Exercise Set #4 (with Hints/Solutions)

Fall 2019, CMPUT 204
Department of Computing Science
University of Alberta

Problem 1.

- Show the result after each operatoon below, starting on an NIL BST: `Insert(16)`, `Insert(4)`, `Insert(3)`, `Insert(9)`, `Insert(1)`, `Insert(44)`, `Insert(29)`, `Delete(1)`, `Delete(4)`, `Insert(34)`.
- Draw an AVL tree of height 4 that contains the minimum number of nodes.
- Delete a leaf node (any leaf node) from the AVL tree you showed above so that the resulting tree violates the AVL-property. Then, apply tree-rotation to turn it to an AVL tree.

Problem 2.

a. Sorting lower bound can be argued by proof-by-contradiction. Suppose that there exists an algorithm making $< \log(n!)$ comparisons for *any* instance. Show that this leads to a contradiction.

b. Suppose we are given a sorted list of n numbers, $X = x_1 \leq x_2 \leq \cdots \leq x_n$, and we are asked to check whether or not there are any duplicates in the list. We are limited to algorithms which compare pairs of list elements (with a procedure `Compare`(i, j) which returns $<$, $>$ or $=$, depending on the values of x_i and x_j). Assume the algorithm returns either `Distinct` (if there are no duplicates) or "i-th element equals j-th element" if $x_i = x_j$. Note that if there are mulitple ties, any one can be reported. Prove a good lower bound (i.e., the largest lower bound possible) for the number of calls to `Compare` to solve this problem.

Problem 3. Exercise 11.2-2, 11.2-3 (CLRC p261)

Problem 4. Recall that in a binary tree each node has at most 2 children. Suppose T is a binary tree with $|T| = n$ (i.e. has n nodes). For every node v we use L_v and R_v to denote the left subtree and right subtree of v, respectively. We say the subtree rooted at v is *nearly balanced* if $\frac{1}{2} \leq |R_v|/|L_v| \leq 2$, i.e. the sizes of the two subtrees are within a factor two of each other.

a. Suppose that the *root* of T is nearly balanced. What is the maximum height of T in terms of n? Explain why.

b. We say the whole tree is *nearly balanced* if for every node v of T, the subtree rooted at v is nearly balanced. Denote h as the height of T. Show that if T is nearly balanced then $h \in O(\log n)$ by first showing that $h \leq \lceil \log_{\frac{3}{2}} n \rceil$.

Problem 5. We have a set J of n jars and a set L of n lids such that each lid in L has a unique matching jar in J. Unfortunately all the jars look the same (and all the lids look the same). We can only try fitting a lid ℓ to a jar j for any choice of $\ell \in L$ and $j \in J$; the outcome of such an experiment is either a) the lid is smaller, b) the lid is larger, or c) the lid fits the jar. We cannot compare two lids or two jars directly. Describe an algorithm to match all the lids and jars. The expected number of tries (experiments) of fitting a lid to a jar performed by your algorithm must be $O(n \log n)$.

Problem 6.

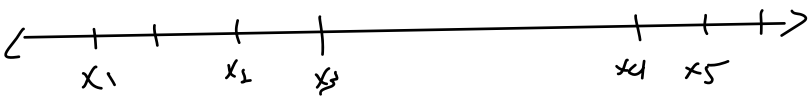
a. Describe an efficient greedy algorithm for making change for a specified value using a minimum number of coins, assuming there are four denominations of coins (called quarters, dimes, nickels, and pennies), with values 25, 10, 5, and 1, respectively. Argue why your algorithm is correct.

b. Give an example set of denominations of coins so that a greedy change making algorithm will not use the minimum number of coins.

Problem 7. In the art gallery guarding problem, a line L represents a long art gallery hallway. We are given a set of location points on it $X = \{x_0, x_1, \ldots, x_{n-1}\}$ of real numbers that specify the positions of the paintings along the hallway. A single guard can guard paintings standing at most 1 meter from each painting on either side. Propose an algorithm that finds the optimal number of guards to guard all the paintings along the hallway.

Problem 8. A native Australian named Oomaca wishes to cross a desert carrying only a single water bottle. He has a map that marks all the watering holes along the way. Assuming he can walk k miles on one bottle of water, design an efficient algorithm for determining where Oomaca should refill his bottle in order to make as few stops possible. Argue why your algorithm is correct.

P7.



Goal: design A that choose k locations such that every position is guarded and k is minimised.

A:

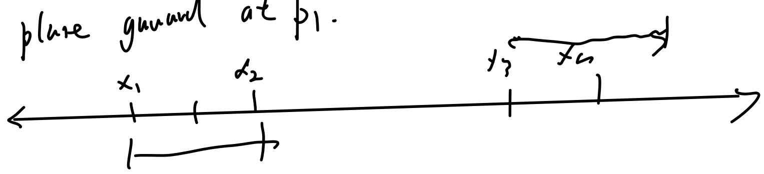
in iteration 1:

$l_1 = x_1 + l$

In subsequent iteration

let x be the leftmost unguarded position.

place guard at p_1.



proof that A is optimal:

let $S_1 = \{ l_1 \dots l_i \}$ is the locations A has chosen by the end of iteration i.

Idea: for each iteration i, S_i is contained in some optimal sol.

Base case. (iteration 1)

$S_1 = \{ l_1, \text{where } l_1 = x_1 + 1 \}$

any optimal sol must contain a guard in $[x_1 - 1, x_1 + 1]$ thus S_1 is contained in some optimal sol.

let $i \geq 1$ be some iteration, $S_i = \{ l_1, \ldots, l_i \}$ is contained in some optimal sol.

want to show:

for iteration $i+1$, $S_{i+1} = \{ l_1, \ldots, l_i, l_{i+1} \}$

is contained in an optimal sol.

by IH, there exists an optimal sol. such that S_i is contained in it

and so is l_{i+1} where $l_{i+1} \in [x_p - 1, x_p + 1]$

$l_{i+1} = x_p + 1$

so S_{i+1} is contained in an optimal sol b/c

$l_{i+1} \in [x_p - 1, x_p + 1]$ and l_{i+1} covers as many paintings

not guarded by S_i as possible.

p2b:

$x = x_1 \leq x_2 \leq \cdots \leq x_n$

↳ distinct

↳ "$i = j$" if $x_i = x_j$

lower bound:
minimum number of comparison required to solve the prob.

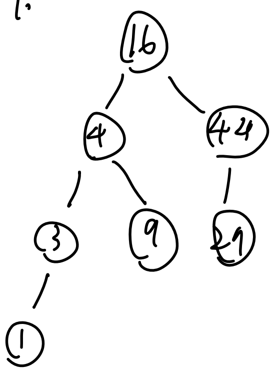
Sps we have an alg. A that performs $n-2$ comparisons

Then there exists a pair of consecutive elements that are not compared — $i, i+1$

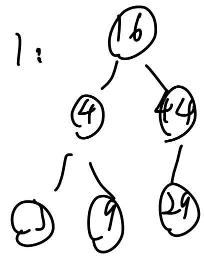
Problem 1.

- Show the result after each operatoon below, starting on an NIL BST: `Insert(16)`, `Insert(4)`, `Insert(3)`, `Insert(9)`, `Insert(1)`, `Insert(44)`, `Insert(29)`, `Delete(1)`, `Delete(4)`, `Insert(34)`.
- Draw an AVL tree of height 4 that contains the minimum number of nodes.
- Delete a leaf node (any leaf node) from the AVL tree you showed above so that the resulting tree violates the AVL-property. Then, apply tree-rotation to turn it to an AVL tree.

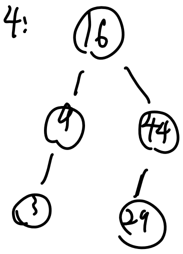
1.



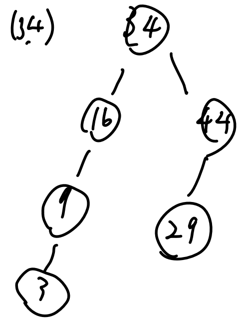
Delete 1:



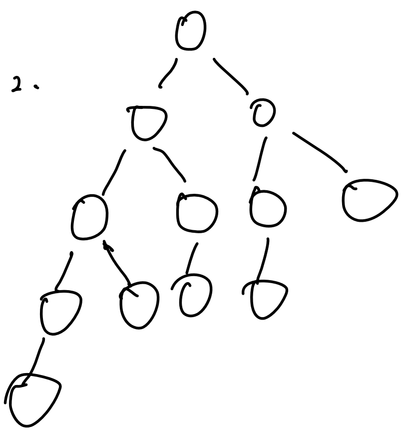
Delete 4:



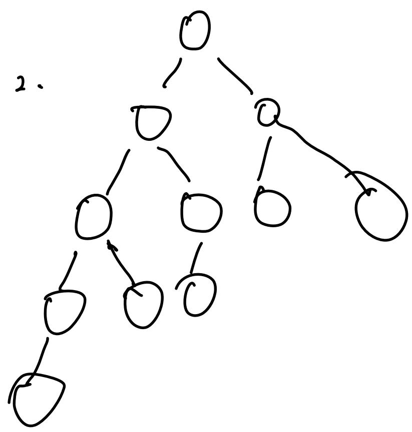
Insert (34)

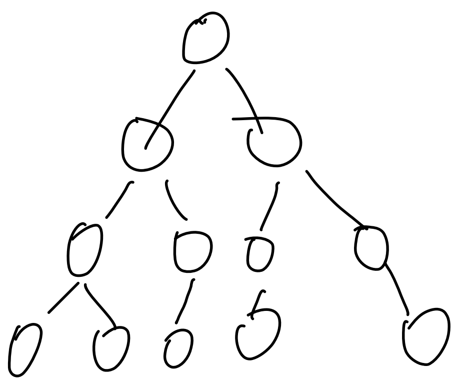


2.



2.





Problem 2.

a. Sorting lower bound can be argued by proof-by-contradiction. Suppose that there exists an algorithm making $< \log(n!)$ comparisons for *any* instance. Show that this leads to a contradiction.

b. Suppose we are given a sorted list of n numbers, $X = x_1 \leq x_2 \leq \cdots \leq x_n$, and we are asked to check whether or not there are any duplicates in the list. We are limited to algorithms which compare pairs of list elements (with a procedure Compare(i, j) which returns $<$, $>$ or $=$, depending on the values of x_i and x_j). Assume the algorithm returns either Distinct (if there are no duplicates) or "i-th element equals j-th element" if $x_i = x_j$. Note that if there are mulitple ties, any one can be reported. Prove a good lower bound (i.e., the largest lower bound possible) for the number of calls to Compare to solve this problem.

a. based on the algorithm is a comparison sequence, the algorithm outputs for a permutation of $(1, 2, \cdots, n)$. But there are $n!$ possible permutations by the assumption of $< \log(n!)$ comparisons, at least two permutations results in the same Yes/no sequence, which means the algorithm results the same output on both.

b. 1. assume there exists an Algorithm A that correctly finds the duplicates in $n-1$ times

2. let X be a list such that $x_i = 2i$ for $i = 1$ to n

3. use the algorithm A to input the list X. Since it only takes $(n-2)$ times. There is at least one element that is not compared to its next element

4. find that element $x_{i+1} = 2i$, but $x_{i+1} = 2(i+1)$)

5. The algorithm will not return duplicates, but it is wrong.

P3. h(k)

0%9	0	
1%9	1	10→19→28
2%9	2	20
3%9	3	12
4%9	4	
5%9	5	5
6%9	6	15, 33
7%9	7	
8%9	8	17

11-2-3

successful and unsuccessful search $\Theta(1+ \lg(\alpha))$

insertion and deletion: $\Theta(1+\alpha)$ insert/delete is linear

Problem 4. Recall that in a binary tree each node has at most 2 children. Suppose T is a binary tree with $|T| = n$ (i.e. has n nodes). For every node v we use L_v and R_v to denote the left subtree and right subtree of v, respectively. We say the subtree rooted at v is *nearly balanced* if $\frac{1}{2} \le |R_v|/|L_v| \le 2$, i.e. the sizes of the two subtrees are within a factor two of each other.

a. Suppose that the *root* of T is nearly balanced. What is the maximum height of T in terms of n? Explain why.

b. We say the whole tree is *nearly balanced* if for every node v of T, the subtree rooted at v is nearly balanced. Denote h as the height of T. Show that if T is nearly balanced then $h \in O(\log n)$ by first showing that $h \le \lceil \log_{\frac{3}{2}} n \rceil$.

a. the number of nodes in the right subtree could have at most 2 times the nodes of the left subtree. Assume $|Lv| = k$, then $|Rv| = 2k$, since whole tree contains n nodes, $k + 2k + 1 = n$. so in the worst case

$3k + 1 = n$ and thus $|Rv| = \frac{2(n-1)}{3}$. the height equals to

$3k = n-1$

$|Lv| = k = \frac{n-1}{3}$

the number of nodes -1. Since the largest subtree have $\frac{2(n-1)}{3}$ nodes, the height of the whole tree is $\frac{2(n-1)}{3} + 1 - 1 = \frac{2(n-1)}{3}$

b. In right subtree: $\frac{2(n-1)}{3}$

level 2: $\left(\frac{2}{3}\right)\left(\left(\frac{2}{3}\right)(n-1) - 1\right)$

$= \left(\frac{2}{3}\right)^2 (n-1) - \frac{2}{3}$

$H_{max}(n) = 1 + H_{max}\left(\frac{2}{3}(n-1)\right)$

$= 1 + 1 + \left(\left(\frac{2}{3}\right)^2 (n-1) - \frac{2}{3}\right)$

$= 1 + 1 + 1 + H_{max}\left(\left(\frac{2}{3}\right)^3 (n-1) - \left(\frac{2}{3}\right)^2 - \left(\frac{2}{3}\right)\right)$

$= 1 + 1 + 1 + \cdots + H_{max}\left(\frac{2}{3}^k (n-1) - \frac{2}{3}^{k-1} - \frac{2}{3}^{k-2} - \cdots - \frac{2}{3}\right)$

$= k + H_{max}\left(\frac{2}{3}^k (n-1) - \frac{1-\left(\frac{2}{3}\right)^k}{1-\frac{2}{3}} + 1\right)$

$= k + H_{max}\left(\left(\frac{2}{3}\right)^k (n+2) - 2\right)$

since # of holes equals $k+1$

$\therefore (\frac{2}{3})^k (n+2) - 2 = 1$

$$(\frac{2}{3})^k = \frac{n+2}{3}$$

$$k = \log_{\frac{3}{2}} \frac{n+2}{3} \le \log_{\frac{3}{2}} n$$

Problem 5. We have a set J of n jars and a set L of n lids such that each lid in L has a unique matching jar in J. Unfortunately all the jars look the same (and all the lids look the same). We can only try fitting a lid ℓ to a jar j for any choice of $\ell \in L$ and $j \in J$; the outcome of such an experiment is either a) the lid is smaller, b) the lid is larger, or c) the lid fits the jar. We cannot compare two lids or two jars directly. Describe an algorithm to match all the lids and jars. The expected number of tries (experiments) of fitting a lid to a jar performed by your algorithm must be $O(n \log n)$.

if $n=1$ (only one jar and only one lid) then they must be matching. return this as the sol.

if $n>1$

a. take one of the lid, say $L(1)$ as the pivot

b. compare the lid against all other jars to partition the jar as three group: 1) J_s as the jars that are smaller than $L(1)$ 2) J_l as the jars larger than $L(1)$ 3). the 3rd set containing the unique j. that fits $L(1)$

c.) use jar J_0 which is matching lid l to compare against all the lids in L to partion L into a sets of smaller, as L_s. and sets of larger as L_l, takes $n-1$ comparisons.

d) J_s are now the jars smaller than l and L_s are now the lids smaller than j. therefore the lids of jars in J_s are exactly in L_s; similarly the lids of jar in L_l are exactly in J_l. solve the two subproblem recursively

Problem 6.

a. Describe an efficient greedy algorithm for making change for a specified value using a minimum number of coins, assuming there are four denominations of coins (called quarters, dimes, nickels, and pennies), with values 25, 10, 5, and 1, respectively. Argue why your algorithm is correct.

b. Give an example set of denominations of coins so that a greedy change making algorithm will not use the minimum number of coins.

a. if $n < 5$: use n pennies

if $5 \le n < 10$: use 1 nickles and $n-5$ pennies

if $10 \le n < 25$: use $\lfloor \frac{n}{10} \rfloor$ dimms, and then 1 of 2 previous cases for $n - 10\lfloor \frac{n}{10} \rfloor$

if $n \ge 25$, use $\lfloor \frac{n}{25} \rfloor$ quarters, and then 1 of 3 previous cases for $n - 25\lfloor \frac{n}{25} \rfloor$

let S_k be the optimal solution when there are k types of changes.

when $k = 3$, we have dimes, nickels and pennies for changes

To obtain $k+1$ changes, a greedy choice of using largest change first will yield optimal solution $k+1$. This is because for all $k+1$ types of changes, they have greatest common divisor of 5. in other words, an optimal sol. can always be obtained by changing small change to large change.

b. when denominators are 0.25, 0.24, 0.01, such algorithm for making 0.48 would give 1 quarter and 23 pennis.

Problem 7. In the art gallery guarding problem, a line L represents a long art gallery hallway. We are given a set of location points on it $X = \{x_0, x_1, \ldots, x_{n-1}\}$ of real numbers that specify the positions of the paintings along the hallway. A single guard can guard paintings standing at most 1 meter from each painting on either side. Propose an algorithm that finds the optimal number of guards to guard all the paintings along the hallway.

m : track # of guards using

y_i : position of guards i

set $m \leftarrow 1$, $y_i \leftarrow x_0 + 1$

for $i = 1$ to $n-1$

  if $|y_m - x_i| > 1$ // Guards m can not guard painting at X

    $m \leftarrow m+1$ // add new guard

    $y_m = x_i + 1$ // one unit of the right of painting at x_i

  end if

end for

  return $(m, y_1, \cdots, y_m)$

Problem 8. A native Australian named Oomaca wishes to cross a desert carrying only a single water bottle. He has a map that marks all the watering holes along the way. Assuming he can walk k miles on one bottle of water, design an efficient algorithm for determining where Oomaca should refill his bottle in order to make as few stops possible. Argue why your algorithm is correct.

assume there are n holes along the way

located $h(1), h(2), \ldots, h(n)$, x is the end point with full water at $h(0)$

Find-hole(h, k)

current $= 0$

while current $+ k < x$ do

  let i be the largest index such that $h(i) \leq$ current $+ k$

  current $\leftarrow h[i]$