CS3230 Semester 1 2024/2025
Design and Analysis of Algorithms

Tutorial 04
D&C (2) + Decision Tree
For Week 05

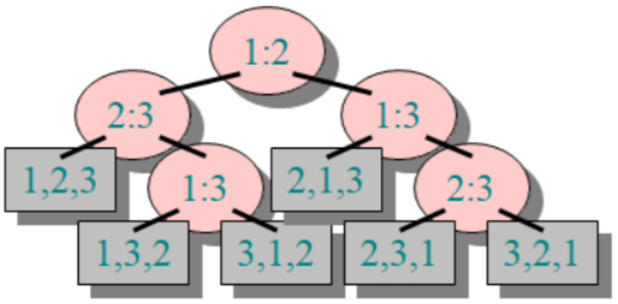
Document is last modified on: August 27, 2024

1 Lecture Review: Decision Tree

A decision tree contains:

- (Internal) Vertices: A comparison
- Branches: Outcome of comparison
- Leaves: Output / decision for the input

Worst case runtime is the height of the decision tree.



The classic example to illustrate the usage of decision tree is for showing the lower bound of comparison-based sorting is $\Omega(n \log n)$. Here is a picture of decision tree of sorting $n = 3$ elements and there are $3! = 6$ possible outputs (decision for the inputs) which must all been catered for. As each comparison of two comparable elements a versus b yields two possible outcomes: $a < b$ (which means a must be in front of b) or $a \geq b$ (which means b must be in front of a if $a > b$ or b can be in front of a if $a = b$ — does not matter here). This decision tree is thus a binary tree. The height of binary decision tree so that its number of leaves is at least $n!$ is $\log n! \approx n \log n$ (use Stirling's formula).

2 Tutorial 04 Questions

Q1, Q2, Q3, and Q4 (hidden) involve Polynomial Multiplication of two polynomials of degree n.
Let $A(x) = a_n \cdot x^n + \ldots + a_2 \cdot x^2 + a_1 \cdot x + a_0$.
Let $B(x) = b_n \cdot x^n + \ldots + b_2 \cdot x^2 + b_1 \cdot x + b_0$.
Let $C(x) = A(x) \times B(x) = c_{2n} \cdot x^{2n} + \ldots + c_2 \cdot x^2 + c_1 \cdot x + c_0$
Assume all coefficients a_i, b_i, c_i are all Integers.
Assume that all addition and multiplication operations of two Integers take $O(1)$ time.

We can compute the coefficients c_i of $C(x)$ in $O(n^2)$ using complete search:
For each $i \in [2n..0]$, $c_i = \sum a_j \cdot b_{i-j}$ where both j and $i-j$ are between 0 and n (inclusive).

Q1). Let $x = 10$ to make it easier to visualize this as a normal base 10 multiplication and $n = 2$.
Let $A(10) = 352 = 3 \cdot 10^2 + 5 \cdot 10 + 2$, i.e., $a_2 = 3, a_1 = 5, a_0 = 2$.
Let $B(10) = 221 = 2 \cdot 10^2 + 2 \cdot 10 + 1$, i.e., $b_2 = 2, b_1 = 2, b_0 = 1$.
Compute the coefficients of $C(10) = A(10) \times B(10) == 77\,792$ using the $O(n^2)$ algorithm above.

Q2). Suppose that you are given the following Divide and Conquer (D&C) algorithm:
Rewrite $A(x) = x^{\frac{n}{2}} \cdot A_1(x) + A_2(x)$
Rewrite $B(x) = x^{\frac{n}{2}} \cdot B_1(x) + B_2(x)$
where $A_1(x), A_2(x), B_1(x), B_2(x)$ are all now polynomials of degree (up to) $\frac{n}{2}$.

We now compute four smaller polynomial multiplications:
$A_1(x) \times B_1(x), A_1(x) \times B_2(x), A_2(x) \times B_1(x), A_2(x) \times B_2(x)$

And we compute:
$C(x) = x^n \cdot (A_1(x) \times B_1(x)) + x^{\frac{n}{2}} \cdot (A_1(x) \times B_2(x) + A_2(x) \times B_1(x)) + A_2(x) \times B_2(x)$

Apply this D&C algorithm to compute the multiplication of the same two polynomials of degree $n = 2$:
Rewrite $A(10) = 352 = 10 \cdot (3 \cdot 10 + 5) + 2$
Rewrite $B(10) = 221 = 10 \cdot (2 \cdot 10 + 2) + 1$

Q2). First, compute $A_1(10) \times B_1(10), A_1(10) \times B_2(10), A_2(10) \times B_1(10), A_2(10) \times B_2(10)$.
Then, compute $C(10)$.

Q3). What is the time complexity of that recursive D&C algorithm?

Q4). is hidden, but it is a small extension of Q1, Q2, and Q3 above.

Q5). H-index

In research (if one day you decide to do PhD in NUS or any other (top) University), one way to measure how good you are is by looking at your H-index[1].

Basically, each paper has a certain number of citations (a citation is a quotation from or reference

[1]Prof Steven Halim did not do domain research since 2010 and focusing on education, thus his 2024 h-index according to Google scholar is 11. Prof Chang Yi-Jun, on the other hand, is an active algorithm researcher, thus his 2024 h-index according to Google scholar is 19.

to a book, paper, or author, especially in a scholarly work). Your H-index is the largest number H such that you have written and published H papers with at least H citations. Given the number of citations c_i on each of the n papers you have published in non-increasing (decreasing or plateau), what is your H-index?

Design the best algorithm that you can think of to solve this problem as efficiently as possible. Analyze the time complexity in Big O notation.

Q6). Decision Tree

You are given x - the value of x will be revealed on the spot during tutorial - balls, all of which have the same weight, but one is heavier. Your job is to find which of the balls is heavier. Your friend has a balance scale, but will charge you for each weighing. You want to minimize the minimum number of weighings needed. What is the minimum number of weighings needed to find the ball?

You can assume that we only use comparison model (comparison returns $<$, $=$, or $>$). You can decide how to compare the ball(s). What is the lower bound of <u>any</u> algorithm to solve this problem?