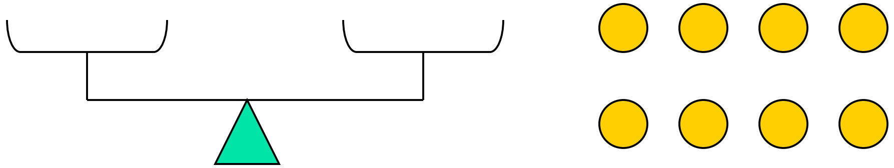
Chapter 5
Lower Bounds

References:
[CLRS 8.1]

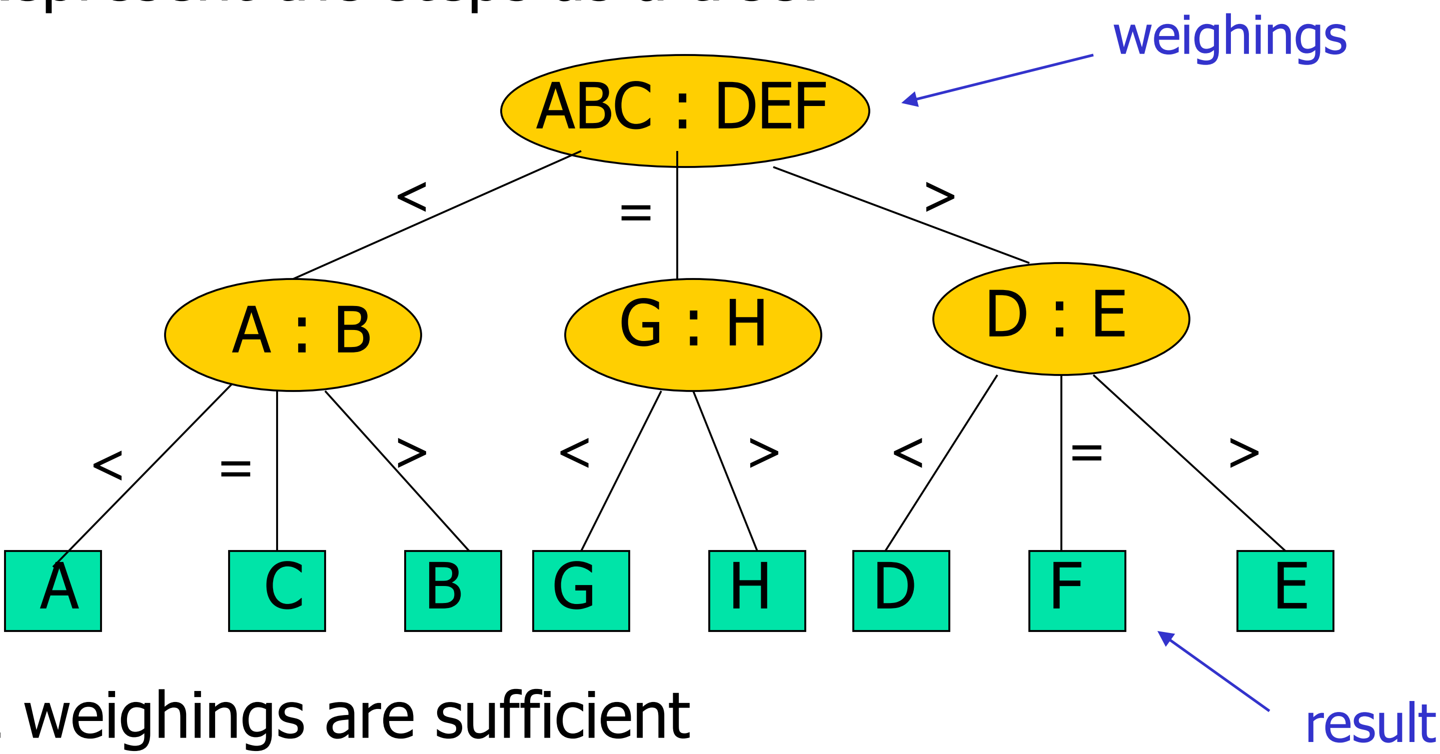
Coin Weighing Revisited

- Given:
  - 8 coins, one of which is counterfeit
  - Counterfeit coin is lighter
  - A pan balance (shows lighter/equal/heavier)
- Goal: to find the counterfeit coin using as few weighings as possible
- How? [upper bound]
  - 3 weighings?
  - 2 weighings?



Algorithm for 8 Coins

- Represent the steps as a tree:



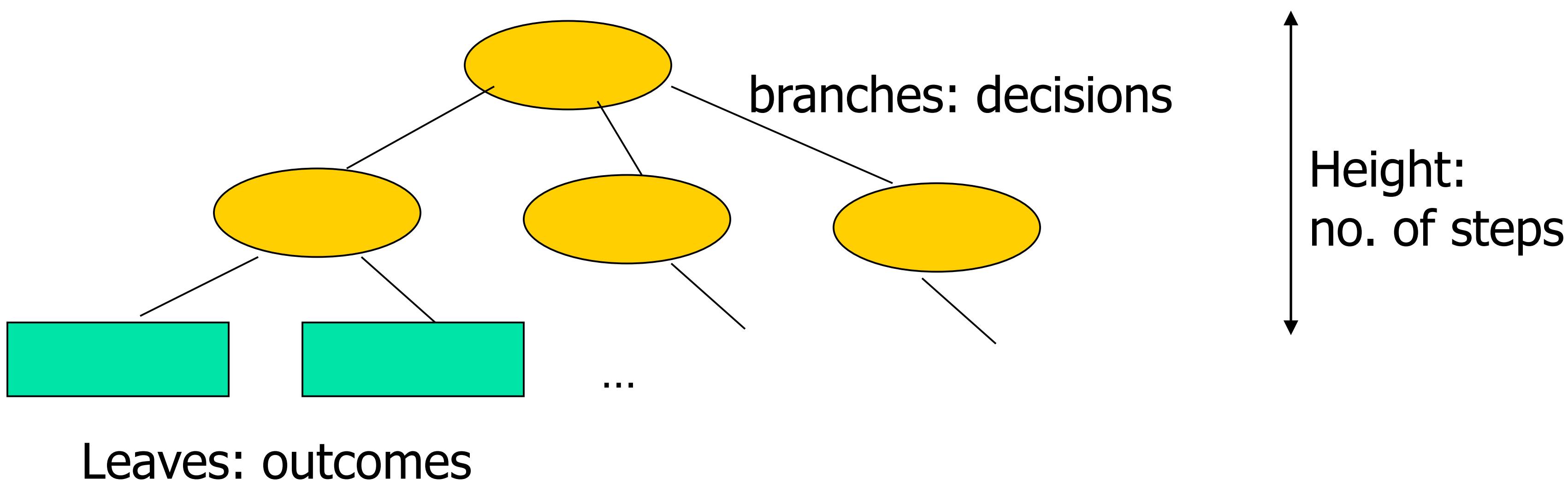
- 2 weighings are sufficient

Is 2 weighings Optimal?

- Can we do it in fewer than 2 weighings?
- Recall the definition of *lower bound:* the number of operations necessary to solve a problem by any algorithm
- We prove: no algorithm can find the counterfeit coin in fewer than 2 weighings (a lower bound of 2)
- Rough idea of argument:
  - There are 8 different configurations/outcomes
  - Each weighing differentiates 3 possibilities
  - So, 1 weighing $\rightarrow$ 3 outcomes, 2 weighings $\rightarrow$ 3x3 = 9 outcomes

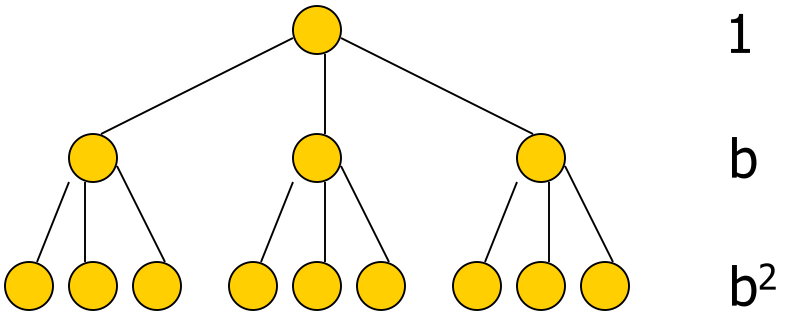
Decision Tree

- We can represent the actions of an algorithm using a decision tree:
  - Number of leaves = no. of different outcomes
  - No. of branches = no. of outcomes per decision (step)
  - Height = number of decisions (step)



Decision Tree Model

- All algorithms representable by a tree
- Lower bound on height of any tree = lower bound on worst-case number of steps by any algorithm
- How to lower-bound the height of a tree?
- In general, for a tree with branching factor b, height h and n leaves
  - Level k has b^k nodes
  - $b^h \geq n$
  - So $h \geq \lceil \log_b n \rceil$



Lower Bound for Coin Weighing

- What about n coins? What is the minimum number of weighings necessary by any algorithm?
  - Number of different outcomes: n
  - Number of branches: 3
- Decision tree: n leaves with branch factor 3
  - Height $\geq \lceil \log_3 n \rceil$
- So, lower bound = $\lceil \log_3 n \rceil$

Coin Weighing Upper Bound

- Can you give an algorithm with matching upper bound?
  - Hint: divide and conquer!
- Algorithm:
  - Divide n coins into 3 groups, of size n/3 each
  - One weighing of two groups
  - Eliminate two groups
  - Recurse on remaining n/3 coins
  - Base case: 1 coin (no weighing)

Coin Weighing Algorithm

- In pseudocode:
  - Call CW(A, 1, n)

```
CW(A, i, j) // n coins
{
  if (i==j) return i // base case
  k := (j-i+1)/3
  Weigh A[i..i+k-1] and A[i+k..i+2k-1]
  if A[i..i+k-1] lighter
    CW(A, i, i+k-1);
  else if A[i+k..i+2k-1] lighter
    CW(A, i+k, i+2k-1);
  else // equal
    CW(A, i+2k, j);
}
```

Coin Weighing Algorithm: Analysis

- $T(n) = T(n/3) + 1, T(1) = 0$

$$
\begin{aligned}
T(n) &= T(n/3) + 1 \\
&= T(n/9) + 1 + 1 \\
&= \ldots \\
&= T(n/3^k) + k \\
&= T(1) + \log_3 n \\
&= \log_3 n
\end{aligned}
$$

Lower Bound of Sorting

- Recall that we studied O(n log n) time sorting algorithms
  - Are they optimal? (Are there better algorithms?)
- A trivial lower bound: $\Omega(n)$
  - Proof: you need to read all inputs!
- We now show that those algorithms are optimal, by giving an $\Omega(n \log n)$ lower bound
- *Comparison-based model:* only allowed operation is comparison of two elements
- Lower bound on number of comparisons

Decision Tree for Sorting

- Total n elements (assume all different)
- How many different outcomes?
  - 3 elements: 6 outcomes (a>b>c, a>c>b, b>a>c, b>c>a, c>a>b, c>b>a)
  - n elements? nx(n-1)x(n-2)x…x1 = n! (factorial)
- So number of leaves in decision tree = n!
- Each comparison: 2 possible outcomes (>, <)
- So height = $\lceil \log_2(n!) \rceil$

Log n! = Θ(n log n)

- This direction is easy:
  - $n! = 1 \times 2 \times \ldots \times n$
    $\leq n \times n \times \ldots \times n = n^n$
  - Hence $\log n! \leq \log n^n = n \log n = O(n \log n)$
- But what we need is the other direction:
  - $n! = 1 \times 2 \times \ldots \times (n/2) \times \ldots\ldots\ldots\ldots\ldots \times n$
    $\geq \quad (n/2) \times (n/2) \times \ldots \times (n/2) \geq (n/2)^{n/2}$
  - $\log n! \geq (n/2) \log (n/2) = (n/2)(\log n - 1) = \Omega(n \log n)$
- Hence $\log n! = \Omega(n \log n)$ is the lower bound of comparison-based sorting

Decision Tree for Sorting

- Decision tree of one possible algorithm (for 3 elements):