Tableau Construction

Constructing a Tableau

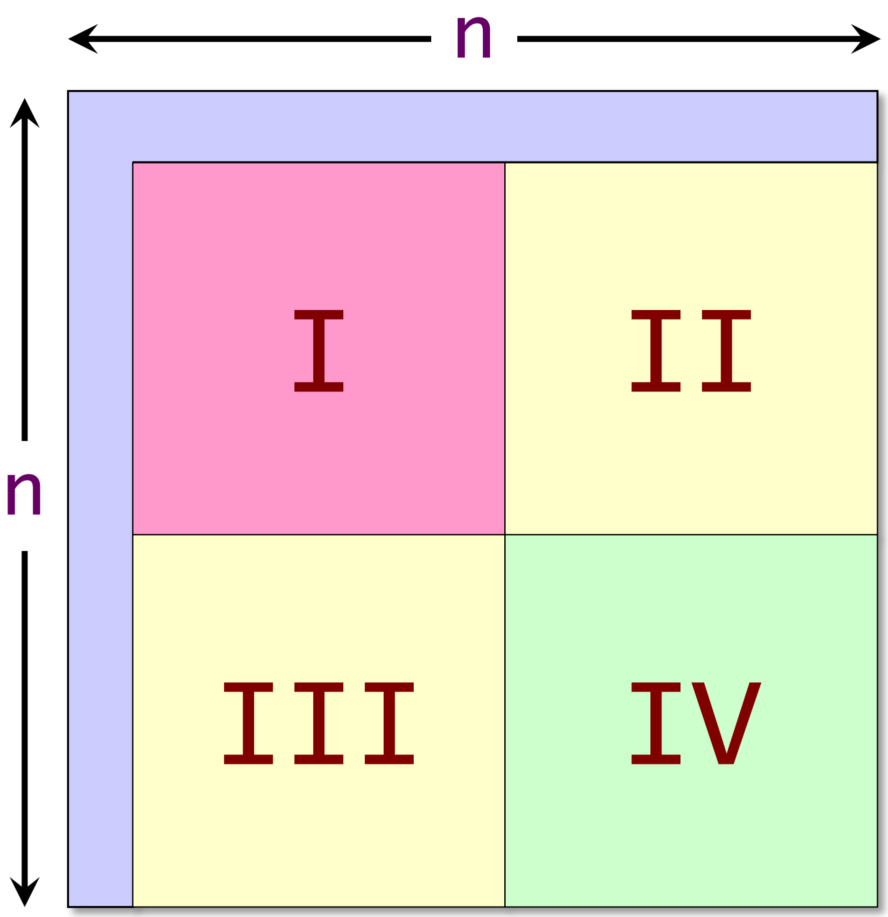
Problem: Fill in an $n \times n$ tableau A, where
$A[i][j] = f(A[i][j-1], A[i-1][j], A[i-1][j-1])$.

00	01	02	03	04	05	06	07
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37
40	41	42	43	44	45	46	47
50	51	52	53	54	55	56	57
60	61	62	63	64	65	66	67
70	71	72	73	74	75	76	77

Dynamic programming
- Longest common subsequence
- Edit distance
- Dynamic time warping

Work: $\Theta(n^2)$.

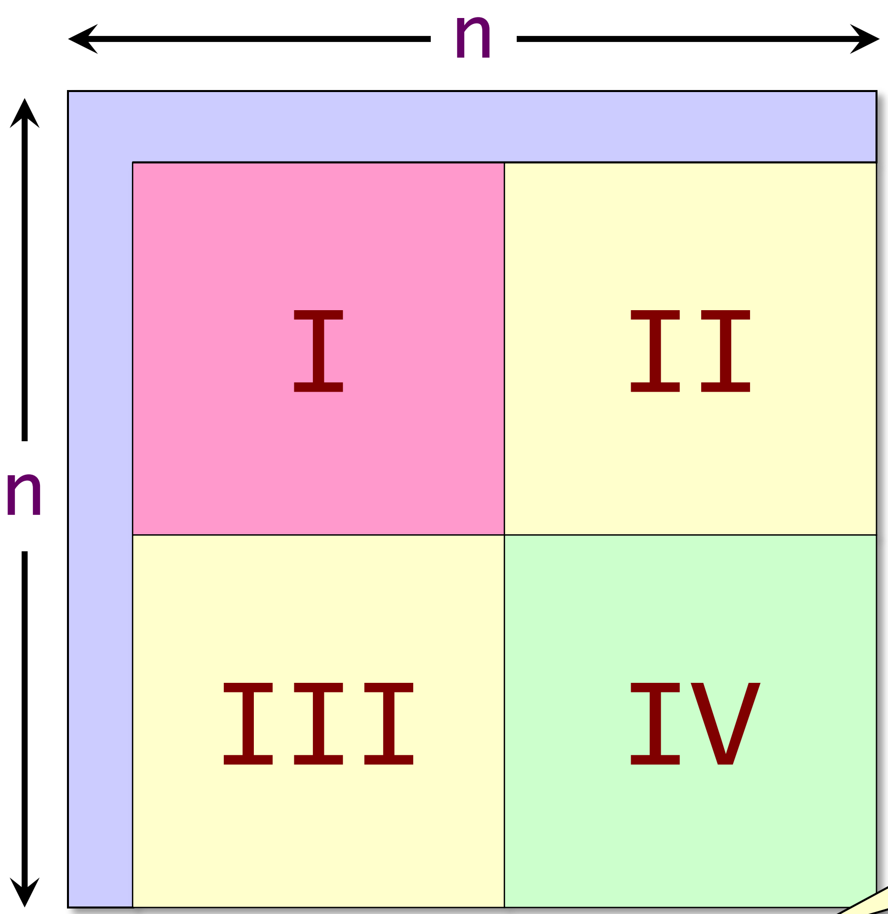
Recursive Construction



Parallel code

```
I;
cilk_spawn II;
III;
cilk_sync;
IV;
```

Recursive Construction



Parallel code

```
I;
cilk_spawn II;
III;
cilk_sync;
IV;
```

CASE 1

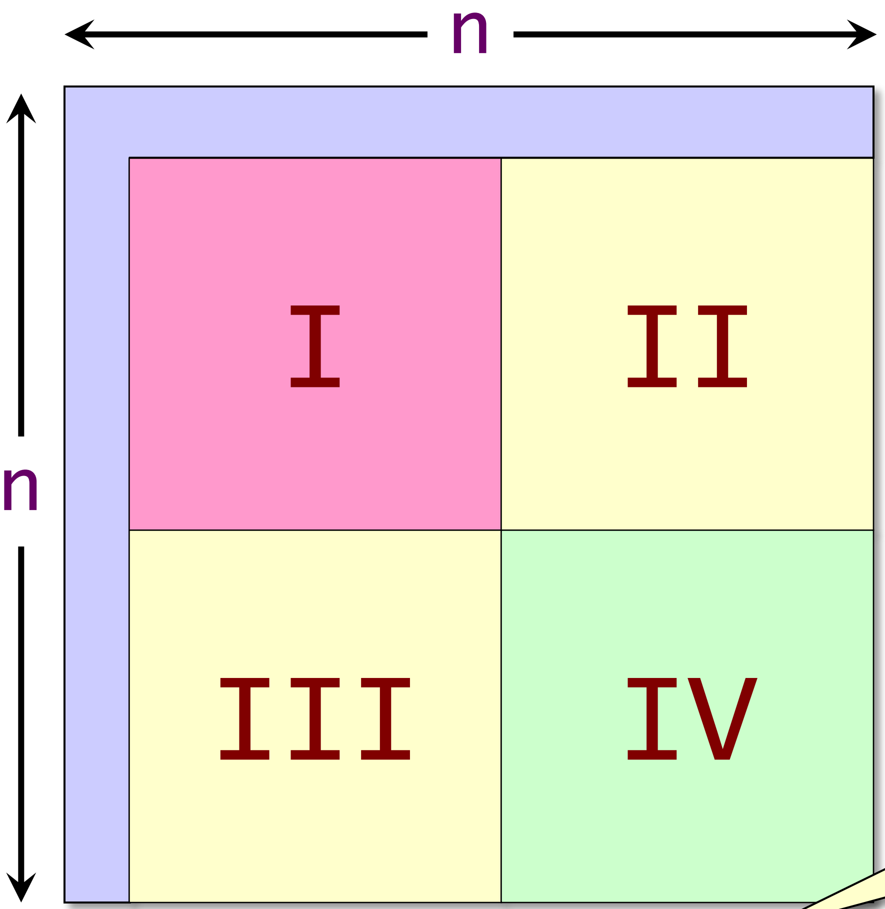
$n^{\log_b a} = n^{\log_2 4} = n^2$

$f(n) = \Theta(1)$

Work: $T_1(n) = 4T_1(n/2) + \Theta(1)$

$= \Theta(n^2)$

Recursive Construction



Parallel code

```
I;
cilk_spawn II;
III;
cilk_sync;
IV;
```

CASE 1

$n^{\log_b a} = n^{\log_2 3} = n^{\lg 3}$

$f(n) = \Theta(1)$

Span: $T_\infty(n) = 3T_\infty(n/2) + \Theta(1)$

$= \Theta(n^{\lg 3})$

Analysis of Tableau Constr.

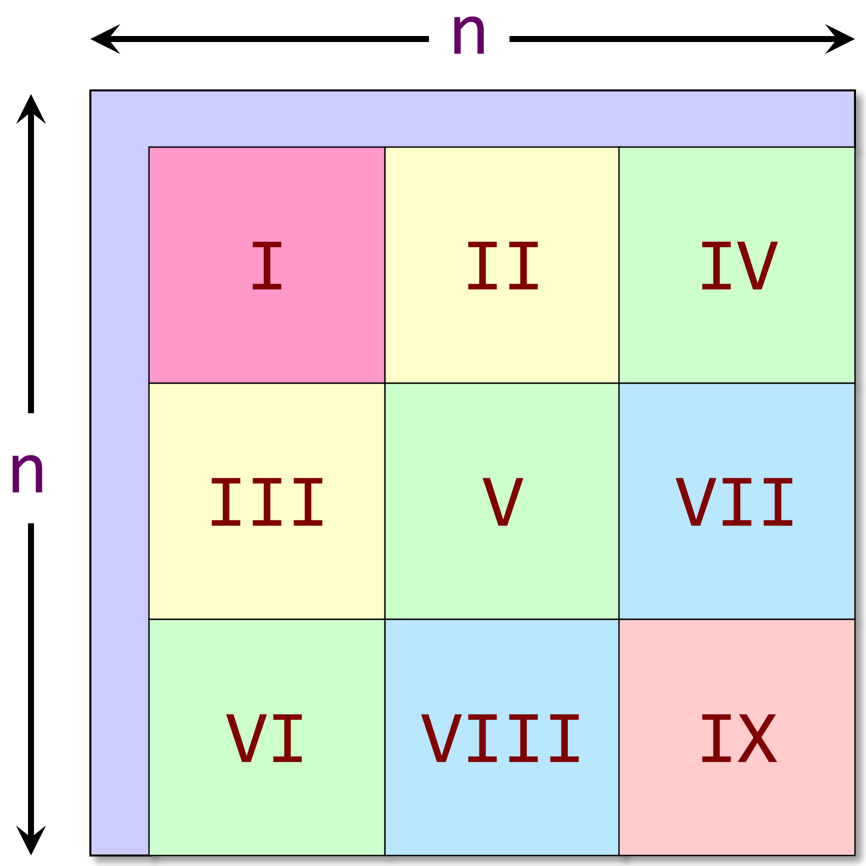
Work: $T_1(n) = \Theta(n^2)$

Span: $T_\infty(n) = \Theta(n^{\lg 3}) = O(n^{1.59})$

Parallelism: $$\frac{T_1(n)}{T_\infty(n)} = \Theta(n^{2-\lg 3})$$

$$= \Omega(n^{0.41})$$

A More-Parallel Construction




```
I;
cilk_spawn II;
III;
cilk_sync;
cilk_spawn IV;
cilk_spawn V;
VI;
cilk_sync;
cilk_spawn VII;
VIII;
cilk_sync;
IX;
```

Work: $T_1(n) = 9T_1(n/3) + \Theta(1)$

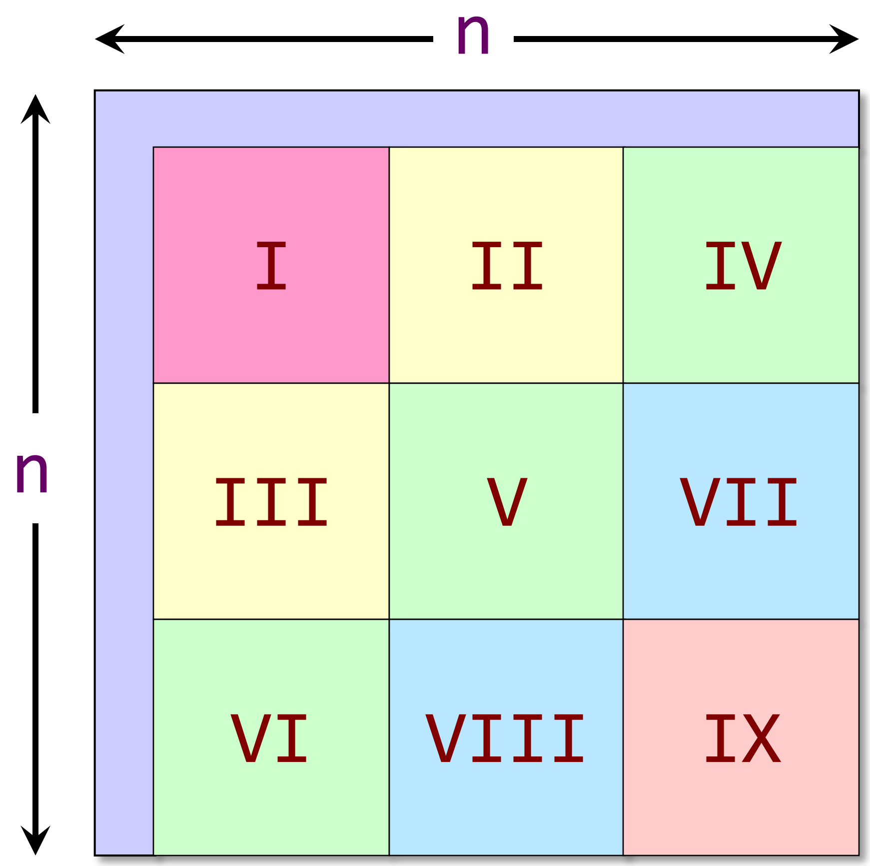
$= \Theta(n^2)$

CASE 1

$n^{\log_b a} = n^{\log_3 9} = n^2$

$f(n) = \Theta(1)$

A More-Parallel Construction




```
I;
cilk_spawn II;
III;
cilk_sync;
cilk_spawn IV;
cilk_spawn V;
VI;
cilk_sync;
cilk_spawn VII;
VIII;
cilk_sync;
IX;
```

Span: $T_\infty(n) = 5T_\infty(n/3) + \Theta(1)$

$= \Theta(n^{\log_3 5})$

CASE 1

$n^{\log_b a} = n^{\log_3 5}$

$f(n) = \Theta(1)$

Analysis of Revised Method

Work: $T_1(n) = \Theta(n^2)$

Span: $T_\infty(n) = \Theta(n^{\log_3 5}) = O(n^{1.47})$

Parallelism: $$\frac{T_1(n)}{T_\infty(n)} = \Theta(n^{2-\log_3 5}) = \Omega(n^{0.53})$$

Nine-way divide-and-conquer has about $\Theta(n^{0.12})$ more parallelism than four-way divide-and-conquer, but it exhibits less cache locality.

Puzzle

What is the largest parallelism that can be obtained for a $\Theta(n^2)$-work algorithm for the tableau-construction problem using *pure* Cilk?

- You may only use basic fork-join control constructs (`cilk_spawn`, `cilk_sync`, `cilk_for`) for synchronization.
- No using locks, atomic instructions, synchronizing through memory, etc.