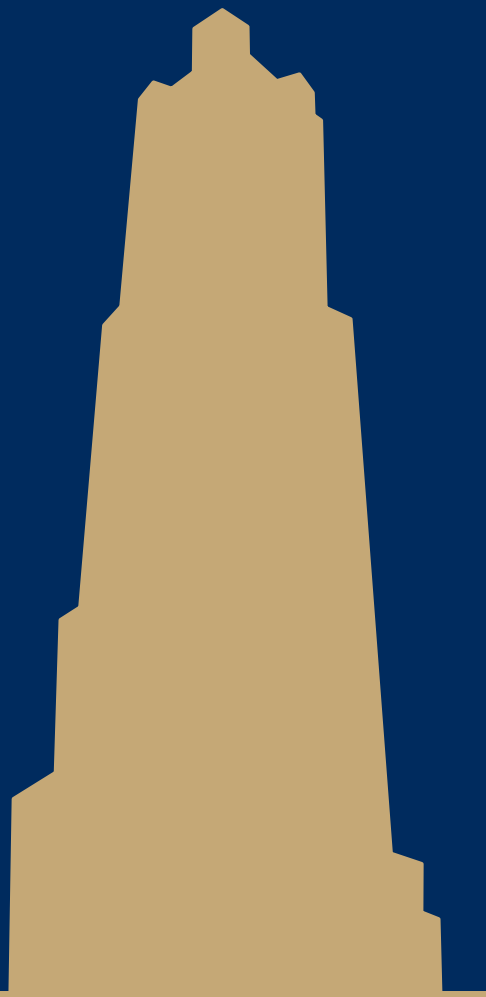
CS/COE 1501

www.cs.pitt.edu/~lipschultz/cs1501/

Additional Dynamic Programming Examples

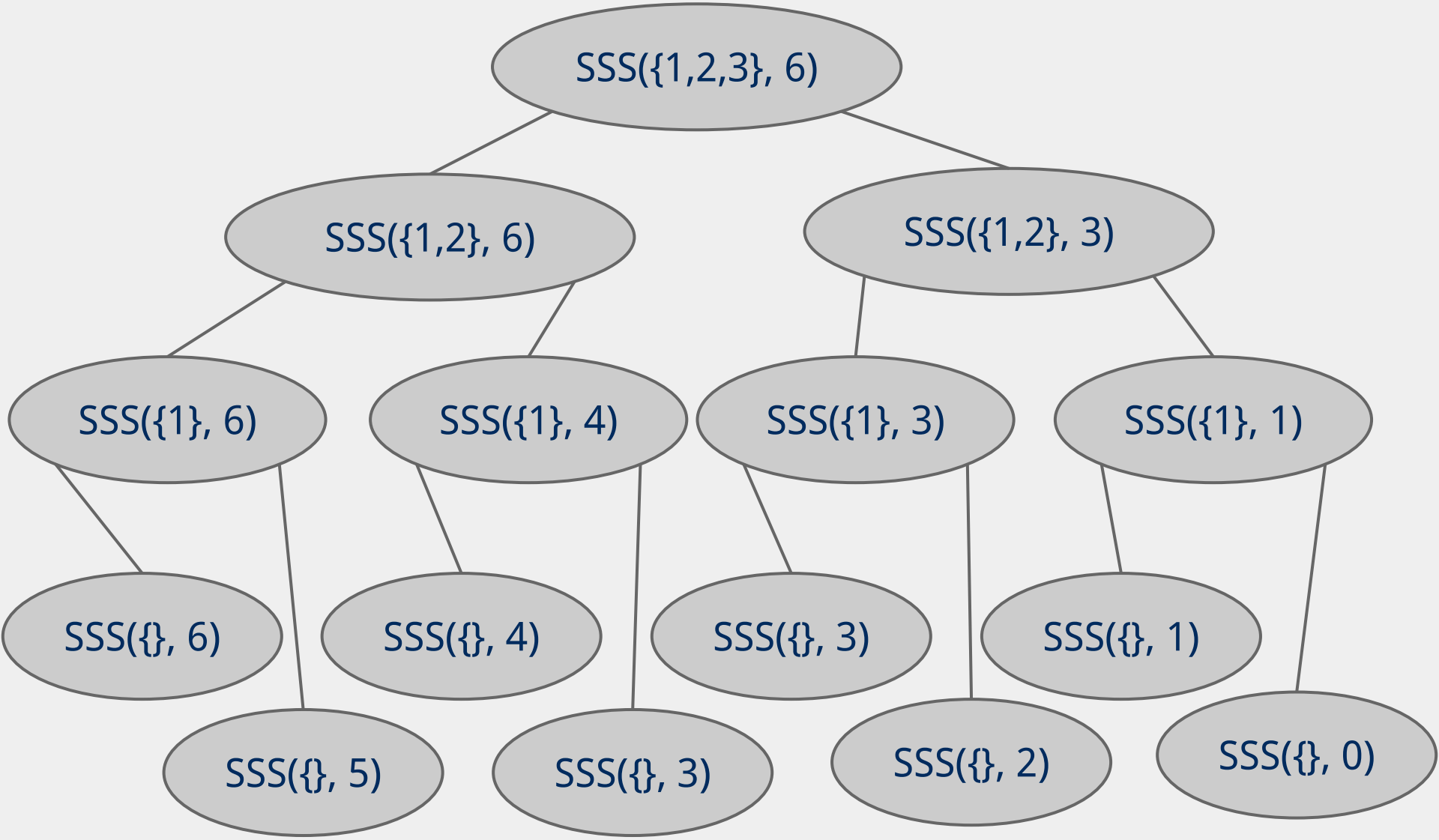


Subset sum

- Given a set of non-negative integers S and a value k, is there a subset of S that sums to exactly k?

```
boolean SSS(int set[], int sum, int n) {
    if (sum == 0)
        return true;
    if (sum != 0 && n == 0)
        return false;
    if (set[n-1] > sum)
        return SSS(set, sum, n-1);
    return SSS(set, sum, n-1)
            || SSS(set, sum-set[n-1], n-1);
}
```

Subset sum calls



Reviewing our naive solution

- What would a memoization data structure look like?

```
boolean SSS(int set[], int sum, int n) {
    if (sum == 0)
        return true;
    if (sum != 0 && n == 0)
        return false;
    if (set[n-1] > sum)
        return SSS(set, sum, n-1);
    return SSS(set, sum, n-1)
            || SSS(set, sum-set[n-1], n-1);
}
```

Subset sum bottom-up dynamic programming

```
boolean SSS(int set[], int sum, int n) {
    boolean[][] subset = new boolean[sum+1][n+1];
    for (int i = 0; i <= n; i++) subset[0][i] = true;
    for (int i = 1; i <= sum; i++) subset[i][0] = false;
    for (int i = 1; i <= sum; i++) {
        for (int j = 1; j <= n; j++) {
            subset[i][j] = subset[i][j-1];
            if (i >= set[j-1])
                subset[i][j] = subset[i][j]
                               || subset[i - set[j-1]][j-1];
        }
    }
    return subset[sum][n];
}
```

Longest Common Subsequence

- Given two sequences, return the longest common subsequence
  - A **Q** S R **J** K **V** B **I**

    Q B W F **J** **V** **I** T U
- We'll consider a relaxation of the problem and only look for the *length* of the longest common subsequence

LCS dynamic programming example

x = A Q S R J B I

y = Q B I J T U T

i\j	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0							
2	0							
3	0							
4	0							
5	0							
6	0							
7	0							

LCS dynamic programming solution

```
int LCSLength(String x, String y) {
    int[][] m = new int[x.length + 1][y.length + 1];
     for (int i=0; i <= x.length; i++) {
         for (int j=0; j <= y.length; j++) {
              if (i == 0 || j == 0) m[i][j] = 0;
              if (x.charAt(i) == y.charAt(j))
                 m[i][j] = m[i-1][j-1] + 1;
              else
                 m[i][j] = max(m[i][j-1], m[i-1][j]);
    return m[x.length][y.length];
}
```

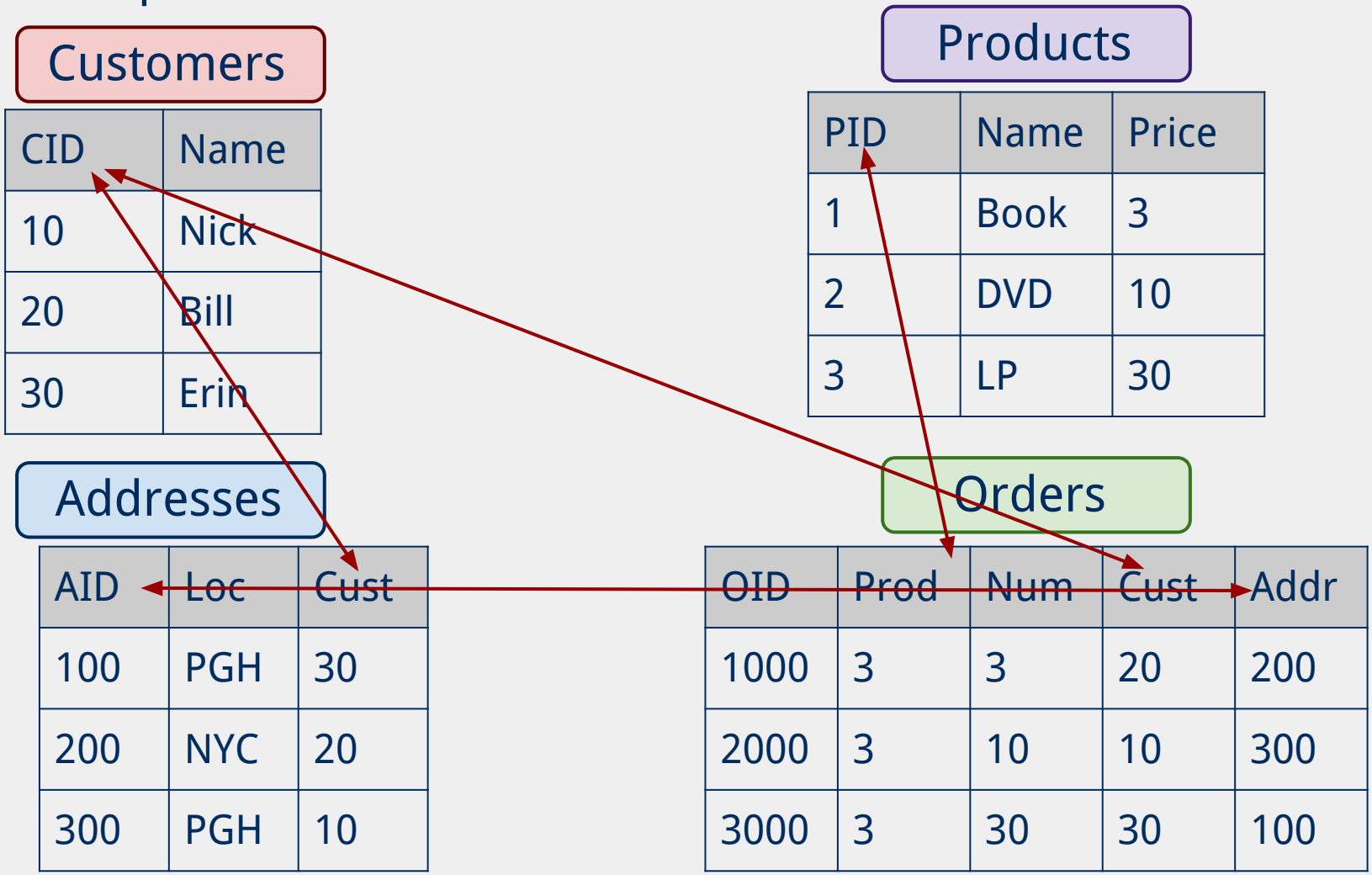
How might this be modified so it can return the longest common sub-sequence, not its length?

Database query optimization

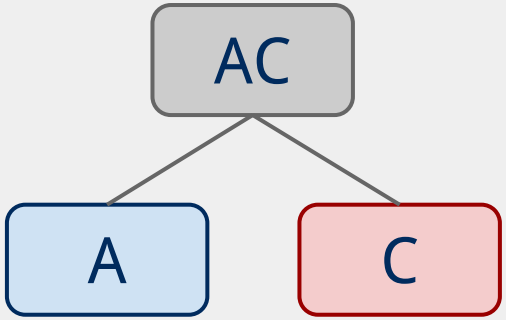
- SQL is a very popular database query language
  - It is *declarative*
    - User's don't specify what the database should do to answer their query, just what they want
    - The *query optimizer* then finds the best plan to find the result of the user's query
      - Dynamic programming is a popular approach to query optimization

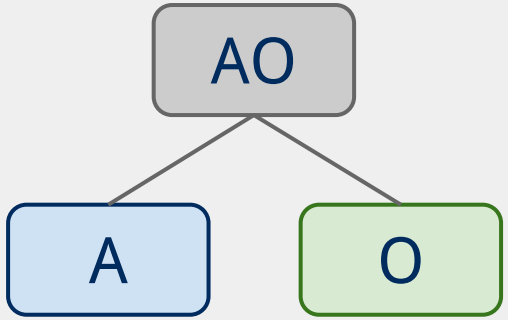
Database query optimization example

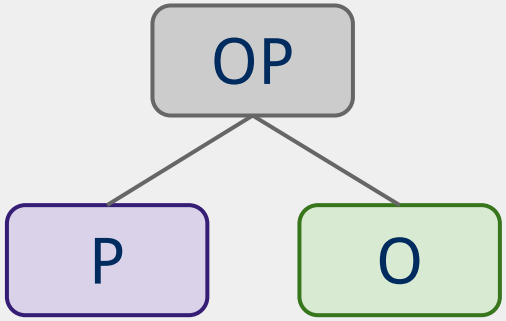
- Example database:

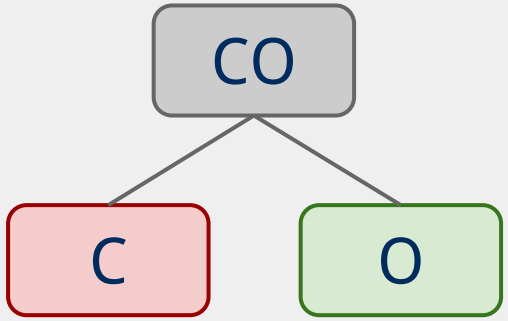


First, find plans to join pairs of tables



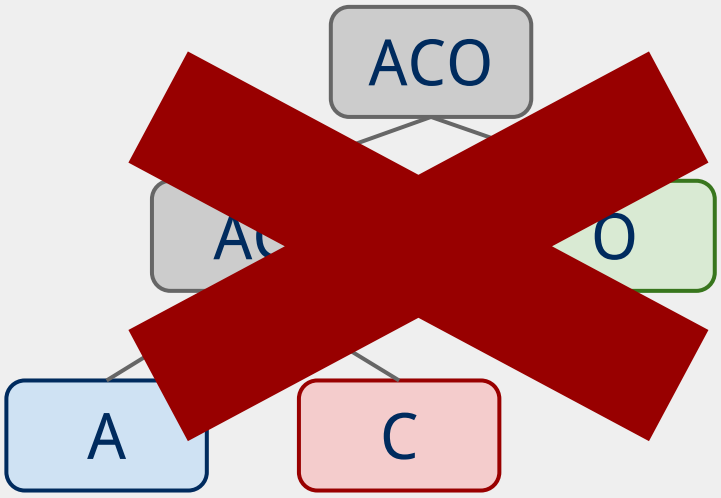


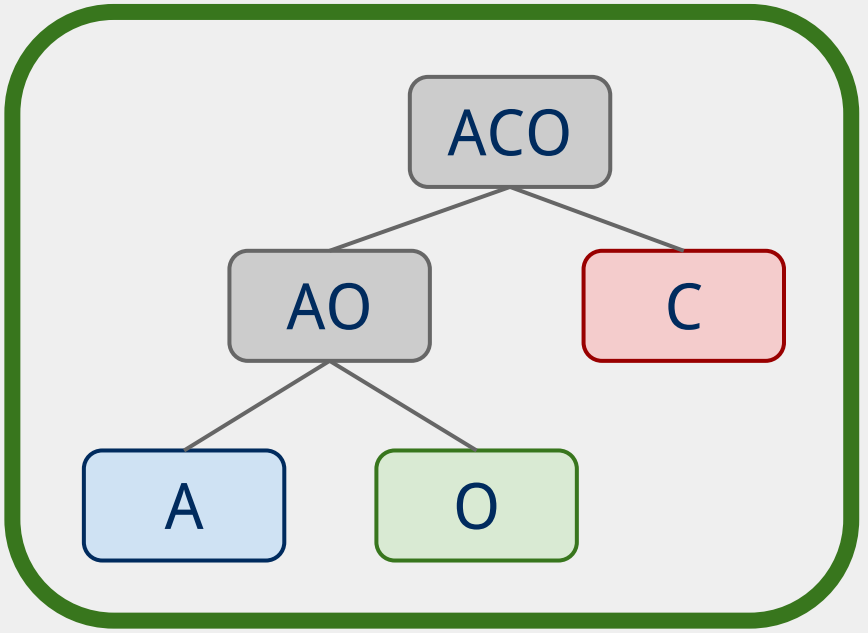


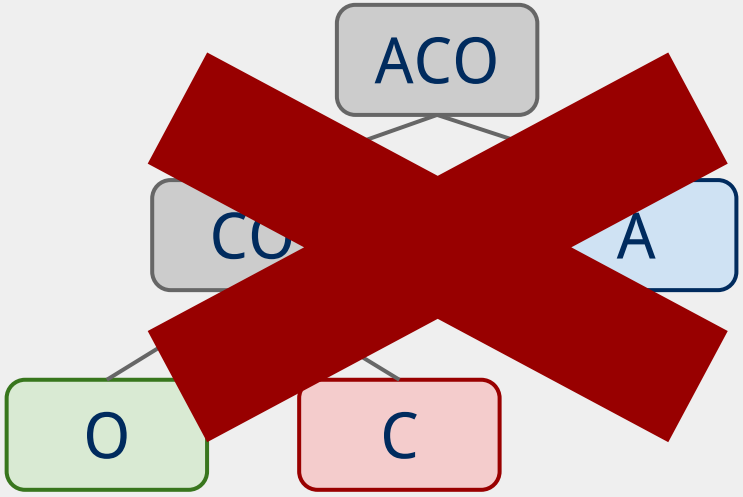


Next, find best plans to join triplets of tables

- ACO, for example:

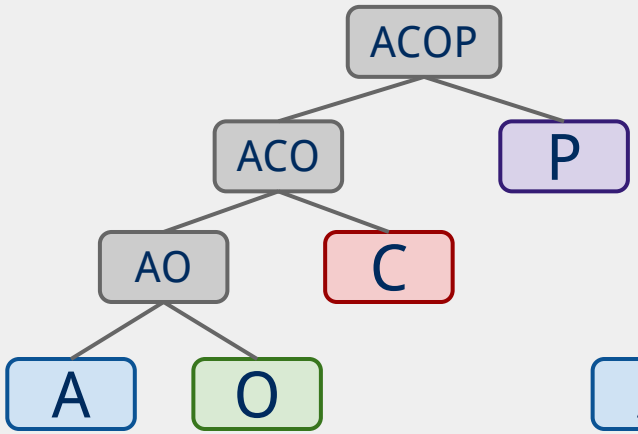


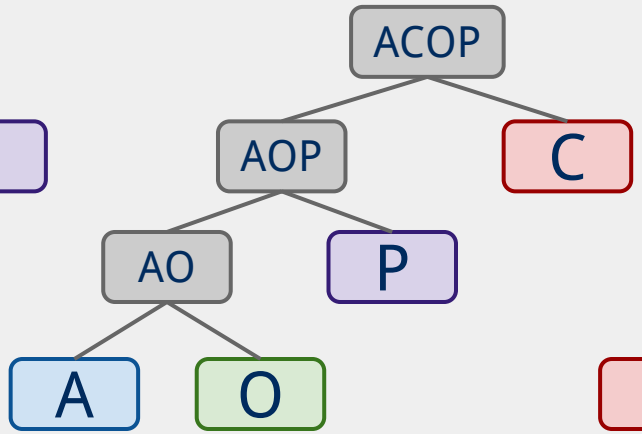


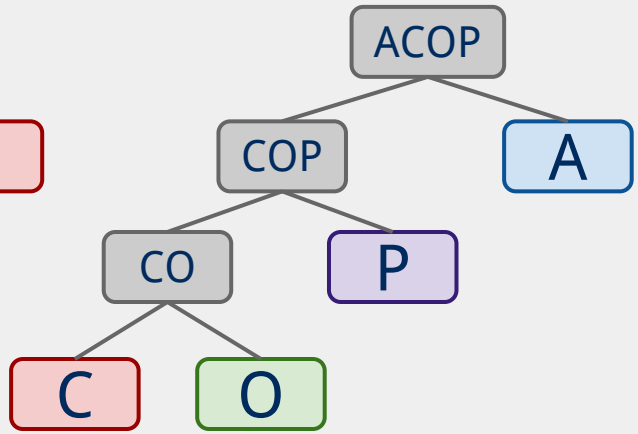


Query optimizer uses properties of the tables and data stored in them to determine ...

Continue until you find all n-way plans







Already know this is the most efficient ACO plan, don't need to try any others!