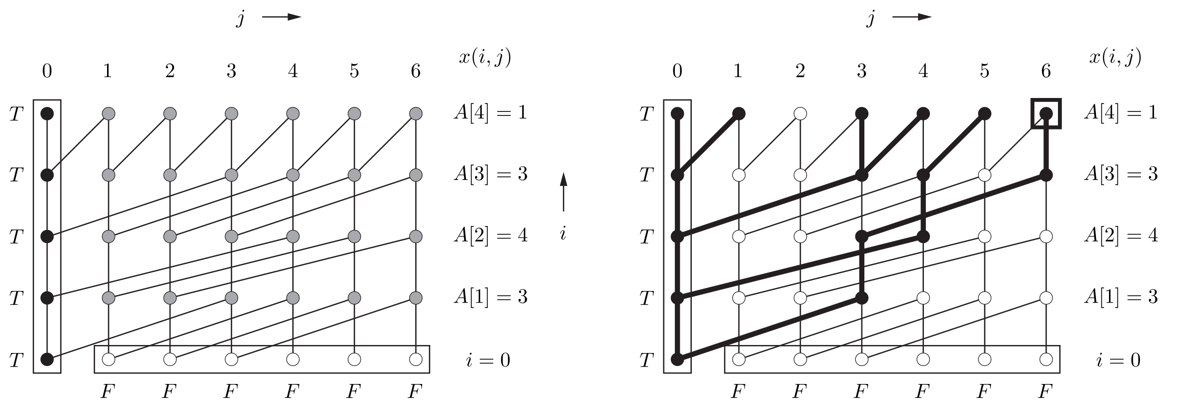
Lecture 19: Pseudo-polynomial

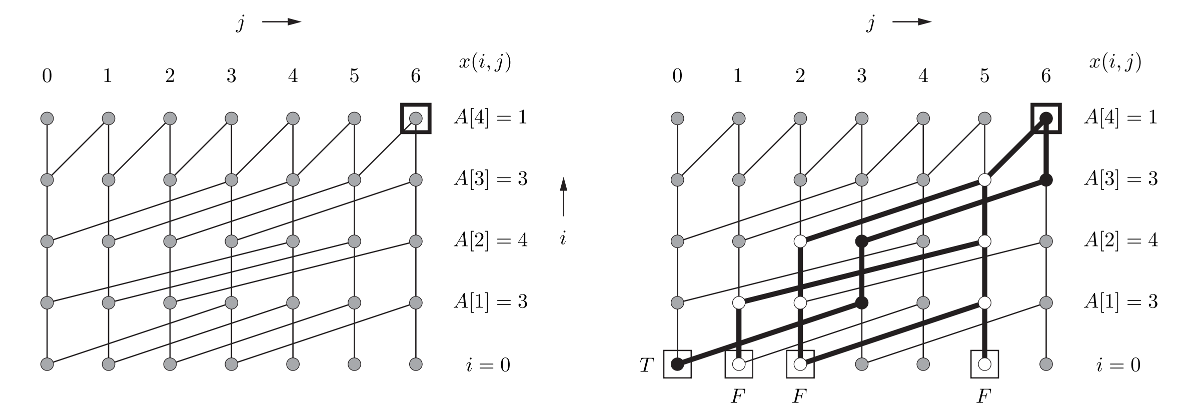
Subset Sum

- Input: Set of n positive integers $A = \{a_1, \ldots, a_n\}$
- Output: Is there a subset from A that sums exactly to S? (i.e., $\exists A' \subseteq A$ s.t. $\sum_{a \in A'} a = S$?)
- Example: $A = (1, 3, 4, 12, 19, 21, 22)$, $S = 47$ allows $A' = \{3, 4, 19, 21\}$
- Optimization problem? Decision problem! Answer is yes or no, T or F
- In example, answer is yes. However, answer is no for some S, e.g. $2, 6, 9, 10, 11, \ldots$

1. **Subproblems:**
   - $x(i, j)$: T if can make sum j using items a_1 to a_i, F otherwise
2. **Relate:**
   - Idea: Is last item a_i in a valid subset? (Guess!)
   - If yes, then try to sum to $j - a_i \geq 0$ using remaining items
   - If no, then try to sum to j using remaining items
   - $x(i, j) = \text{OR} \begin{cases} x(i-1, j - A[i]) & \text{if } j \geq A[i] \\ x(i-1, j) & \text{always} \end{cases}$
   - $i \in \{1, \ldots, n\}, j \in \{1, \ldots, S\}$
   - Subproblems $x(i, j)$ only depend on strictly smaller i, so acyclic
   - Solve in order of increasing i, then increasing (or arbitrary) j
3. **Base Case:**
   - $x(i, 0) = T$ for $i \in \{0, \ldots, n\}$ (space packed exactly!)
   - $x(0, j) = F$ for $j \in \{1, \ldots, S\}$ (no more items available to pack)
   - Solve subproblems via recursive top down or iterative bottom up
4. **Solution:**
   - Maximum evaluated expression is given by $x(n, S)$
   - Example: $A = (3, 4, 3, 1), S = 6$ solution: $A' = \{3, 3\}$
   - Bottom up: Solve all subproblems (Example has 35)



- Top down: Solve only **reachable** subproblems (Example, only 14!)



5. **Time:**

- # subproblems: $O(nS)$, $O(1)$ work per subproblem, $O(nS)$ time

Is this polynomial?

- Assume that each $a_i \leq S$ (otherwise they could not be used in any sum)
- What is size of input? If need S space to store number S then size of input is $O(nS)$
- If numbers written in binary, then each a_i is storable in $\lceil \log S \rceil$ bits
- Input has size $O(n \log S)$, which is **exponentially smaller** than $O(nS)$
- If numbers polynomially bounded, $S = O(n^c)$ for fixed $c > 0$, $O(nS)$ also polynomial
- This is called a **pseudo-polynomial**-time algorithm
- Is Subset Sum solvable in polynomial time when numbers are not polynomially bounded?
- No if **P** $\neq$ **NP**. What does that mean? Next Lecture!