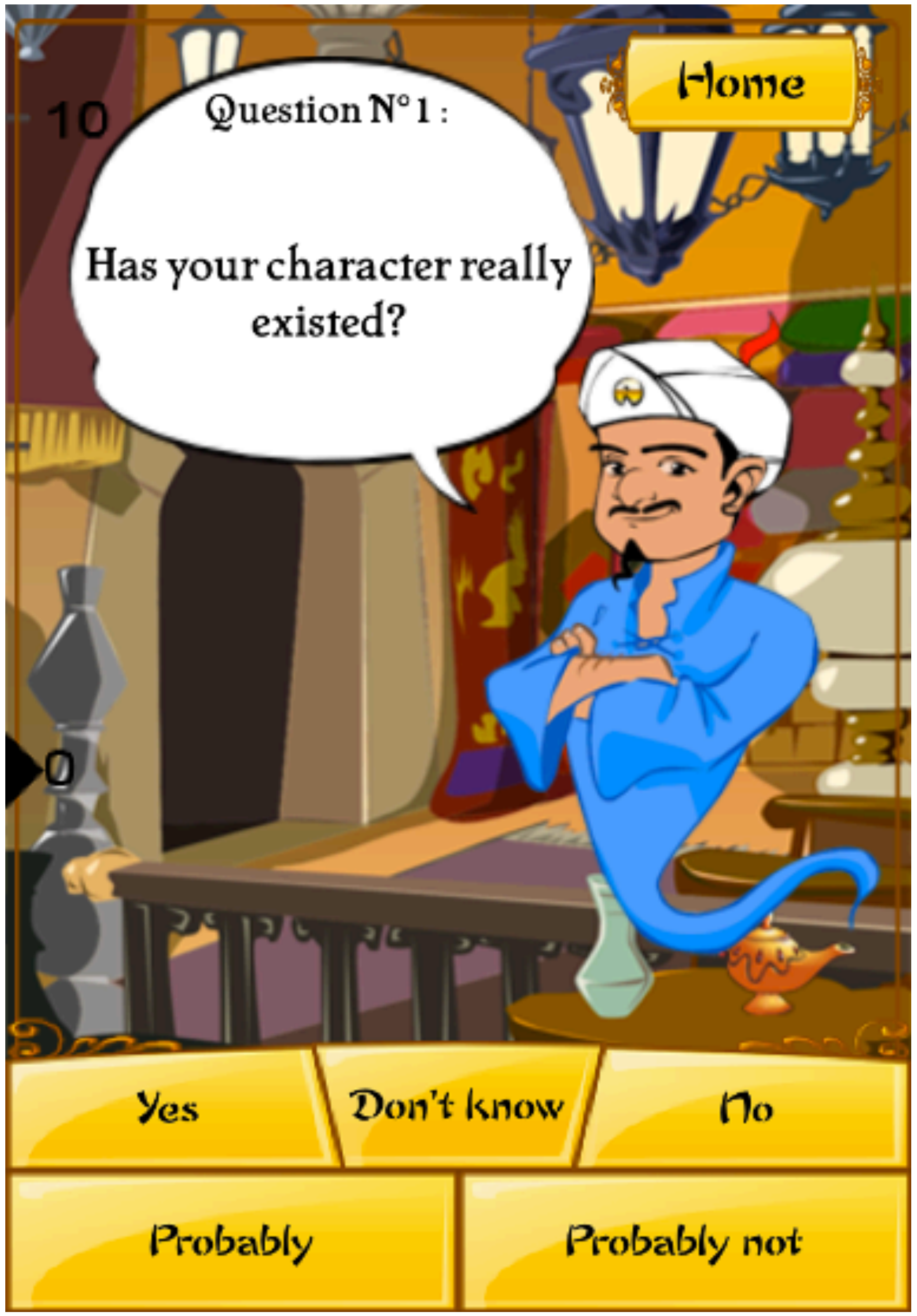
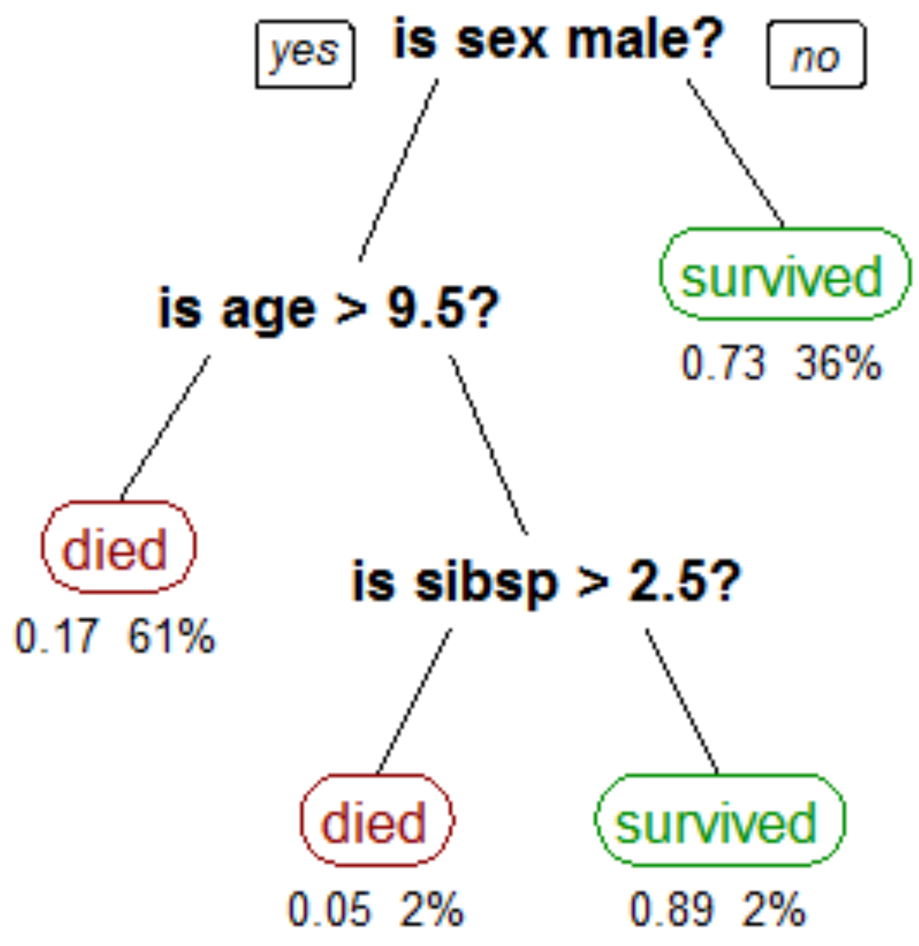
Lower bounds on sorting

CS 146 - Spring 2017





A tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard). The figures under the leaves show the probability of outcome and the percentage of observations in the leaf.

http://www.freemake.com/blog/the-akinator-game-play-ask-and-answer-with-arabic-genius/
https://en.wikipedia.org/wiki/Decision_tree_learning

All sorting algorithms we have seen so far are **comparison sorts**:

- only use comparisons to determine relative order of elements
- insertion sort, selection sort, mergesort, quicksort, heapsort
- in particular, mergesort is worst case n log n - n
- can we do better?

Decision tree model for sorting

- input: sort [a1, a2, a3, a4] with mergesort
- each internal node labeled i:j
- models all possible executions of a comparison sort
    - one tree per input size per algorithm
    - 1 execution = 1 root-to-leaf path
    - 1 leaf = outcome of the execution
    - running time on input = length of root

Any decision tree that can sort n elements must have height Omega(n log n)

- tree must contain ≥ n! leaves, one per permutation.
- relationship between height #leaves: **#leaves ≤ 2^h**.
- **height ≥ log(n!) ≥ n log n - n** (by Stirling's formula)

Corollary

- heapsort and mergesort are asymptotically optimal comparison-based sorting algorithms.

Forget about these algorithms… how would you sort…

1 trillion integers, all of which are between 1 and 10?