Section 5.1 Analyzing Algorithms

Let P be a problem and A an algorithm to solve P. The running time of A can be analyzed by counting the number of certain operations that are performed during its execution. This count may also depend on the size of an input, which depends on the problem.

Example. If P asks whether a given item is in a given list, then we might count the comparison operations executed by A, which will depend on the length of the list.

A *worst-case input* of size n is an input that causes A to execute the largest number of operations.

Example. A worst-case input for an algorithm to search a list for an item would be a list without the item.

Let $W_A(n)$ denote the maximum running time of A over all inputs of size n. W_A is called the *worst-case function* for A.

An algorithm A to solve P is *optimal in the worst case* if every algorithm B to solve P satisfies the inequality $W_A(n) \leq W_B(n)$ for all $n > 0$.

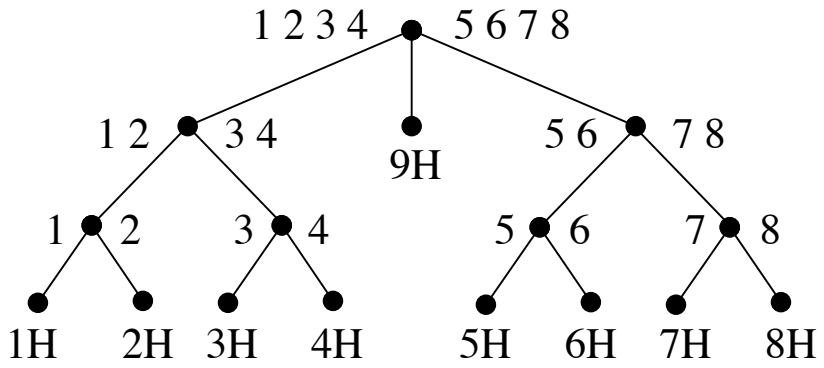
Method to find optimal algorithm in the worst case:

1. Find an algorithm A to solve P and do some analysis to find $W_A(n)$.
2. (*lower bound*) Find a function F where $F(n) \leq W_B(n)$ for all $n > 0$ and all B to solve P.
3. Compare F and W_A. If $F(n) = W_A(n)$ for all n, then A is optimal in the worst case. Otherwise try to find a better lower bound or a better algorithm.

A *decision tree* for an algorithm is a tree whose nodes represent decision points in the algorithm and whose leaves represent outcomes.

Example/Quiz. Given a set of nine coins, one of which is heavier than the others, use a pan-balance to find the heavy coin.

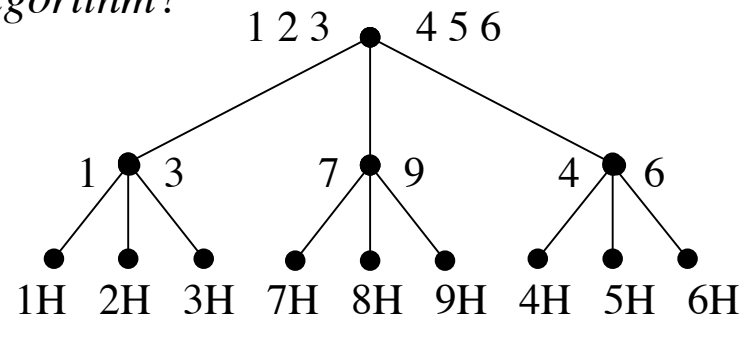
A Solution. Let the coins be 1, 2, …, 9. The numbers on either side of each internal node of the decision tree represent coins on the pan balance. The 9 outcomes are 1H, …, 9H.



The worst-case performance of this algorithm is 3 weighings.

Is the algorithm optimal in the worst case? A ternary decision tree with depth d has at most 3^d leaves. Since there are 9 possible outputs, it follows that $3^d \geq 9$. Take $\log_3$ to get $d \geq \log_3 9$. Since d is a natural number, $d \geq \lceil \log_3 9 \rceil = 2$. So a lower bound is 2.

Is there a better lower bound or a better algorithm?
Here's an optimal worst-case algorithm:



Example. Given a set of nine coins, one of which is bad, meaning that it is heavier or lighter than the others. Find a good lower bound for a pan-balance algorithm that finds the bad coin and states whether it is heavy or light.

A Solution. A lower bound: If the coins are numbered 1, 2, …, 9 and we let H mean heavy and let L mean light, then there are 18 possible outputs: 1H, 1L, …, 9H, 9L. A ternary decision tree with depth d has at most 3^d leaves. It follows that $3^d \geq 18$. Take $\log_3$ to obtain $d \geq \log_3 18$. Since d is a natural number, we have $d \geq \lceil \log_3 18 \rceil = 3$. So 3 is a good lower bound on the number of weighings.

Quiz. Find an optimal worst-case pan-balance algorithm to detect the one bad coin among a set of nine coins, where the output states whether the bad coin is heavy or light.

A Solution: