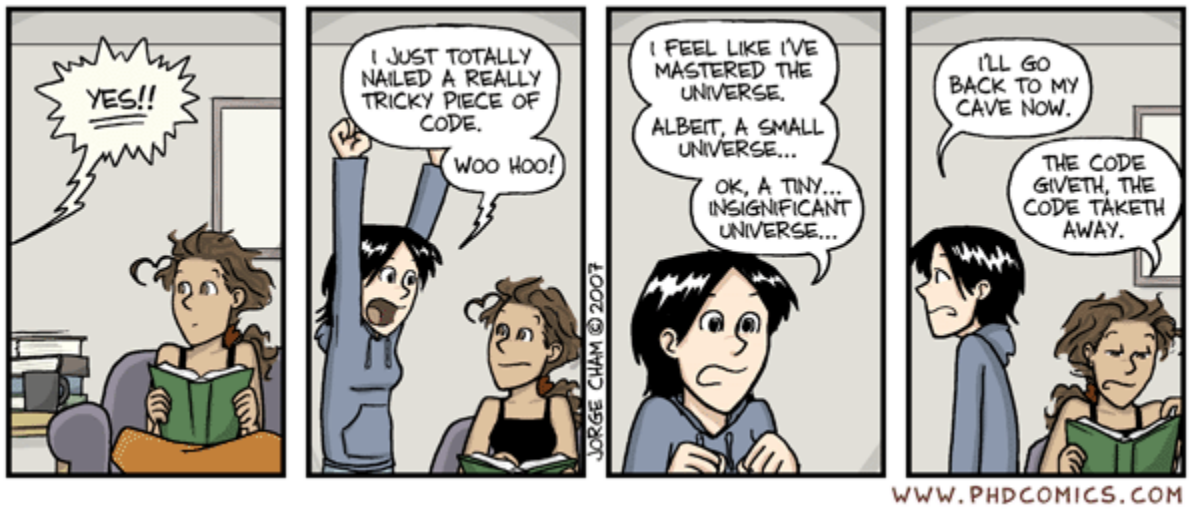
CSE 143

read: 12.5

Lecture 17: recursive backtracking

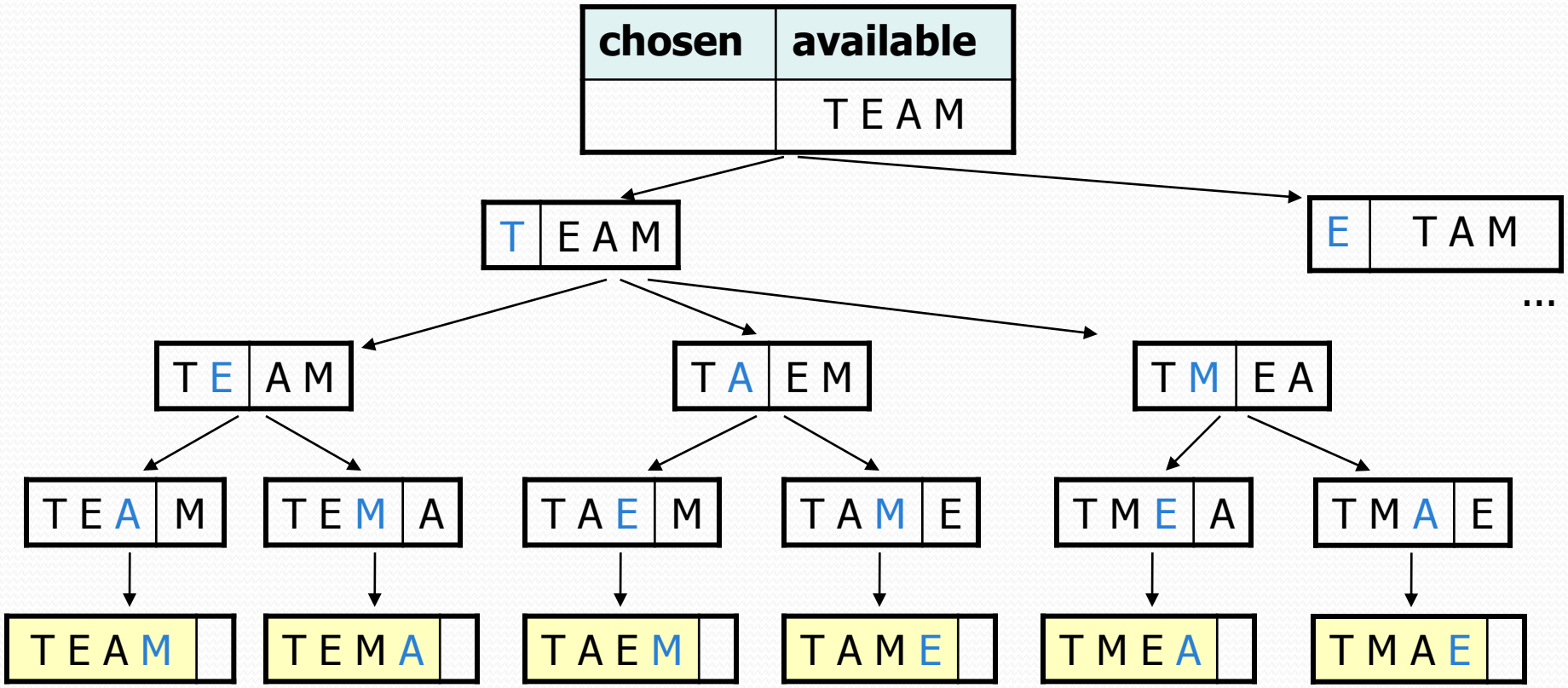


Exercise: Permutations

- Write a method `permute` that accepts a string as a parameter and outputs all possible rearrangements of the letters in that string. The arrangements may be output in any order.
  - Example: `permute("TEAM")` outputs the following sequence of lines:

```
TEAM    ATEM
TEMA    ATME
TAEM    AETM
TAME    AEMT
TMEA    AMTE
TMAE    AMET
ETAM    MTEA
ETMA    MTAE
EATM    META
EAMT    MEAT
EMTA    MATE
EMAT    MAET
```

Decision tree



Backtracking

- Useful to solve problems that require making decisions
  - Each decision leads to new choices
  - Some (but not all!) sequence(s) of choices will be a solution
  - Insufficient information to make a thoughtful choice
- Systematically prune out infeasible solutions

Exercise: solve maze

- Write a method `solveMaze` that accepts a `Maze` and a starting row/column as parameters and tries to find a path out of the maze starting from that position.
  - If you find a solution:
    - Your code should **stop** exploring.
    - You should **mark** the path out of the maze on your way back out of the recursion, using backtracking.
  - (As you explore the maze, squares you set as 'explored' will be printed with a dot, and squares you 'mark' will display an X.)

```
##########
#    xx  #
# ###x## #
# # xx # #
# # x# # #
# ##x#####
# #.xx   #
# #.#x # #
#####x####
#...#xxxx?
#.#..xx#.#
##########
```

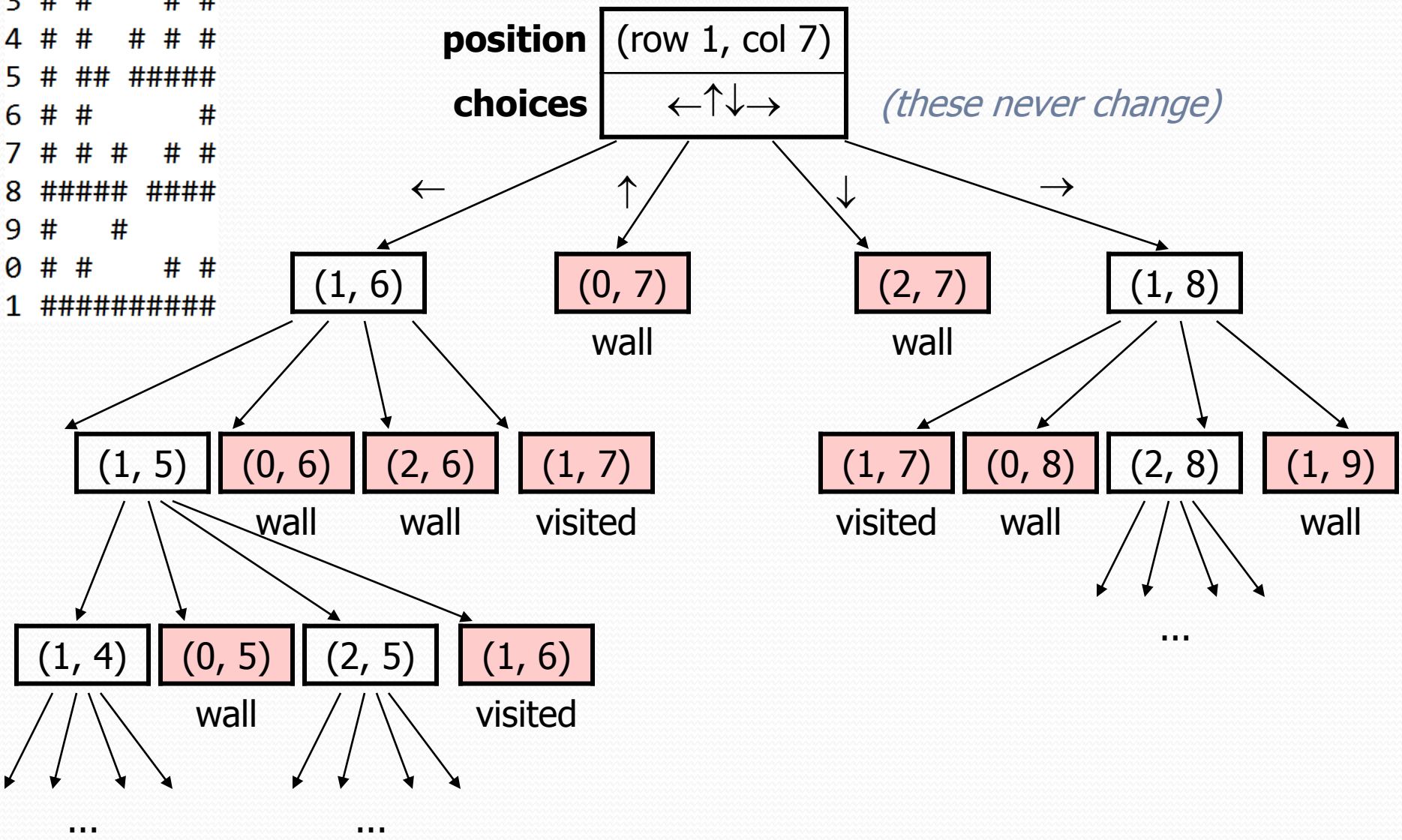
Maze class

- Suppose we have a `Maze` class with these methods:

Method/Constructor	Description
`public` **`Maze`**`(String text)`	construct a given maze
`public int` **`getHeight`**`(),` **`getWidth`**`()`	get maze dimensions
`public boolean` **`isExplored`**`(int r, int c)` `public void` **`setExplored`**`(int r, int c)`	get/set whether you have visited a location
`public void` **`isWall`**`(int r, int c)`	whether given location is blocked by a wall
`public void` **`mark`**`(int r, int c)` `public void` **`isMarked`**`(int r, int c)`	whether given location is marked in a path
`public String` **`toString`**`()`	text display of maze

```
  0123456789
0 ##########
1 #        #
2 # ### ## #
3 # #    # #
4 # #  # # #
5 # ## #####
6 # #      #
7 # # #  # #
8 ##### ####
9 #   #
0 # #    # #
1 ##########
```

Decision tree



Recall: Backtracking

A general pseudo-code algorithm for backtracking problems:

Explore(**choices**):

- if there are no more **choices** to make: stop.
- else, for each available choice **C**:
  - Choose **C**.
  - Explore the remaining **choices**.
  - Un-choose **C**, if necessary. (backtrack!)