Homework 1: Dil Rawat

October 31, 2025

1 [Not in a book]

Show how to sort a list of five elements using no more than seven comparisons.

Answer:
Let's say our 5 items are: n1, n2, n3, n4, n5:

We will divide the first four items into two pairs. Then sort each pair.

Sort n1 and n2. 1 comparison

Sort n3 and n4. 1 comparison

So far we have: $n1 < n2$ and $n3 < n4$

Now we compare with large items in each pair. n2 and n4. 1 comparison

If $n2 < n4$: our order: n1, n2, n3, n4

— We know that: $n1 < n2 < n4$.

— Insert n5. upto 2 comparison

— Then insert n3. upto 2 comparison

Else: we swap the pairs: n3, n4, n1, n2

— we know : $n3 < n4 < n2$

— Insert n5. upto 2 comparison

— Then insert n1. upto 2 comparison

Result: Sorted sequence: n1, n2, n3, n4, n5

Outcomes: CS 6, 5870-1

2 [GT A-9.3]

Suppose you are the postmaster in charge of putting a new post office in a small town, where all the houses are along one street, where the new post office should go as well. Let us view this street as a line and the houses on it as a set of real numbers, $\{x_1, x_2, \ldots, x_n\}$, corresponding to points on this line. To make everyone in town as happy as possible, the location, p, for the new post office should minimize the sum

$$\sum_{i=1}^{n} |p - x_i|$$

Describe an efficient algorithm for finding the optimal location for the new post office, show that your algorithm is correct, and analyze its running time.

Answer:

I am using Quick Select Algorithm to find the median and place the lamp post there.

Algorithm 1 quickSelect(S, m)

Require: Input: S as Set of n real numbers that corresponds a house on a line(street) , and an integer $m = \lceil n/2 \rceil$.

Ensure: Output: Location p for the new post office that minimize the sum distance from each house to the post office p.

1: **if** $n = 1$ **then**
2: return first element of S as the location for a new post office.
3: **end if**
4: pick a random element x of S
5: remove all the emelents from S and put them into 3 sequences:

- L, storing the elements in S less than x.
- E, storing the elements in S sequal to x.
- G, storing the elements in S greater than x.

6: **if** $m \leq |L|$ **then**
7: quickSelect(L, m)
8: **else if** $m \leq |L| + |E|$ **then**
9: return x
10: **else**
11: quickSelect$(G, m - |L| - |E|)$
12: **end if**

We place the post across the street when there is only one house. If two houses we can put the post in between houses. The sum of the distance to post office from those two houses is minimum if we put the post office in between those two houses.

If we got more than two houses, we find the median and keep going until we are left with one or two houses

Analysis: It depends on selected pivot. We are selecting it randomly in the above algorithm

Worst Case: If the pivot is the smallest or largest item in the Sequence S with n elements.

This means either:

$L = 0$, $|G| = n - 1$ or $L = n - 1$, $|G| = 0$

Since this involves recusion, we will need a recurrence equation:

$T(n)$ = Time to partition (L, E, R) + T(Size of Subproblem)

$T(n) = O(n) + T(p)$, where p is the partion

$T(n) = T(n-1) + O(n)$ We can use subsutition to solve this recurrence equation which will give us:

$T(n) = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$

This equates to: $T(n) = O(n^2)$

Average Case: In this case the pivot partition the sequence S at most 3n/4

$T(n) = T(\frac{3n}{4}) + O(n)$

We can use master theorem to solve this:

$a = 1, b = 4/3, log_b a = 1, f(n) = O(n)$

As per Case 3: $O(n)$

Therefore on average, the quick select algorithm runs on linear time.

Outcomes: CS 6, 5870-1

3 [GT C-9.8]

Show how a deterministic $O(n)$-time selection algorithm can be used to design a quicksort-like sorting algorithm that runs in $O(n \lg n)$ *worst case* time on an n-element sequence.

Answer:

DeterministicSelect Algo:

Divides the n items into a group of 5 items and finds their median.

Finds a median(pivot) of medians, that almost divides the n-element sequence into half.

i.e : $L = n/2$ and $G = n/2$. L has items less than pivot and G has greater ones.

We recursively run the DeterministicSelect in L and G until we hit base case of n = 1.

Then sorted items are concatenated together after the end of each recursive call giving us a sorted n-element sequece.

At each level we go throught the element and put them into 3 sequences: L,E, and G. This takes O(n).

Recurrence equation : $T(n) = 2T(n/2) + O(n)$

where, $a = 2, b = 2, f(n) = O(n), log_b a = 1$

So, as per rule 2 of MT: $O(nlogn)$

Outcomes: CS 6, 5870-1

4 [GT A-7.4]

The game of *Hex* is said to have, as one of its inventors, the mathematician John Nash, who is the subject of the book and movie *A Beautiful Mind*. In this game, two players, one playing black and the other playing white, take turns placing stones of their respective colors on an $n \times n$ hexagonal grid. Once a stone is placed, it cannot be moved. The black player's goal is to connect the top and bottom sides of the grid, and the white player's goal is to connect the left and right sides of the grid, using stones of their respective colors. Two cells are considered connected if they share an edge and both have the same color stone. (See Figure 7.12.) Describe an efficient scheme where you can determine after each move whether black or white has just won a game of Hex.

Answer:

We will have four nodes for: Top, Bottom, Left and Right sides.

A node for each cell in the grid.

Black Player's goal: Connect Top and Bottom

White Player's goal: Connect Left and Right

If the black stone is placed on Top or Bottom nodes we do:

Union(stone, Top) or Union(stone, Bottom)

Plus check if the neighbor has black stone.

If so, we do the Union with the neighbor.

If black stone is placed anywhere else, we check if it has black neighbor.

If so we do Union with neighbor.

We do same steps for the White.

To find the winner:

We check if Top and Bottom are in same set everytime black stone is placed. Find(Top) = Find(Bottom).

If so Black wins. This means top and bottom are connected with same color stones

We check if Left and Right are in same set everytime white stone is placed. Find(Left) = Find(Right).

If so White wins. This means left and right are connected with same color stones.

Outcomes: CS 6, 5870-1

5 [GT C-7.7]

Suppose we implement the tree-based union-find data structure using the union-by-size heuristic and path-compression heuristics. Show that the total running time for performing a sequence of m union and find operations, starting with n singleton sets, is $O(m)$, if $m \geq n \lg n$.

Answer:

Theorem 7.6: In a sequence α of m union and find operations performed using union by size andpath compression, starting with a collection of n single element sets, the total time to perform the operations in α is $O((n+m)\alpha(n))$.

In the question we are given:

$$m \geq nlgn$$
$$n \leq \frac{m}{logn}$$

Subsutite in Theorem 7.6 : $T(n,m) = O((n+m)\alpha(n))$

$$= O((\frac{m}{logn} + m)\alpha(n))$$

$$= O(m(\frac{1}{logn} + 1)\alpha(n))$$

$$= O(m(1)(1))$$

$$= O(m)$$

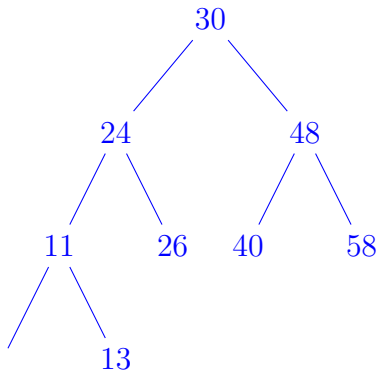
if $m \geq n \lg n$

Outcomes: CS 6, 5870-1

6 [GT R-4.1]

Consider the insertion of items with the following keys (in the given order) into an initially empty AVL tree: $30, 40, 24, 58, 48, 26, 11, 13$. Draw the final tree that results.

Answer:

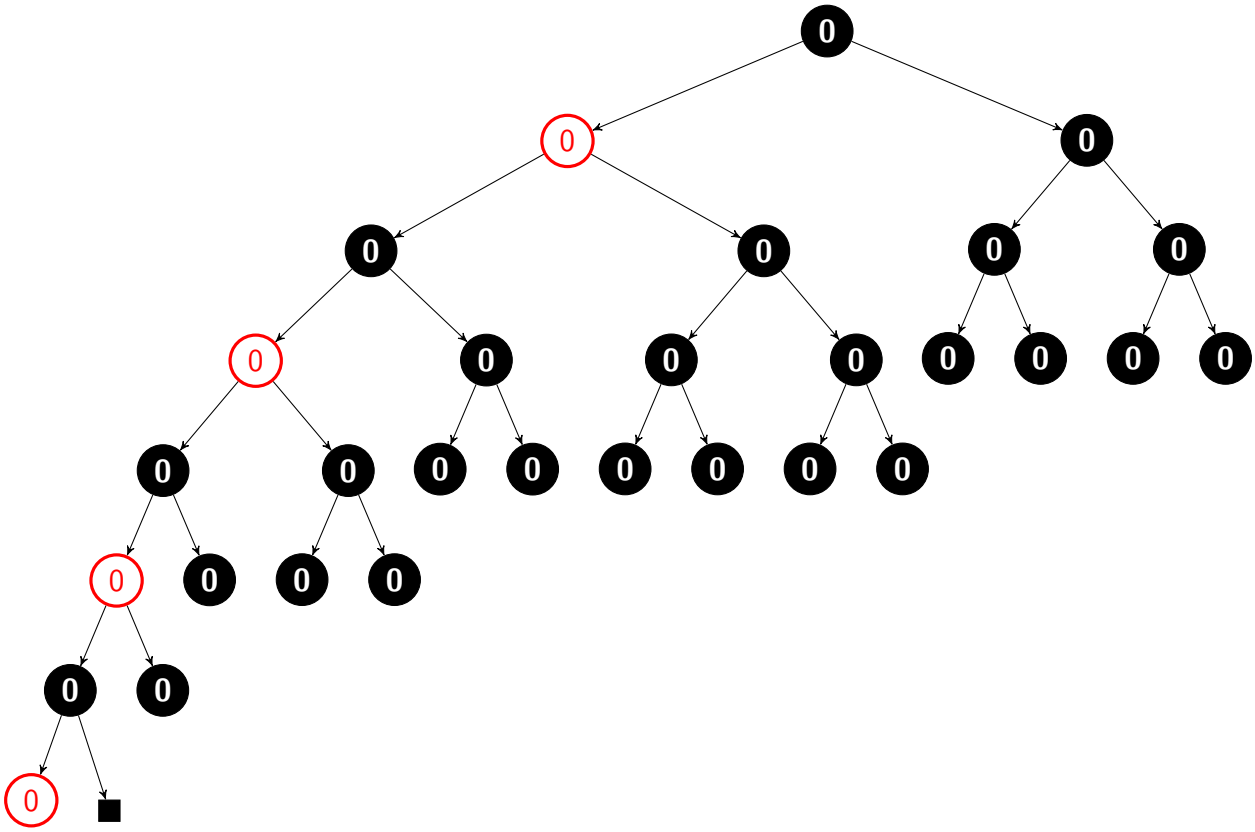


Outcomes: CS 6, 5870-1

7 [GT R-4.7]

What is the minimum number of nodes in a red-black tree of height 8?

Answer:



Min nodes: 30

Here we have a black node count of 4 when we visit every possible path. We don't have two reds.

Outcomes: CS 6, 5870-1

8 [SW 3.2.3]

Give five orderings of the keys **A X C S E R H** that, when inserted into an initially empty BST, produce the *best-case* tree.

Answer:

H C A E S R X

H S R X C A E

H C S A E R X

H S C R X A E

H S C A E R X

Outcomes: CS 6, 5870-1

9 [SW 3.2.11]

How many binary tree shapes of N nodes are there with height N? How many different ways are there to insert N distinct keys into an initially empty BST that result in a tree of height N?

Answer:

2^{N-1} ways.

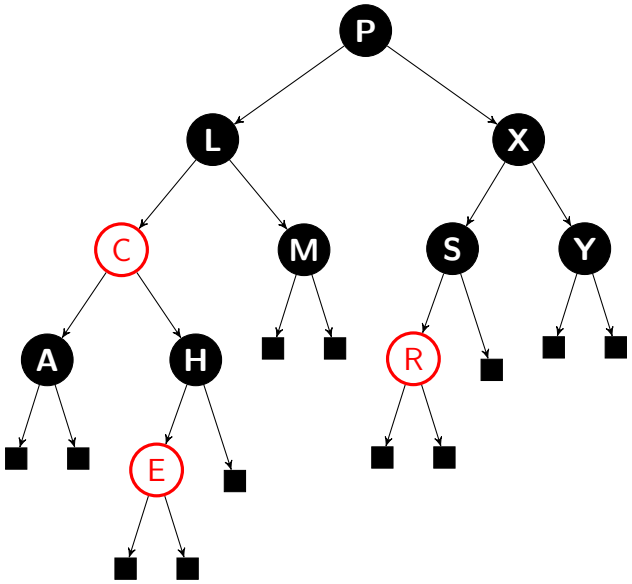
If $N > 1$, we have two choices at each node: either to insert on the left or right.

Outcomes: CS 6, 5870-1

10 [SW 3.3.11]

Draw the *left leaning* red-black BST that results when you insert items with the keys **Y L P M X H C R A E S** in that order into an initially empty tree.

Answer:



Outcomes: CS 6, 5870-1

11 [GT A-7.5]

Consider the game of Hex, as in the previous exercise, but now with a twist. Suppose some number, k, of the cells in the game board are colored gold and if the set of stones that connect the two sides of a winning player's board are also connected to $k' \leq k$ of the gold cells, then that player gets k' bonus points. Describe an efficient way to detect when a player wins and also, at that same moment, determine how many bonus points they get. What is the running time of your method over the course of a game consisting of n moves?

Answer:

One way to find a bonus points would be: after finding a winner, we can loop through the gold cells and check Find(goldCells) in the winning set.

This will add $O(k)$ additional run time on top of $O(n)$ time for finding winner. In total: $O(n + k)$.

Another apporach I can think of is: When a player places a stone on the board, we can keep count of gold cells. And when a winner is found we can use the count to give bonus points.

This will add more Find() operations. However, the total runtime should still be linear: $O(n)$.

Outcomes: CS 6, 5870-1

12 [GT C-4.3]

Show by induction that the minimum number, n_h of internal nodes in an AVL tree of height h, as defined in the proof of Theorem 4.1, satisfies the following identity, for $h \geq 1$:

$$n_h = F_{h+2} - 1$$

where F_k denotes the Fibonacci number of order k defined in the previous exercise.

Answer:

To show:

$$n_h = F_{h+2} - 1$$

Base case: $h = 1$

$n_1 = 1$

$F_{h+2} - 1 = F_{1+2} - 1 = F_3 - 1 = 2 - 1 = 1$

Base case: $h = 2$

$n_2 = 2$

$F_{2+2} - 1 = F_4 - 1 = 3 - 1 = 2$

Now assume $n_h = F_{h+2} - 1$ for $h = k$ and h $= k - 1$

To show: $n_{k+1} = F_{k+3} - 1$

Subsitute $h = k + 1$, $n_{k+1} = F_{k+3} - 1$

$n_{k-1} = F_{k+1} - 1 = F_{k-1} + F_k - 1$

Inductive Hypothesis: $n_k = F_{k+2} - 1 = F_k + F_{k+1} - 1$

From the theorem 4.1 in the book:

$n_h = 1 + n_{h-1} + n_{h-2}$

So, $1 + n_{k-1} + n_{k-2} = F_k + F_{k+1} - 1$

$n_{k+1} = 1 + n_k + n_{k-1} = 1 + (F_k + F_{k+1} - 1) + (F_{k-1} + F_k - 1)$

$n_{k+1} = F_{k+2} + F_{k+1} - 1$

$n_{k+1} = F_{k+3} - 1$

Hence, $n_h = F_{n+2} - 1$

Outcomes: CS 6, 5870-1