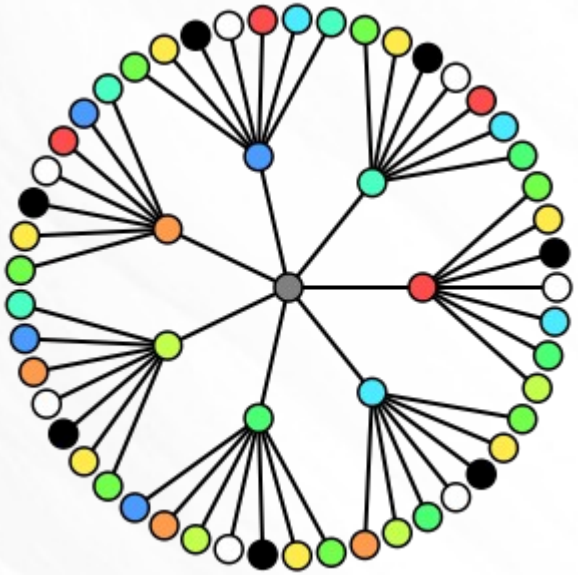
Brute force

- “Obvious” solution
- Enumerate all possibilities
- Reliable way to score 30-50%

Depth-first search

- Recursively enumerate all possibilities
- Takes n^m time, where n is the depth and
  m is the branching factor
- Backtracking can speed this up



permutations/combinations

- Permutations: possible orderings of the elements of a set
- For each element, generate the permutations for the remaining elements and prepend that element
- Combinations: ways of selecting subsets of size K
- For each element from 1 to N-K-1, recursively generate combinations of size K-1 for elements after that one

{1, 2, 3}

{1, 3, 2}

{2, 1, 3}

{2, 3, 1}

{3, 1, 2}

{3, 2, 1}

{1, 2, 3}

{1, 2}
{1, 3}
{2, 3}

Generating all subsets

Solution 1:

- Use bits of integer as flags for membership
- Increment
- x & (1<<n) is non-zero if (n+1)th bit is set

Solution 2:

- Depth-first search

000

001

010

011

100

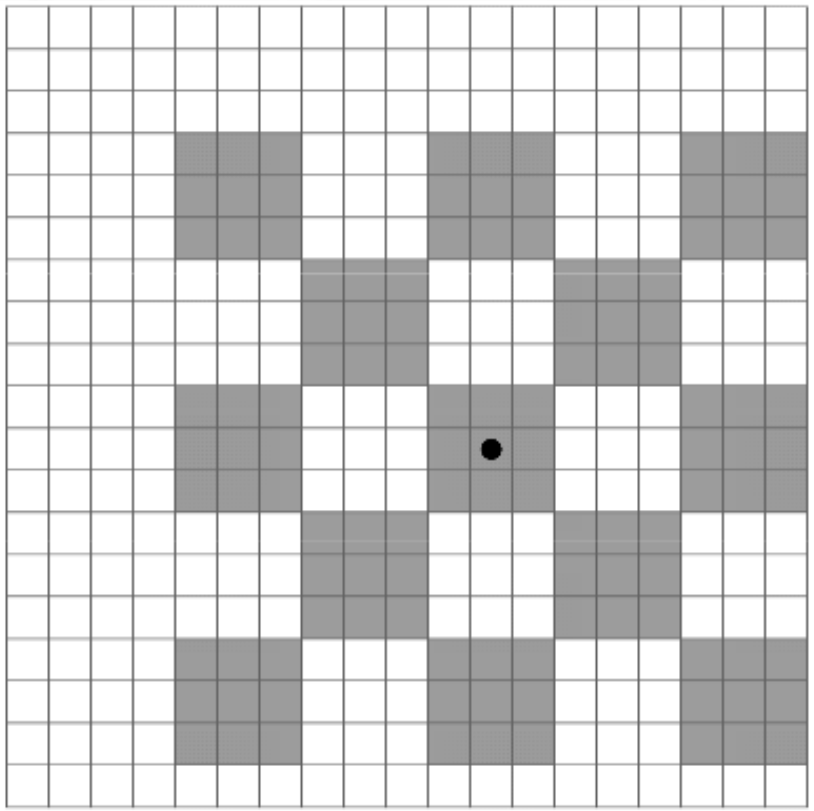
101

110

111

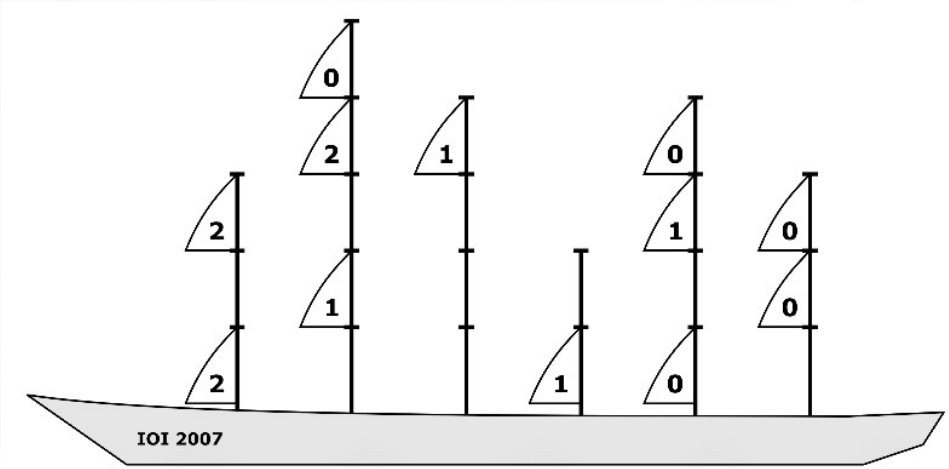
Problem example 1 - aliens

- Brute force: O(n)
- Brute force marks: 40
- Linear search



Problem example 2 - sails

- “In test cases worth a total of 25 points, the total number of ways to arrange the sails will be at most 1000000”
- Generate all combinations for each mast
- Other suboptimal solutions may be easier



Problem example 3 - flood

- Walls flooded on one side only collapse in 1 hour
- Determine which walls will not collapse
- Brute force: O(W.H)
- Brute force marks: 40
- 0-1 BFS