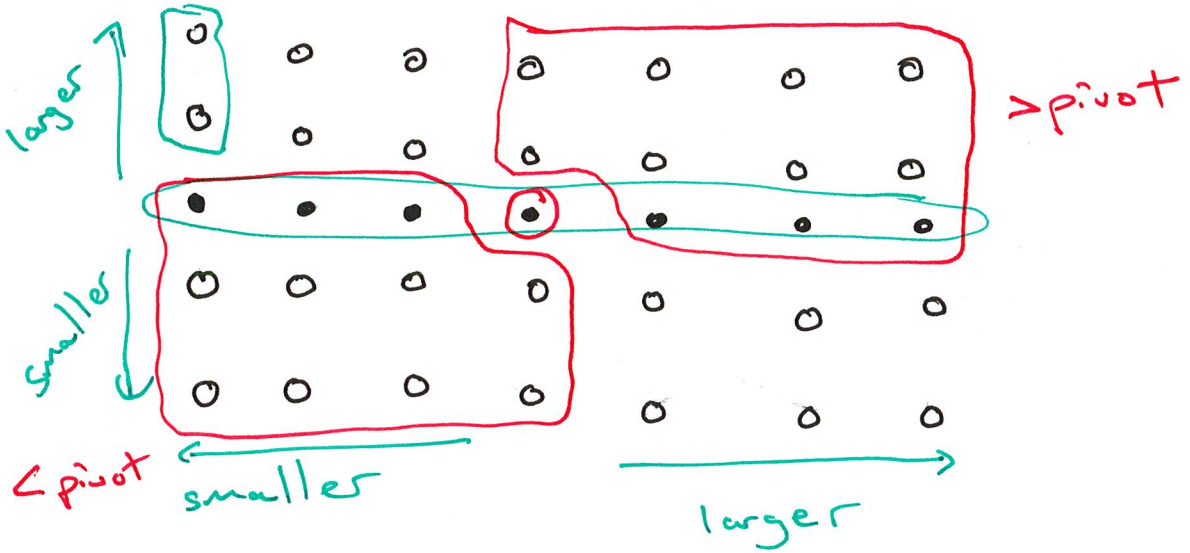
Median of Medians (Mom Select)

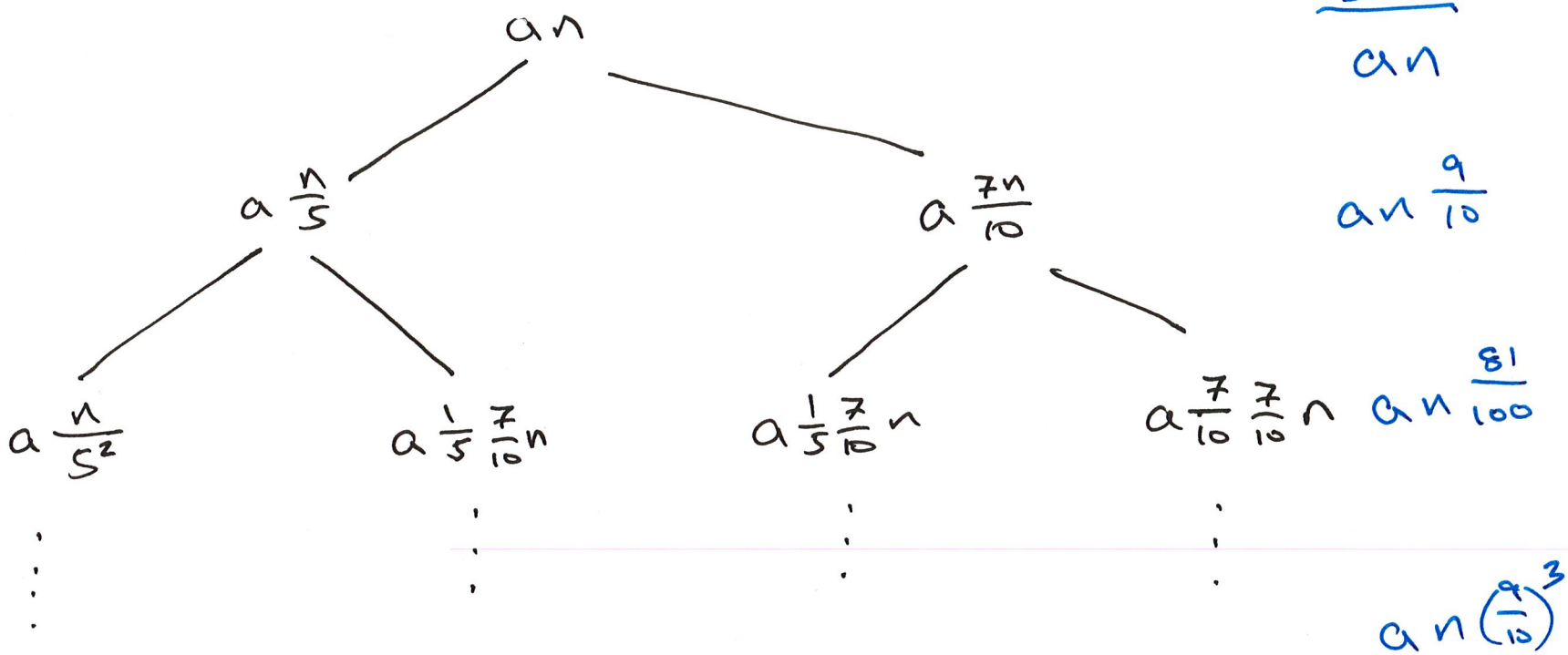
1. Divide the n elements in $\lceil \frac{n}{5} \rceil$ groups each with 5 elements $O(n)$
2. Find the median of each group of 5 by any method $O(n)$
3. Invoke MomSelect recursively on the $\lceil \frac{n}{5} \rceil$ medians to obtain the pivot
4. Swap the pivot into $A[r]$ $O(n)$ $O(1)$



$$\frac{1}{2} 3 \frac{n}{5} = \frac{3n}{10}$$

- half the columns contribute at least 3 items to the set of item larger than the pivot
- We recurse on at most $\frac{7n}{10}$ items

$$T(n) \leq an + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right)$$



$$T(n) \leq an \sum_{i=0}^{\infty} \left(\frac{9}{10}\right)^i$$

$$= an\left(\frac{1}{1-\frac{9}{10}}\right) = 10an \in O(n)$$

Dynamic Programming

- Many divide and conquer solution are slow if implemented in straight forward way
  - because they solve the same subproblem multiple times.
- dynamic programming avoids this by storing solutions to each subproblem.
- In order to apply Dynamic Programming the problem must have 2 properties.
  1. ~~Op~~ Optimal Substucture
  2. Overlapping Subproblems

Optimal Substructure

- the optimal solution can be obtained by combining optimal solutions to subproblems.

Example: Shortest Path

- consider a graph $G = (V, E)$
- the shortest path p from u to v has optimal substructure
- take any vertex w on path p
  - ~~opt~~ split p into p_1 from u to w and p_2 from w to v

- p_1 must be the shortest path from u to w
  if not we could substitute that path into p reducing its length
- since p is optimal this can't happen

Example: Longest Path

Does not have optimal substructure

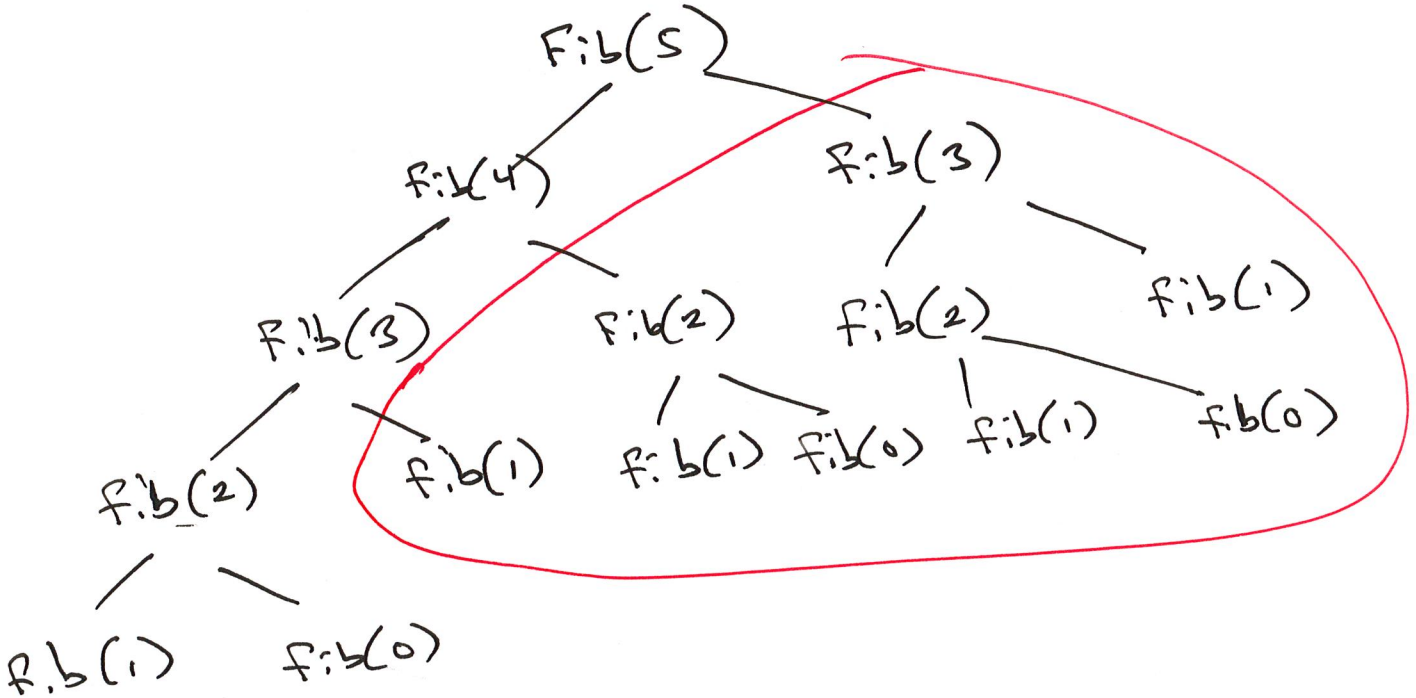
- consider a graph $G = (V, E)$
- find the longest simple path from u to v
- let p be the longest simple path

- choose intermediate vertex w on path p
- split p into p_1 from u to w and p_2 from w to v
    - p_1 and p_2 might not be longest
- take the longest path from u to w and w to v and combine them $p_1' p_2'$
- Is $p_1' p_2'$ the longest path from u to v
- the conjunction of 2 simple paths is not neccessarily cycle free.

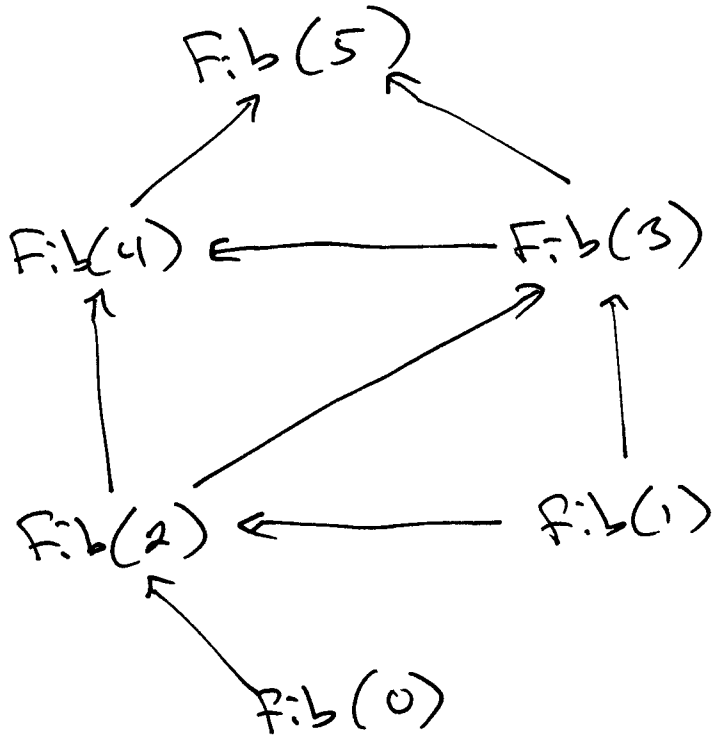
Fibonacci

```
Fib(n)
    if n ≤ 1
        return n
    return Fib(n-1) + Fib(n-2)
```

$\Theta(\phi^n)$



- We can visualize dynamic programming as a directed acyclic graph (DAG) in which each subproblem result can contribute to multiple parents



Memoization

- A top-down approach that directly falls out of the recursive algorithm. Modify the recursive algorithm to save the solution to a subproblem before returning it. Before trying to solve a subproblem check if we already have a solution.

Tabulation

- A bottom-up solution where we restructure the problem so that subproblems are solved first.