parallel do

RIGHT()

MIDDLE()

LEFT()

BELOW()

procedure PARALLELFOO()

ABOVE()

spawn

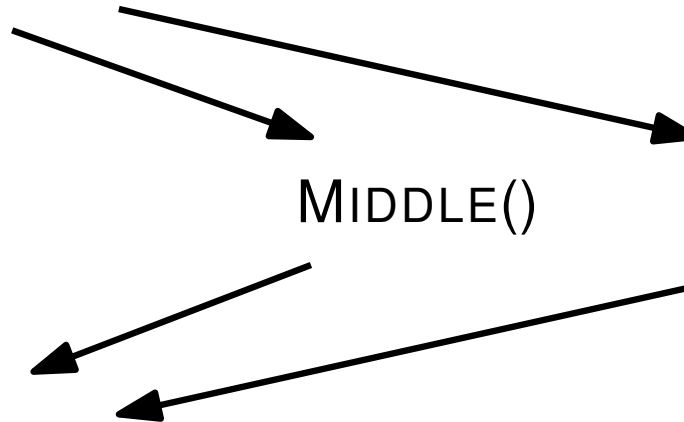


LEFT()



sync

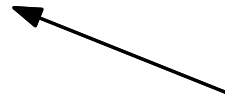
BELOW()



Parallel **for** loop

for $i = 1$ to n **in parallel do**

$a[i] = a[i] + 1$



Spawn n threads t_1, t_2, \dots, t_n

Each thread t_i (where $i = 1, 2, \dots, n$) **do**:

$a[i] = a[i] + 1$

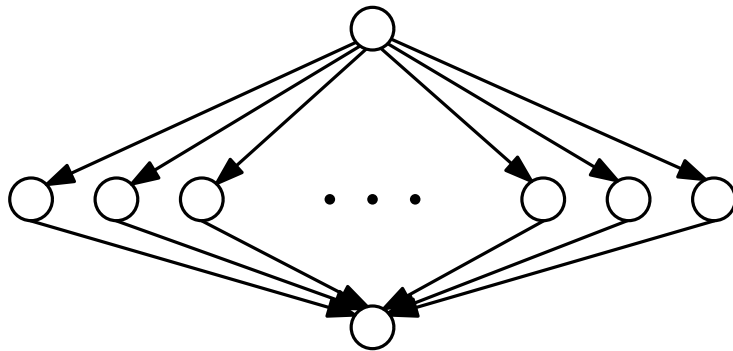
Synchronize all n threads

Parallel **for** loop

for $i = 1$ to n **in parallel do**

$a[i] = a[i] + 1$

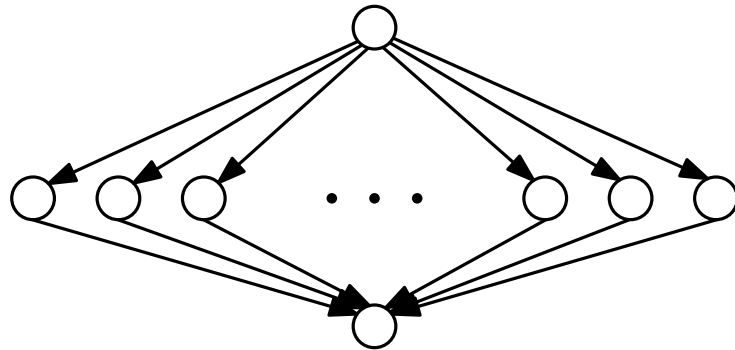
Spawn n threads t_1, t_2, \dots, t_n
Each thread t_i (where $i = 1, 2, \dots, n$) **do**:
 $a[i] = a[i] + 1$
Synchronize all n threads



Parallel **for** loop

for $i = 1$ to n **in parallel do**
 $a[i] = a[i] + 1$

Spawn n threads t_1, t_2, \dots, t_n
Each thread t_i (where $i = 1, 2, \dots, n$) **do**:
 $a[i] = a[i] + 1$
Synchronize all n threads



Parallel Runtime: $O(1)$

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

5	8	3	4	1
---	---	---	---	---

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

5	8	3	4	1
---	---	---	---	---



$i = 1$
L: $a[i] = a[i] + 1$
 $i = i + 1$
if $i \leq n$: JUMPTO L

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

$i = 1$



5	8	3	4	1
---	---	---	---	---



$i = 1$

L: $a[i] = a[i] + 1$

$i = i + 1$

if $i \leq n$: JUMPTO L

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

$i = 1$



6	8	3	4	1
---	---	---	---	---



$i = 1$

L: $a[i] = a[i] + 1$

$i = i + 1$

if $i \leq n$: JUMPTO L

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

$i = 2$
↓

6	8	3	4	1
---	---	---	---	---



$i = 1$
L: $a[i] = a[i] + 1$
 $i = i + 1$
 if $i \leq n$: JUMPTO L

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

$i = 2$



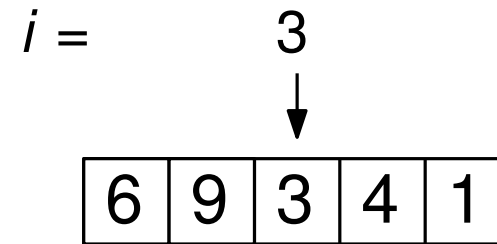
6	9	3	4	1
---	---	---	---	---



$i = 1$
L: $a[i] = a[i] + 1$
 $i = i + 1$
if $i \leq n$: JUMPTO L

Simple example

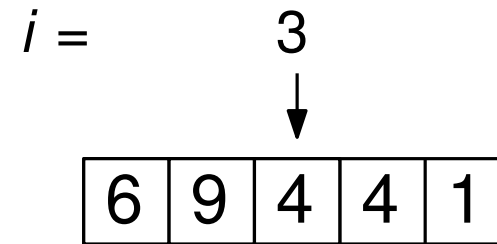
for $i = 1$ to n **do**
 $a[i] = a[i] + 1$



$i = 1$
L: $a[i] = a[i] + 1$
 $i = i + 1$
 if $i \leq n$: JUMPTO L

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$



$i = 1$
L: $a[i] = a[i] + 1$
 $i = i + 1$
 if $i \leq n$: JUMPTO L

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

$i =$

4



6	9	4	4	1
---	---	---	---	---



$i = 1$

L: $a[i] = a[i] + 1$

$i = i + 1$

if $i \leq n$: JUMPTO L

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

$i =$

4



6	9	4	5	1
---	---	---	---	---



$i = 1$

L: $a[i] = a[i] + 1$

$i = i + 1$

if $i \leq n$: JUMPTO L

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$



$i =$

5



6	9	4	5	1
---	---	---	---	---

$i = 1$

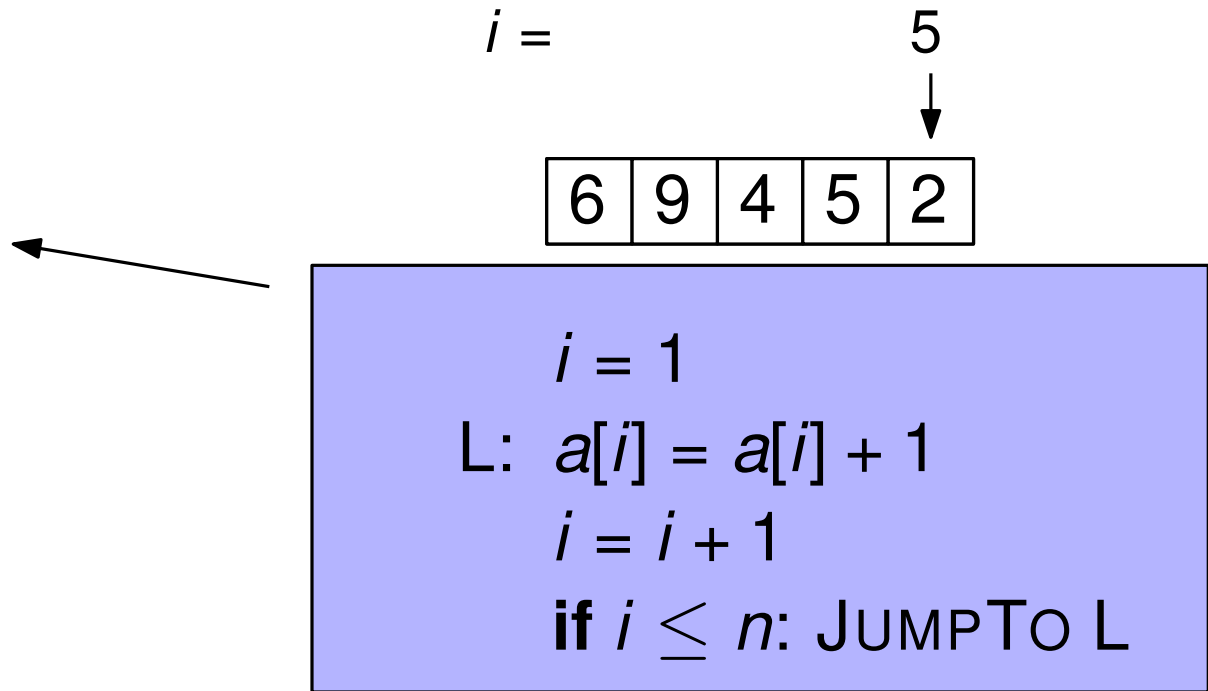
L: $a[i] = a[i] + 1$

$i = i + 1$

if $i \leq n$: JUMPTO L

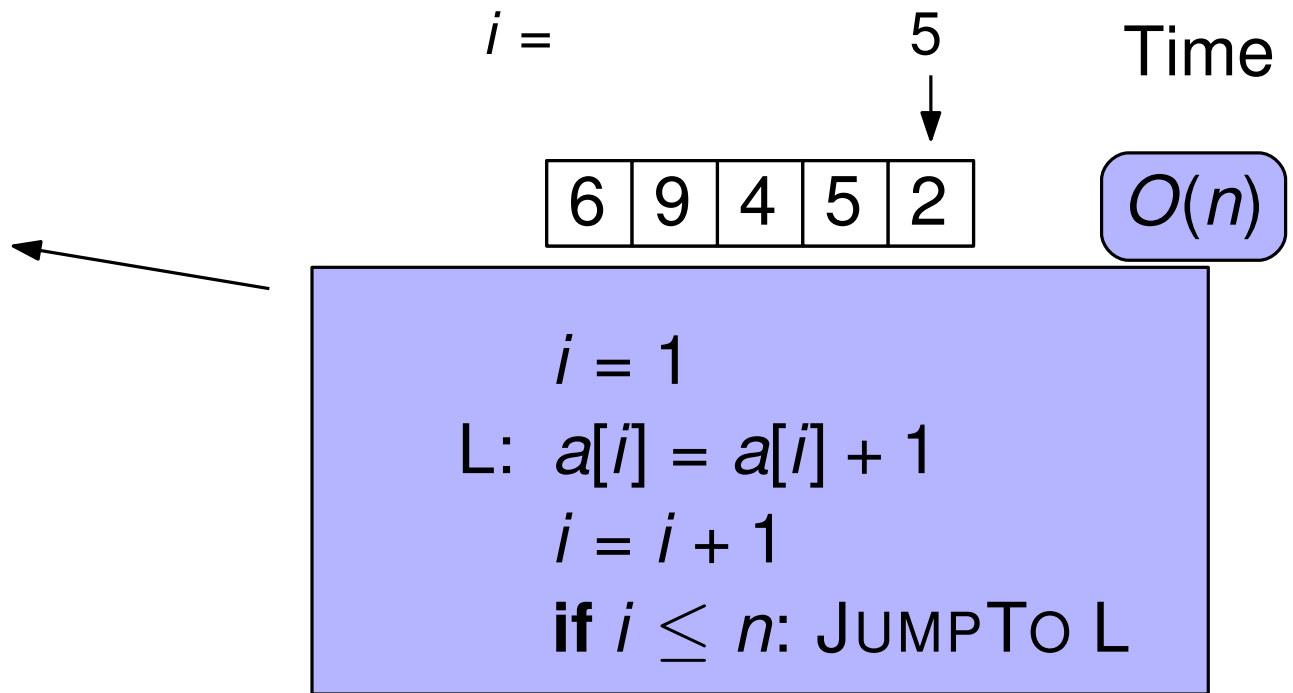
Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$



Simple example

```
for  $i = 1$  to  $n$  do  
   $a[i] = a[i] + 1$ 
```



Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

for $i = 1$ to n **in parallel do**
 $a[i] = a[i] + 1$

6	9	4	5	2
---	---	---	---	---

Time

$O(n)$

Simple example

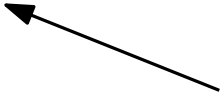
for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

6	9	4	5	2
---	---	---	---	---

Time

$O(n)$

for $i = 1$ to n **in parallel do**
 $a[i] = a[i] + 1$



Start n threads t_1, t_2, \dots, t_n
Each thread t_i (where $i = 1, 2, \dots, n$) **do**:
 $a[i] = a[i] + 1$

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

6	9	4	5	2
---	---	---	---	---

Time

$O(n)$

for $i = 1$ to n **in parallel do**
 $a[i] = a[i] + 1$

5	8	3	4	1
---	---	---	---	---

Start n threads t_1, t_2, \dots, t_n
Each thread t_i (where $i = 1, 2, \dots, n$) **do**:
 $a[i] = a[i] + 1$

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

for $i = 1$ to n **in parallel do**
 $a[i] = a[i] + 1$

Time

$O(n)$

6	9	4	5	2
---	---	---	---	---

$i = 1$	2	3	4	5
↓	↓	↓	↓	↓
5	8	3	4	1

Start n threads t_1, t_2, \dots, t_n
Each thread t_i (where $i = 1, 2, \dots, n$) **do**:
 $a[i] = a[i] + 1$

Simple example

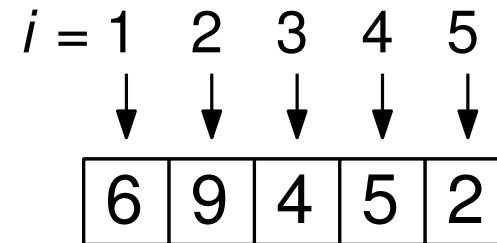
for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

for $i = 1$ to n **in parallel do**
 $a[i] = a[i] + 1$

Time

$O(n)$

6	9	4	5	2
---	---	---	---	---



Start n threads t_1, t_2, \dots, t_n
Each thread t_i (where $i = 1, 2, \dots, n$) **do**:
 $a[i] = a[i] + 1$

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

for $i = 1$ to n **in parallel do**
 $a[i] = a[i] + 1$

6	9	4	5	2
---	---	---	---	---

Time

$O(n)$

$i = 1$	2	3	4	5
↓	↓	↓	↓	↓
6	9	4	5	2

$O(1)$

Start n threads t_1, t_2, \dots, t_n
Each thread t_i (where $i = 1, 2, \dots, n$) **do**:
 $a[i] = a[i] + 1$

Simple example

for $i = 1$ to n **do**
 $a[i] = a[i] + 1$

for $i = 1$ to n **in parallel do**
 $a[i] = a[i] + 1$

6	9	4	5	2
---	---	---	---	---

Time

$O(n)$

$i = 1$	2	3	4	5
↓	↓	↓	↓	↓
6	9	4	5	2

$O(1)$

Start n threads t_1, t_2, \dots, t_n
Each thread t_i (where $i = 1, 2, \dots, n$) **do**:
 $a[i] = a[i] + 1$

Parallel Time = time of the slowest thread