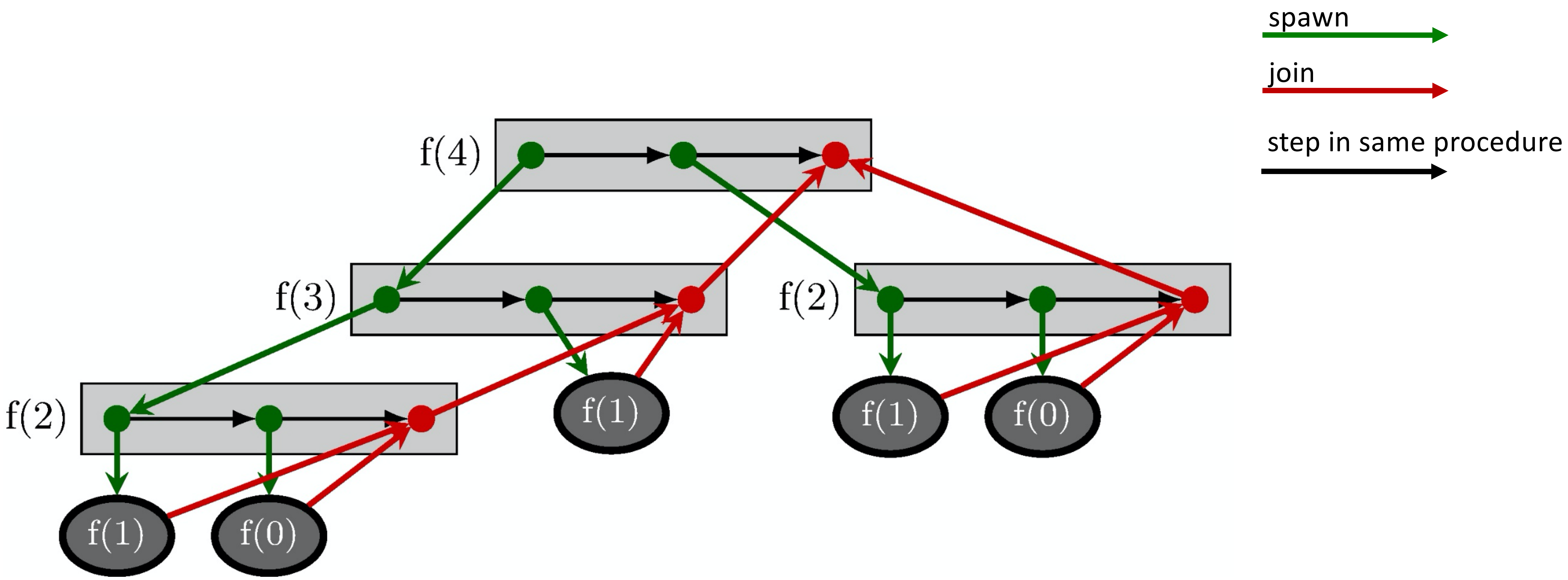
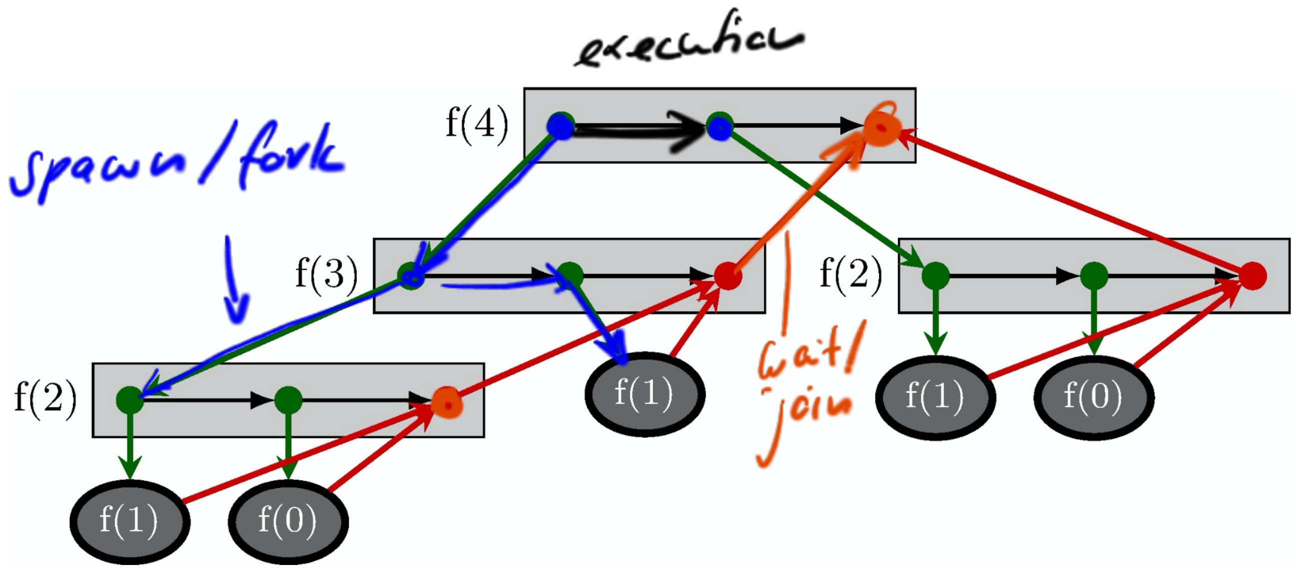
`fib(4)` task graph



The task graph is a directed acyclic graph (DAG)

fib(4) task graph

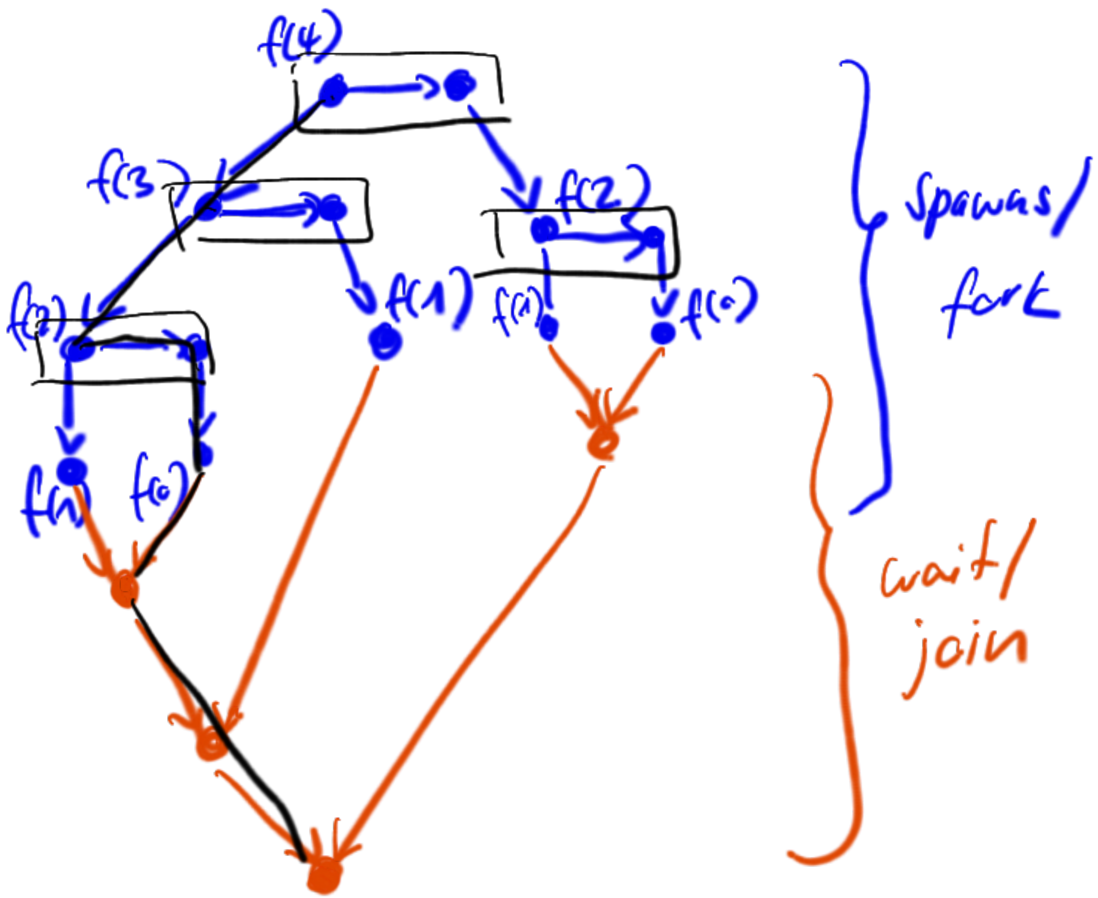



```
public class Fibonacci {
 public static long fib(int n) {
        if (n < 2)
            return n;
        spawn task for fib(n-1);
        spawn task for fib(n-2);
        wait for tasks to complete
        return addition of task results
}}}
```

spawn

join

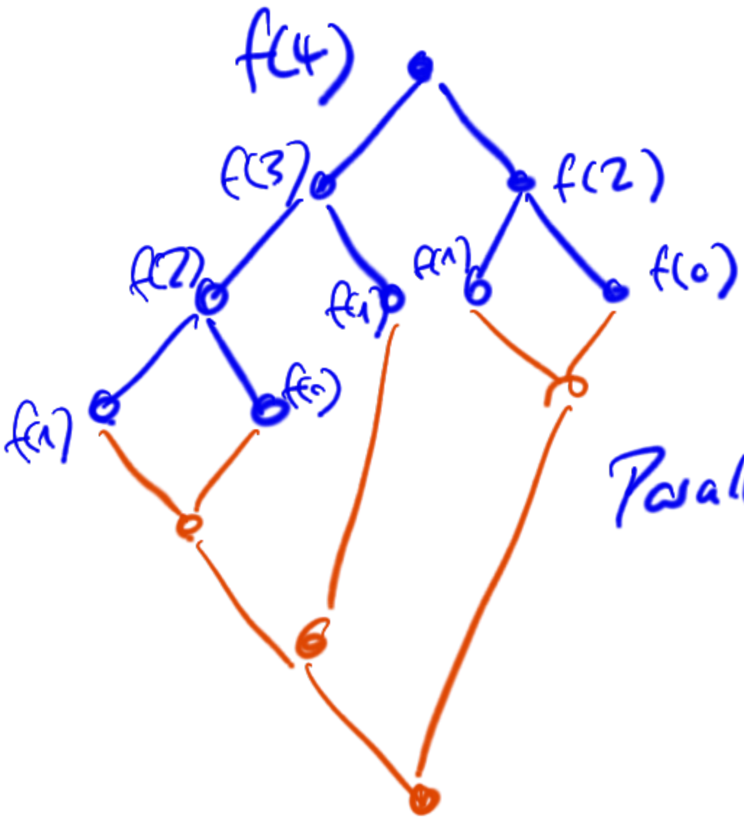
step in same procedure



$T_1 = 17$

$T_\infty = 8$

$\frac{T_1}{T_\infty} = \frac{17}{8} = 2.1$



$T_1 = 13$

$T_\infty = 7$

Parallelism: $\frac{T_1}{T_\infty} = \frac{13}{7} \approx 1.8$

$Fib(n)$

$T_1 = O(2^n)$

$\Rightarrow O\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$ tight bound

$$T_\infty(n) = \max\left(\underline{T_\infty(n-1)},\ T_\infty(n-2)\right) + O(1)$$

$$= \underline{T_\infty(n-1)} + O(1)$$

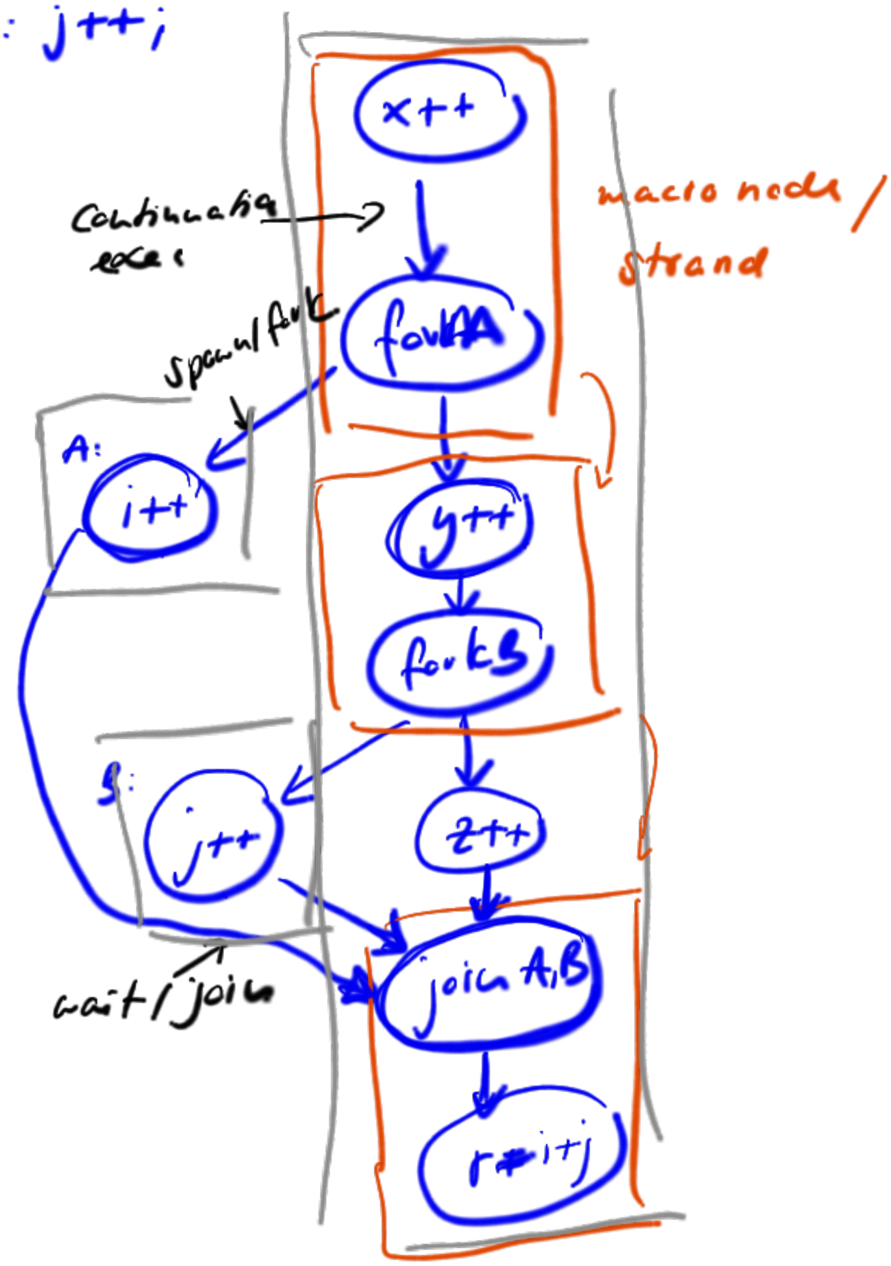
$$= O(n)$$

Parallelism $\frac{T_1}{T_\infty} = \frac{O\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)}{O(n)} = O\left(\frac{\left(\frac{1+\sqrt{5}}{2}\right)^n}{n}\right)$

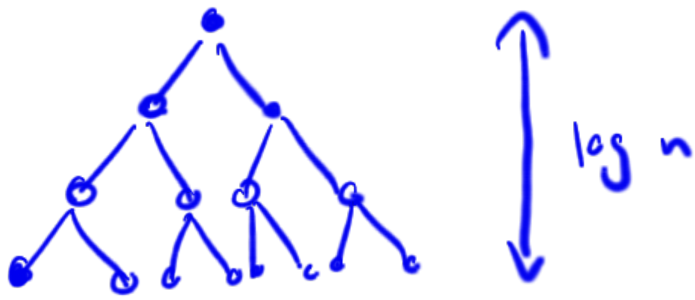
<u>Code</u>

```
x++;
fork A
y++;
fork B
z++;
join A,B
r = i+j;
```

```
A: i++;
B: j++;
```



seq: $O(n)$



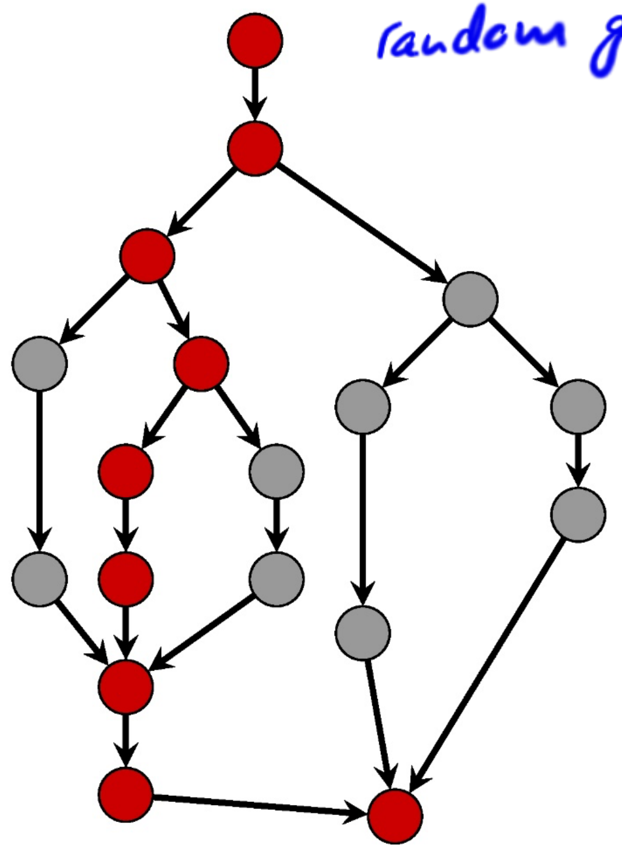
par: $O(\log n)$

Parallelism: $\frac{T_1}{T_\infty} = \frac{O(n)}{O(\log n)}$

Task parallelism: performance model (Bounds)

- **T_∞: span**, **critical path**
  - Time it takes on infinite processors
  - longest path from root to sink
-
- **T_1 / T_∞ → parallelism**
  - "wider" is better
- Lower Bounds:
  - $\mathbf{T_p \geq T_1 / P}$
  - $\mathbf{T_p \geq T_\infty}$

random graph



On this graph, T_∞ is 9