# Complete Search

Problem Solving and Programming Techniques 2014: Lectured by Dr. Min-Zheng Shieh @ I$^2$CRC, NCTU.

# Nickname: Brute Force

- Complete search is the most basic method to solve (optimization) problems.
  - Try all possible cases/solutions, then determine whether the best one exists.
- In general, its implementation is easy.
- It often results in Time Limit Exceeded.
  - You might need some slightly elegant technique.

# UVA 725

- Write a program that finds and displays all pairs of 5-digit numbers that between them use the digits 0 through 9 once each, such that the first number divided by the second is equal to an integer N, where $2 \le N \le 79$. That is,

$$abcde/fghij=N$$

where each letter represents a different digit. The first digit of one of the numerals is allowed to be zero.

# True Brute-Force

- Try 0 to 9 for all ten symbols.
  - $10^{10}$ cases. With high probability, you will get TLE.
- Try all 10! permutations
  - 10!=3268800. It seems better, but you probably will get TLE.

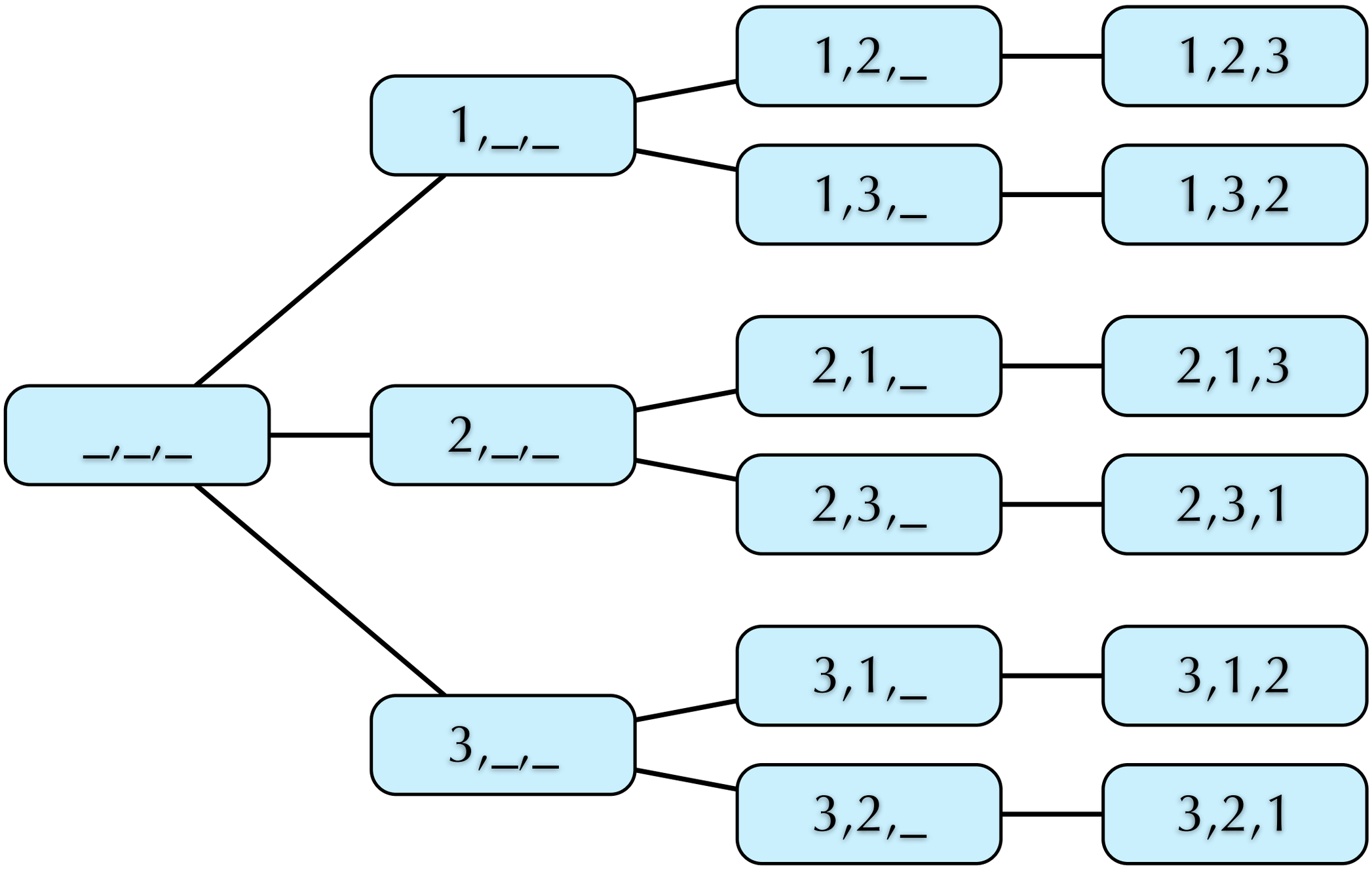
# Slightly elegant method

- “abcde/fghij=N” iff “abcde=N×fghij”
- Let try all possible fghij’s, then test if N×fghij is a valid 5-digit integer.
  - $\leq 10^5$ cases
- Is there any better method?

# UVA 11742

- Straightforward idea: try all permutations
- At most 8!=40320 cases to verify
- The main problem:
- How to enumerate all permutations?

# Permutation generation



# UVA 12455

- Simply verify all possible subsets
- We can do this recursively
- How to do it iteratively?

# UVA 750

- In chess it is possible to place eight queens on the board so that no one queen can be taken by any other. Write a program that will determine all such possible arrangements for eight queens given the initial position of one of the queens.

# True Brute-Force

- 64 positions are on the check board.
- The solution spaces: #ways of choosing 7 out of 63 = 553270671.
- It is too brute! You will get a TLE!

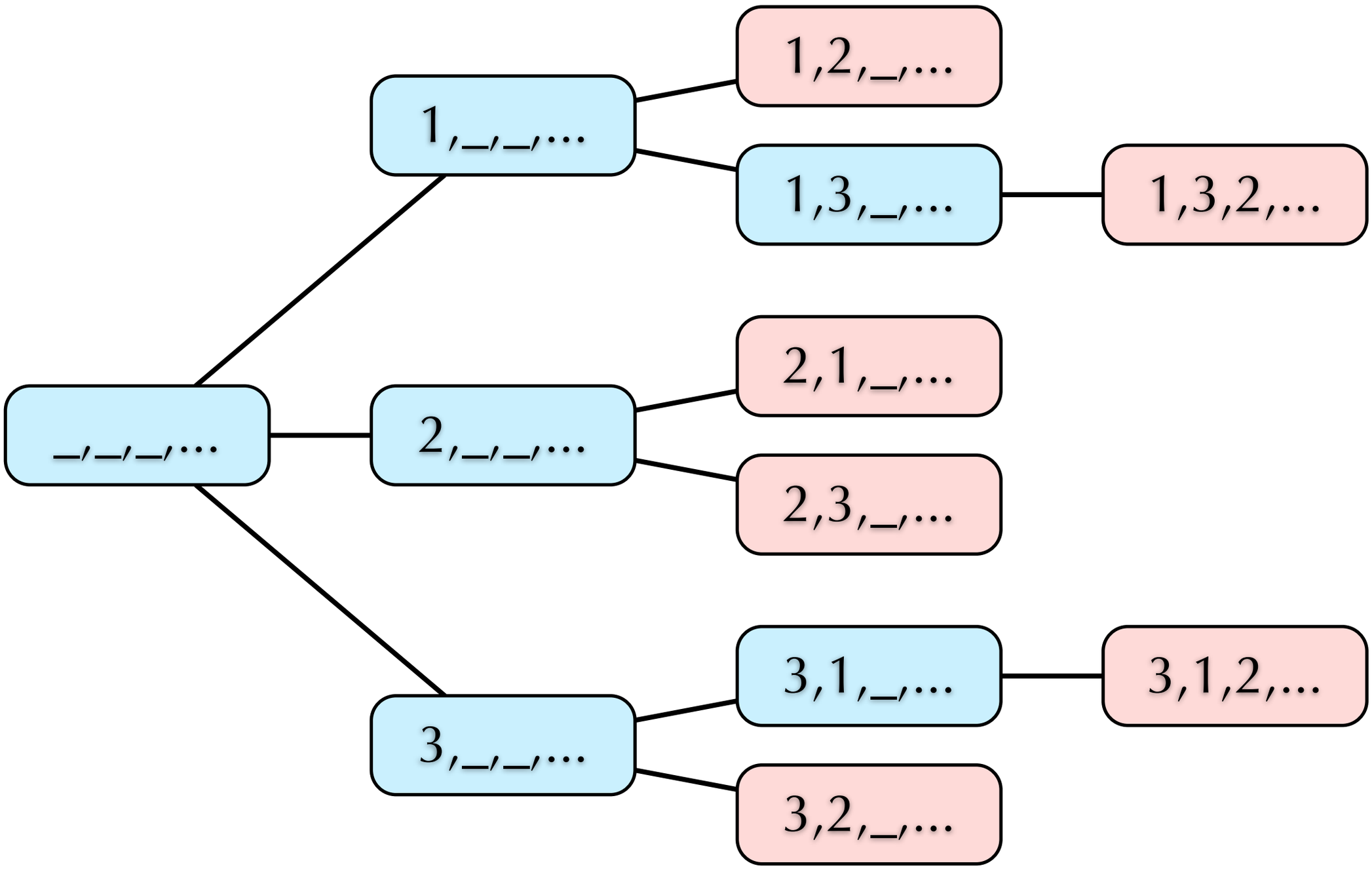
# Slightly elegant method

- No two queens can be placed on a row simultaneously!
- Just try 7!=5040 possible cases
- Fast enough?

# Backtracking

- During the enumeration process, we can eliminate many impossible cases before we recurse.
- It can reduce the running time.
  - The performance is decided by the elimination process.
- It is hard to analyze the true time complexity! (Risky method in contest!)

# Permutation generation



# Implementation

- Recursive calls
- Stack
  - Might be faster
  - Need to maintain more parameters
- Iterative approach
  - By next_permutation
  - Hard to implement backtracking