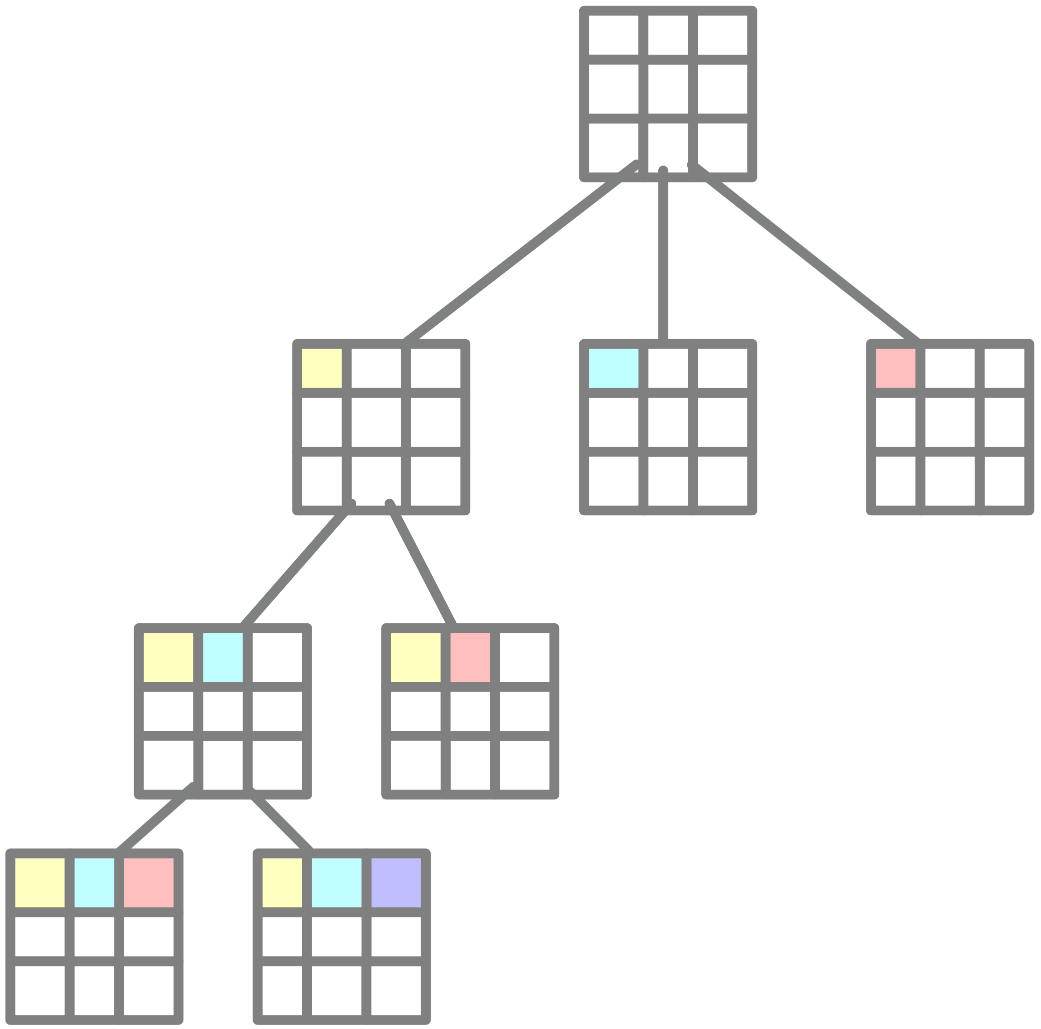
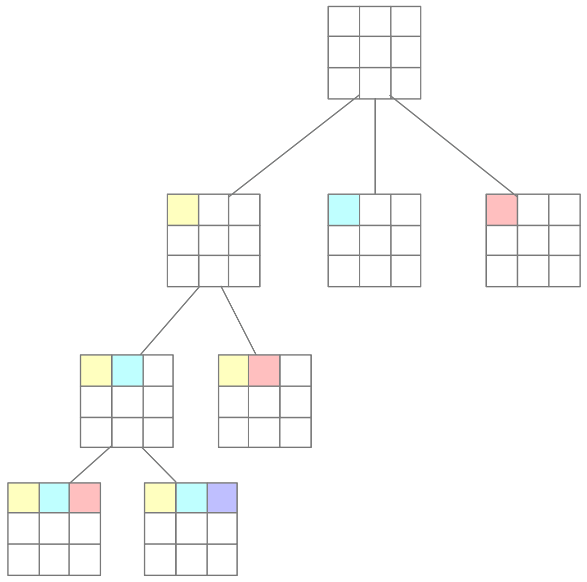
Problem Search with Back-Tracking

A tree of possibilities …

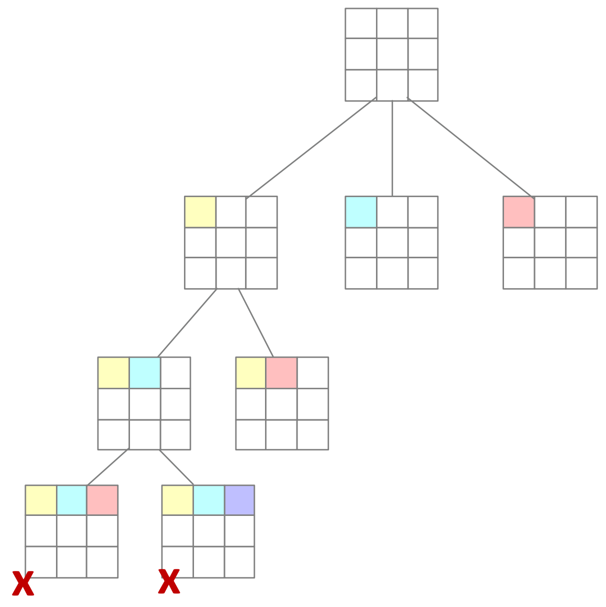


Basic recursive search



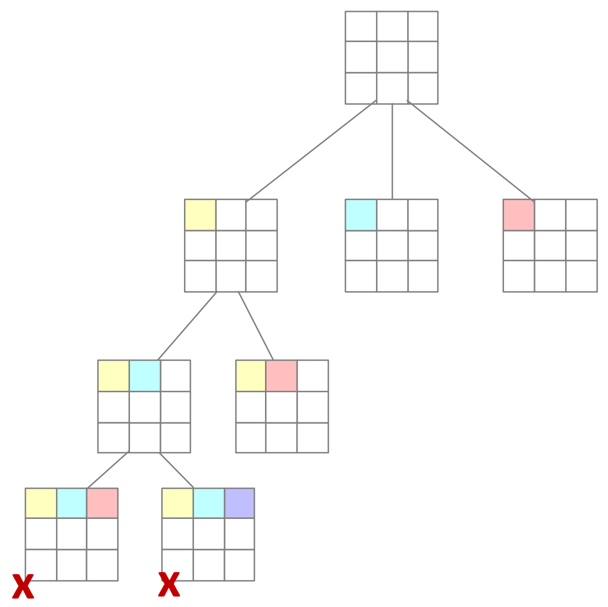
```
Search( partial solution ):
    if the partial solution is complete:
        return True
    for each possible next step:
        Apply the step to partial solution
        if Search(partial solution):
            return True
        else:
            Undo the step
    # All possible next steps have failed
    return False
```

Recursive search with pruning



```
Search( partial solution ):
    if the partial solution is complete:
        return True
    if the partial solution can't possibly work:
        return False
    for each possible next step:
        Apply the step to partial solution
        if Search(partial solution):
            return True
        else:
            Undo the step
    # All possible next steps have failed
    return False
```

Design decisions



Representation?

Feasibility checks?

```
Search( partial solution ):
    if the partial solution is complete:
        return True
    if the partial solution can't possibly work:
        return False
    for each possible next step:
        Apply the step to partial solution
        if Search(partial solution):
            return True
        else:
            Undo the step
    # All possible next steps have failed
    return False
```

How do we choose steps?
Good order to try them?

How do we undo steps?
Save and restore?

Choices for Sudoku

Search by itself would be expensive. Search with constraint propagation is much faster (usually).

Representation?

Sudoku board, partly filled

Feasibility checks?

Check while performing constraint propagation

How do we choose steps?
Good order to try them?

Try all candidates for a Tile
Pick a Tile with few candidates

How do we undo steps?
Save and restore?

Save (with Board.as_list) and Restore (with Board.set_tiles)