ALGORITHMS & DATA STRUCTURES

1st November

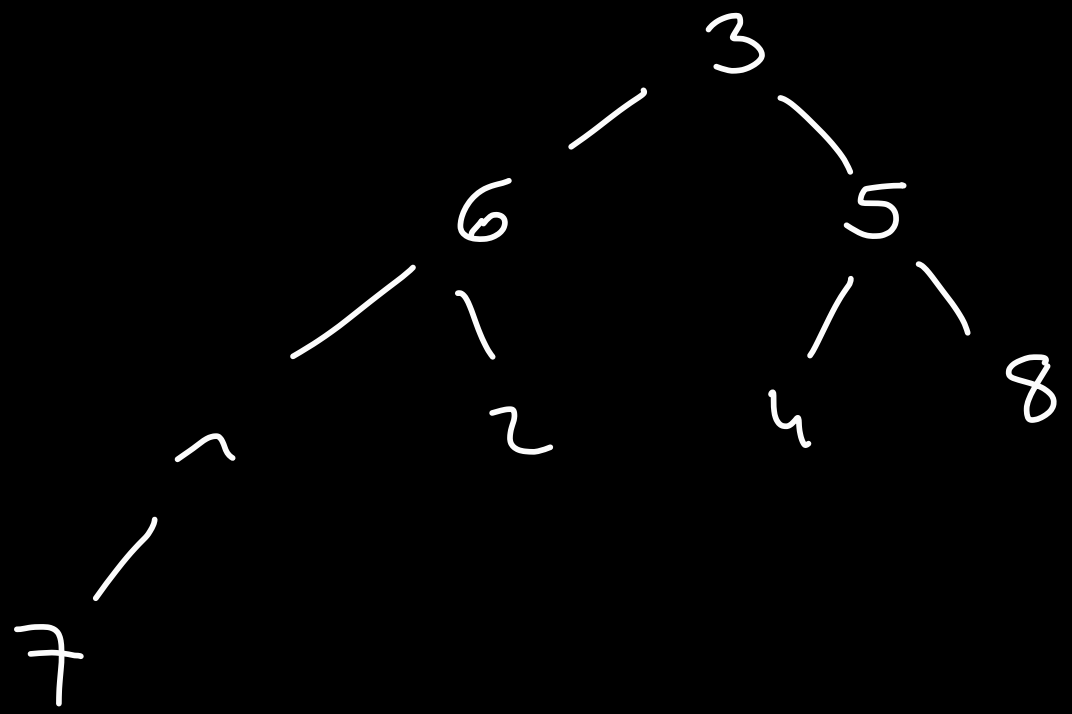
PLAN FOR TODAY

- Bonus Exercises
- Dynamic Programming

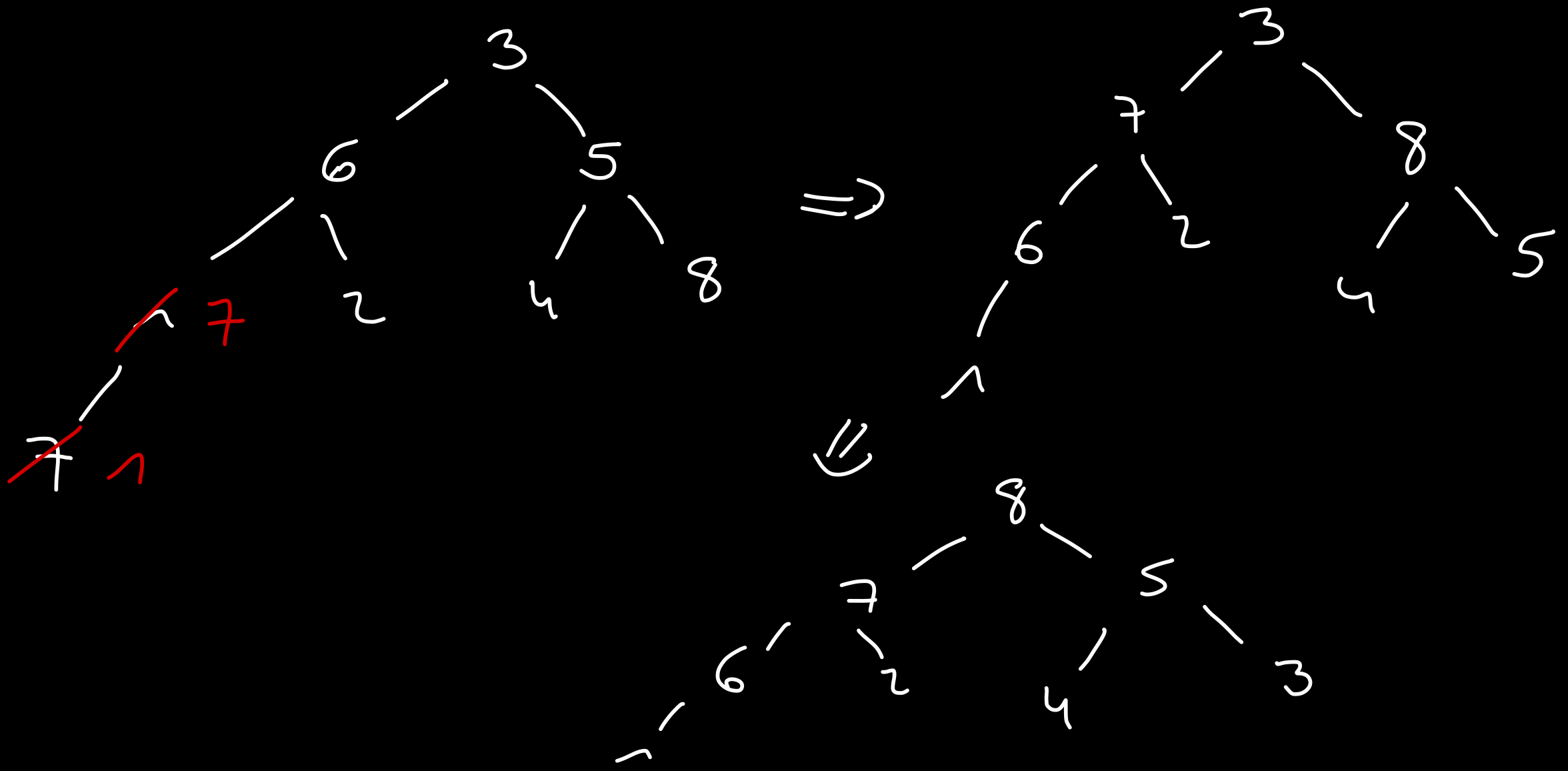
EXERCISE 5.1

Given the array $[3, 6, 5, 1, 2, 4, 8, 7]$, we want to sort it in ascending order using Heapsort.

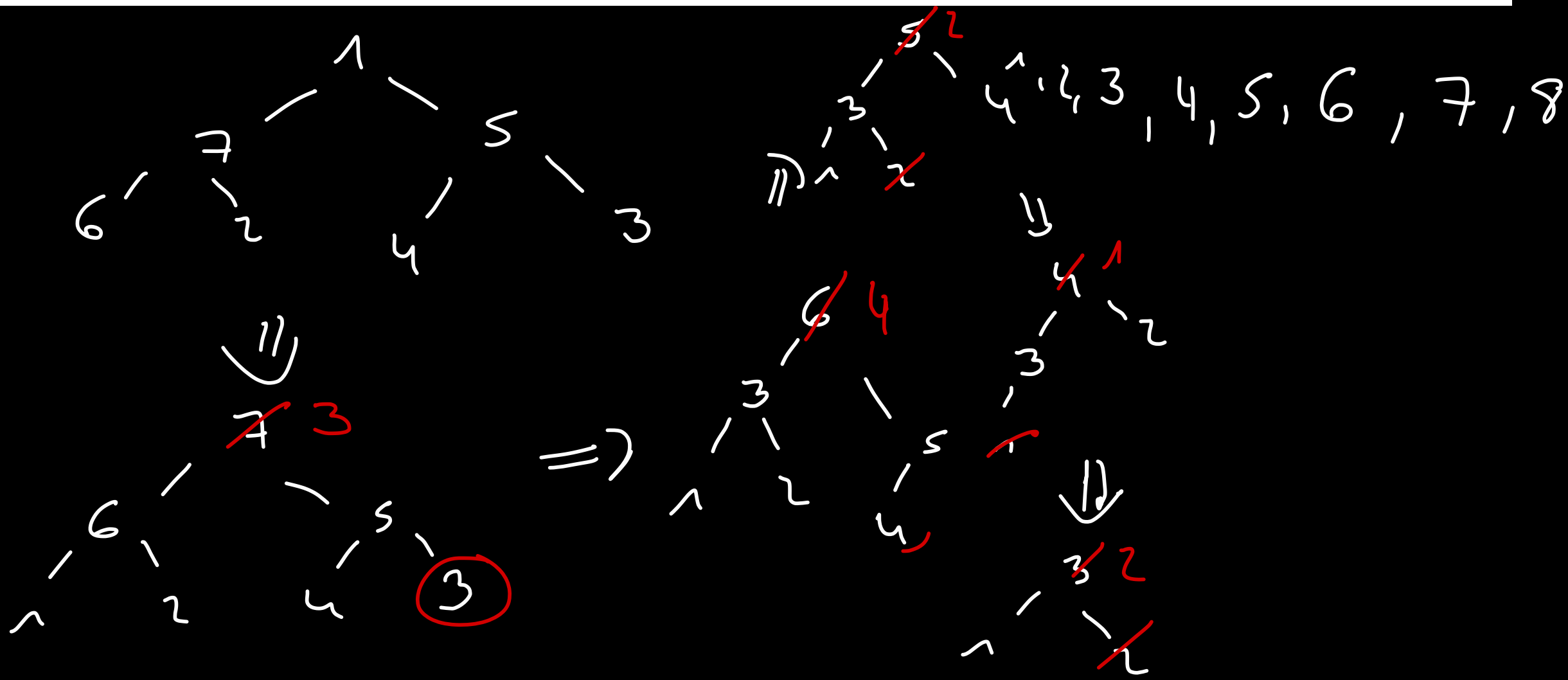
a) Draw the tree interpretation of the array as a heap, before any call of RestoreHeapCondition.



b) In the lecture you have learned a method to construct a heap from an unsorted array (see also pages 35–36 in the script). Draw the resulting max heap if this method is applied to the above array.



c) Sort the above array in ascending order with heapsort, beginning with the heap that you obtained in (b). Draw the array after each intermediate step in which a key is moved to its final position.



EXERCISE 5.2

3	6	5	1	2	4	8	7
3	6	5	1	2	4	8	7
3	5	6	1	2	4	8	7

Insertion Sort

3	6	5	1	2	4	8	7
3	5	1	2	4	6	7	8
3	1	2	4	5	6	7	8

Dubble Sort

3	6	5	1	2	4	8	7
3	6	1	5	2	4	7	8
1	3	5	6	2	4	7	8

Merge Sort

3	6	5	1	2	4	8	7
1	6	5	3	2	4	8	7
1	2	5	3	6	4	8	7

Selection Sort

EXERCISE 5.5

Alice and Bob are playing a game where Alice chooses a secret integer $x \in \{1, \ldots, n\}$ and Bob has to guess the parity of x, i.e., Bob has to guess whether x is even or odd. Note that n is a fixed number that Alice and Bob agree on before starting the game. Bob is allowed to ask Alice comparison questions of the form

"Is x greater than y?"

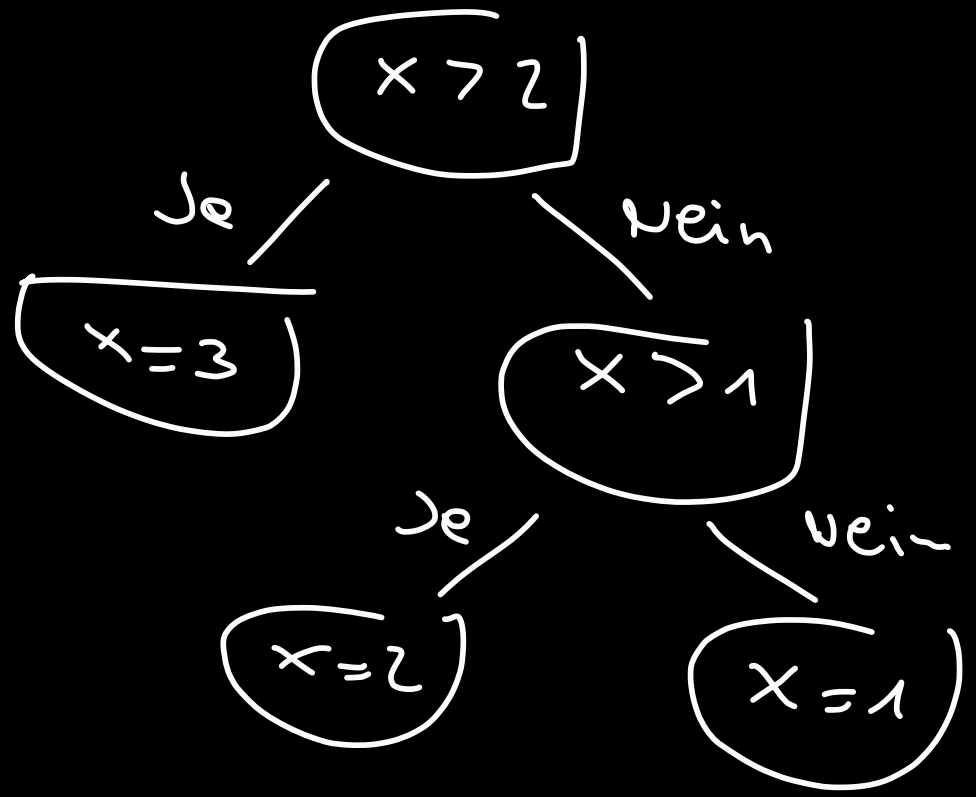
for some $y \in \{1, \ldots, n\}$. Bob is not allowed to ask other forms of questions.

The following is an example of how the game could start:

- Alice and Bob agree on $n = 1000$.
- Alice secretly chooses $x = 541$.
- Bob asks: "Is x greater than 50?"
  - Alice answers "yes".
- Bob asks: "Is x greater than 659?"
  - Alice answers "no".

We emphasize that Bob does not have to guess the exact value of x. He only needs to find the parity of x, and he wishes to achieve this by asking as few questions as possible.

a) Assume that $n = 3$. Devise a strategy of questions for Bob and draw its decision tree



b) Now n is arbitrary. Bob has asked i comparison questions and Alice answered these questions. Let $\mathcal{X}_i \subseteq \{1, \ldots, n\}$ be the collection of all numbers that are consistent with Alice's answers[1]. Show that $\mathcal{X}_i$ is a contiguous subset of $\{1, \ldots, n\}$, i.e., there exist $a, b \in \{1, \ldots, n\}$ such that

$$\mathcal{X} = \Big\{ y \in \{1, \ldots, n\} : \; a \leq y \leq b \Big\}.$$

$i = 1$ $\quad \{1 \ldots n\}$

Falls $x > c$: $\{c+1 \ldots n\}$

Falls $x \leq c$: $\{1 \ldots c\}$

IH ...

IS $i \longrightarrow i+1$ Nach i Fragen die Kandidaten sind $\{a \ldots b\}$

Falls $c \notin \{a \ldots b\} \Rightarrow$ Kandidaten verändern sich nicht und Beh. stimmt

Sonst: $x > c$ $\quad \{c+1 \ldots b\}$

$x \leq c$ $\quad \{a \ldots c\}$

c) Show that in any strategy of questions that Bob can follow, Bob cannot reliably guess the parity of x without reliably guessing the number x itself.

Hint: *Show that after i questions, Bob cannot reliably guess the parity of x unless $\mathcal{X}_i$ contains a single number.*

Nach i Fragen die "Kandidaten" sind

$S = \{a \dots b\}$

Falls $|S| \neq 1$ wir können nicht entscheiden,

ob x gerade/ungerade ist

d) Show that in any strategy of questions that Bob can follow, the number of questions that are required to reliably guess the parity of x is at least $\lceil \log_2(n) \rceil$ in the worst case.

- Jede Strategie ist ein Entscheidungsbaum.
- Der Baum hat $\geq n$ Blätter
- Ein Baum de Höhe h hat $\leq 2^h$ Blätter

$h=0$ 0 ✓.

IH ...

$h \to h+1$ Ein Baum der Höhe $h+1$ hat $\leq 2 \cdot 2^h$ Blätter

IH

$$2^h \geq n \Rightarrow h \geq \lceil \log_2 n \rceil$$

DYNAMIC PROGRAMMING

LEARNING BY DOING

TWO WAYS TO SOLVE A PROBLEM

Iterative

Vorhe Tabelle T

Wie berechnet men T[i]?

Wo ist die Lösung?

Recursive + Memoization

- Löst rekursiv
- Speichert Ergebnisse in einer Tabelle, so dass wir "overlapping subproblems" nicht mehrmals berechnet.

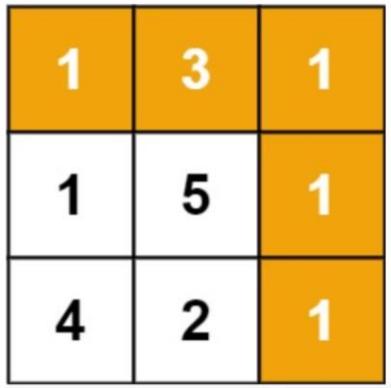
64. Minimum Path Sum

Medium 5973 89 Add to List Share

Given a `m x n grid` filled with non-negative numbers, find a path from top left to bottom right, which minimizes the sum of all numbers along its path.

Note: You can only move either down or right at any point in time.

Example 1:




```
Input: grid = [[1,3,1],[1,5,1],[4,2,1]]
Output: 7
Explanation: Because the path 1 → 3 → 1 → 1 → 1 minimizes the sum.
```

```java
class Solution {
    public int algo(int[][] M, int[][] grid, int x, int y){
        int n = grid.length;
        int m = grid[0].length;

        if(x == n - 1 && y == m - 1) return grid[x][y];
        if(M[x][y] != -1) return M[x][y];

        int res = 10000;
        if(x + 1 < n)
            res = Math.min(res, grid[x][y] + algo(M, grid, x + 1, y));
        if(y + 1 < m)
            res = Math.min(res, grid[x][y] + algo(M, grid, x, y + 1));
        M[x][y] = res;
        return res;
    }
    public int minPathSum(int[][] grid) {
        int[][] M = new int[grid.length][grid[0].length];
        for(int i = 0; i < grid.length; i++){
            for(int j = 0; j < grid[0].length; j++) M[i][j] = -1;
        }
        return algo(M, grid, 0, 0);
    }
}
```

Testcase | Run Code Result | Debugger

Accepted Runtime: 0 ms

Your input `[[1,2,3],[4,5,6]]`

Output `12` Diff

Expected `12`

198. House Robber

Medium 8983 222 Add to List Share

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and **it will automatically contact the police if two adjacent houses were broken into on the same night**.

Given an integer array `nums` representing the amount of money of each house, return *the maximum amount of money you can rob tonight* ***without alerting the police***.

Example 1:

```
Input: nums = [1,2,3,1]
Output: 4
Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).
Total amount you can rob = 1 + 3 = 4.
```

Example 2:

```
Input: nums = [2,7,9,3,1]
Output: 12
Explanation: Rob house 1 (money = 2), rob house 3 (money = 9) and rob
house 5 (money = 1).
Total amount you can rob = 2 + 9 + 1 = 12.
```

```java
class Solution {
    int algo(int[] M, int[] nums, int p){
        if(p >= nums.length) return 0;
        if(M[p] != -1) return M[p];
        int res = nums[p] + algo(M, nums, p + 2);
        if(p + 1 < nums.length)
            res = Math.max(res, nums[p + 1] + algo(M, nums, p + 3));
        M[p] = res;
        return res;
    }
    public int rob(int[] nums) {
        int[] M = new int[nums.length];
        for(int i = 0; i < nums.length; i++) M[i] = -1;
        return algo(M, nums, 0);
    }
}
```

Testcase | Run Code Result | Debugger

Accepted Runtime: 0 ms

Your input [1,2,3,1]

Output 4 Diff

Expected 4

1 2 100 0 1 100