BLG335E Final Exam Fall 2016

Name and Student ID:____________________________

Analysis of Algorithms I
JANUARY 4, 2017 15:00
(2 Hours)

1 (20 pt)	2 (15 pt)	3 (15 pt)	4 (15 pt)	5 (20 pt)	6 (15 pt)	Total(100 pt)

On my honor, I declare that I neither give nor receive any unauthorized help on this exam.

Student Signature:____________________

1) (20 pts) Randomized Analysis / Indicator Random Variables

((a), (b), True or False, and Justify: If the statement is correct, briefly state why. If the statement is wrong, explain why. **NO credits without justification**)

a) (5 pts) An adversary can provide randomized quicksort with an input array of length *n* that forces the algorithm to run in w(nlgn) time on that input.

Solution: **False.** As we saw in lecture, for *any* input, the expected running time of quicksort is $O(n \lg n)$, where the expectation is taken over the random choices made by quicksort, independent of the choice of the input.

b) (5 pts) Let X be an indicator random variable such that E[X] = 1/2. Then, we have

E[sqrt(X)] = 1/sqrt(2).

Solution: **False.** Since X is an indicator random variable, X = 0 or X = 1. For both possible values $\overline{X} = X$, which implies that $E\left[\overline{X}\right] = E[X] = 1/2$.

c) (10 pts) Imagine you are given a bag of *n* balls. You are told that 10% of the balls are blue and 90% of the balls are red. How many balls do you have to draw from the bag to see a blue ball with probability at least 2/3? (You can assume that the balls are drawn with replacement.)

Solution: Since the question only asked the asymptotic number of balls drawn, $\Theta(1)$ (plus some justification) is a sufficient answer. Below we present a more complete answer.

Assume you draw k balls from the bag (replacing each ball after examining it).

Lemma 3 *For some constant k sufficiently large, at least one ball is blue with probability 2/3.*

Proof. Define indicator random variables as follows:

$$X_i = \begin{cases} 1 & \text{if ball i is blue} \\ 0 & \text{if ball i is red} \end{cases}$$

Notice, then, that $\Pr X_i = 1 = 1/10$ and $\Pr X_i = 0 = 9/10$. We then calculate the probability that at least one ball is blue:

$$\begin{aligned} &= 1 - \prod_{i=1}^{k} \Pr(X_i = 0) \\ &= 1 - \left(\frac{9}{10}\right)^k \\ &\geq \frac{2}{3}. \end{aligned}$$

Therefore, if $k = \lg(1/3)/\lg 0.9$, the probability of drawing at least one blue ball is at least $2/3$.

2) (15 pts) Amortized Analysis

a) (3 pts) List the amortized analysis methods you have learned at the lecture.

- the ***aggregate*** method,
- the ***accounting*** method,
- the ***potential*** method.

b) (12 pts) A sequence of n operations is performed on a data structure. The ith operation costs i if i is an exact power of 2, and 1 otherwise. Use **accounting method** to determine the amortized cost per operation. **(NO credits, if you use another method)**

Let c_i = cost of ith operation.

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2}, \\ 1 & \text{otherwise}. \end{cases}$$

Charge each operation \$3 (amortized cost $\hat{c}_i$).

- If i is not an exact power of 2, pay \$1, and store \$2 as credit.
- If i is an exact power of 2, pay \$$i$, using stored credit.

Operation	Cost	Actual cost	Credit remaining
1	3	1	2
2	3	2	3
3	3	1	5
4	3	4	4
5	3	1	6
6	3	1	8
7	3	1	10
8	3	8	5
9	3	1	7
10	3	1	9
⋮	⋮	⋮	⋮

Since the amortized cost is \$3 per operation, $\sum_{i=1}^{n} \hat{c}_i = 3n$.

We know from Exercise 17.1-3 that $\sum_{i=1}^{n} c_i < 3n$.

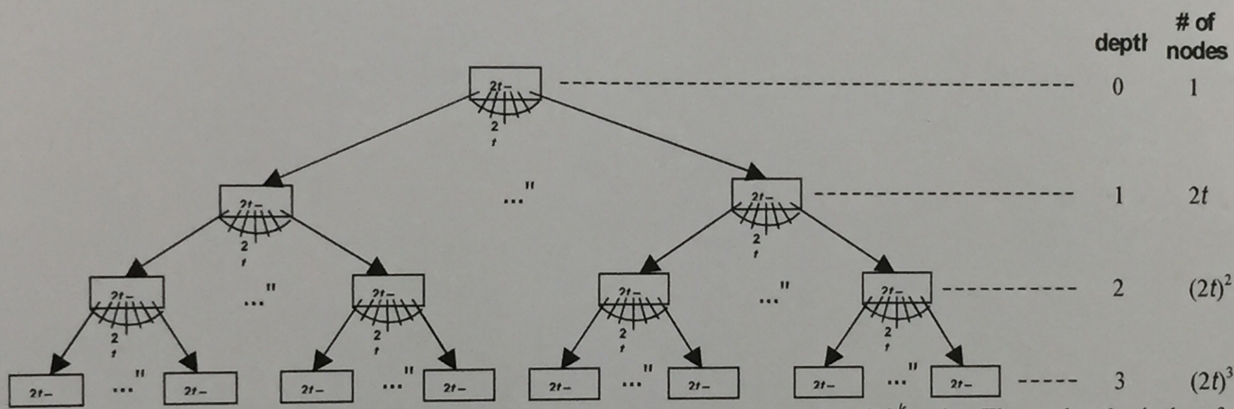
Then we have $\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i \Rightarrow$ credit = amortized cost − actual cost ≥ 0.

Since the amortized cost of each operation is $O(1)$, and the amount of credit never goes negative, the total cost of n operations is $O(n)$.

3) (15 pts) B-Trees

What is the maximum number of keys that can be stored in a B-tree of degree t = 5 and height = 3.

A B-tree with maximum number of keys is shown below:



We can see that each node contains $2t-1$ keys, and at depth k, the tree at most has $(2t)^k$ nodes. The total nodes is therefore the sum of $(2t)^0$, $(2t)^1$, $(2t)^2$, $(2t)^3$, …, $(2t)^h$. Let MaxKeyNum(t, h) as a function that returns the maximum number of keys in a B-tree of height h and the minimum degree t. We can get that:

$$\text{MaxKeyNum}(t, h) = (2t-1)[(2t)^0 + (2t)^1 + (2t)^2 + (2t)^3 + \ldots + (2t)^h]$$ as [keys per node] * [total # of nodes]

$$= (2t-1)\sum_{i=0}^{h}(2t)^i$$ by using Sigma to sum up total # of nodes

$$= (2t-1)\frac{(2t)^{h+1}-1}{2t-1}$$ by the summation formula of geometric series

$$= (2t)^{h+1}-1$$

4) (15 pts) Hashing

Suppose n keys are chosen randomly and the hash function distributes the keys uniformly at random over {0, 1, ..., m-1}.

a. (5 pts) What is the probability that n=2 independently chosen keys have the same hash value i.e. what is the probability there is a collision among those two keys?

$\boxed{1/m}$

b. (10 pts) What is the probability that there are no collisions among any of the n keys?

Assume n <= m.

$$\frac{(m-1)(m-2)\dots(m-n+1)}{m^{(n-1)}}$$

$\Rightarrow$ Prob. NO collision btw k_1 & k_2 = $\frac{m-1}{m}$

" NO collision btw k3 & others = $\frac{(m-2)(m-1)}{m^2}$

⋮

5) Red-Black Trees (20 pts) (ignore NIL nodes)

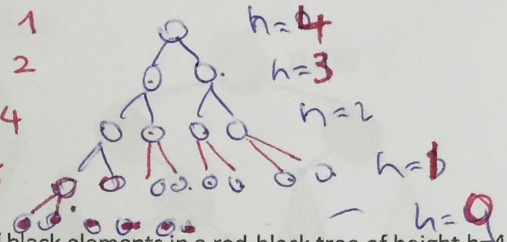
a. (5 pts) What is the maximum number of black elements in a red-black tree of height 4? Explain why.

(Full ~~Partial~~ points for ~~those~~ h=4 including NIL nodes).

Case1 31 all nodes=black

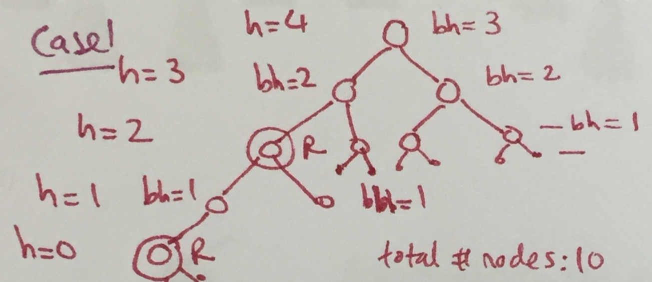
15 (Case2)

1
2
4
8
16



b. (5 pts) What is the minimum number of black elements in a red-black tree of height h=4? Explain why.

Case1

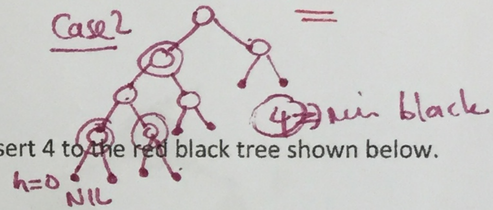


total # nodes: 10

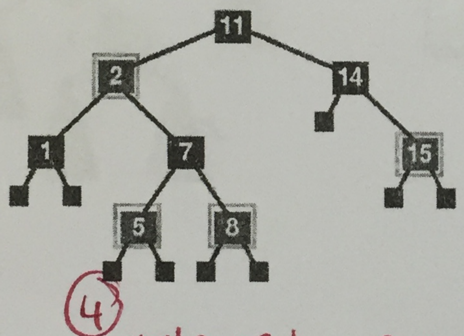
max # red elements on a path = 2

black elements: 8

Case2

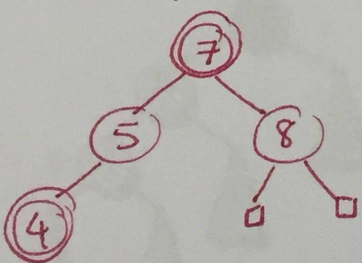


c. (10 pts) Nodes 2, 5, 8 and 15 are RED in the tree below. Insert 4 to the red black tree shown below. Show all your steps at the back of this page.

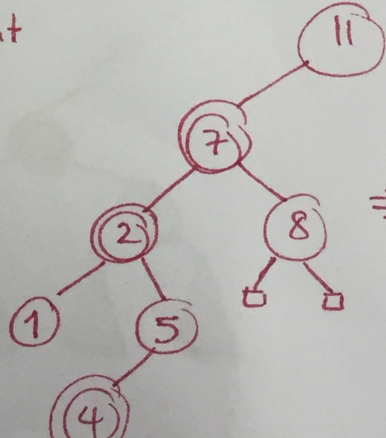


IF: only solution (final tree) is correct → pts. PARTIAL

uncle - red
parent - red ⇒ Push blackness down from grandparent



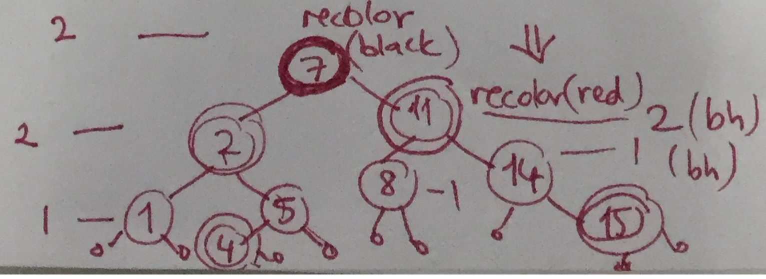
⇒ 2 is red too.
7 is right child.
2 is left child
Rotate left



⇒ 7 and 2 still red

7 is left child,
2 is left child
Rotate Right

⇓



6) (15 pts) Augmenting data structures

a. (5 pts) Write four steps that you follow while augmenting data structures.

b. (10 pts) Can we maintain the black-heights of nodes in a red-black tree as attributes in the nodes of the tree? Show how, or argue why not.

Hint: Use all three cases (Cases 1,2 and 3) performed during the Insertion operation of an RB-Tree, and check if you can maintain the black-heights in the nodes without changing the complexity of the operation.

a) 1. Choose an underlying data structure
2. Determine additional info to maintain underlying data structure
3. Verify that we can maintain additional info for the basic modifying operations on the data str.
4. Develop new operations.

b) Yes. Black-height is based on info at the node and its children. Actually it's based on a child's info.
black-height of a node : bh of its red child OR
bh of its black child + 1
→ Show for three cases in Insert.
↓
Partial for true answer, but wrong argument.

only Yes → 3

Yes, but some cases are wrong → 8/+5