BLG 381E ADVANCED DATA STRUCTURES
FINAL EXAM – JANUARY 9, 2012 12:00-2:00 PM (PART I)

1 (20pt)	2 (10 pt)	3 (10 pt)	4 (10 pt)	5 (15 pt)	6 (15 pt)	7 (20 pt)	Total (100 pt)

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

Student Signature:_____

Duration: 120 minutes. *Write your name on each sheet.Write your answers neatly in the space provided for them. You must show all your work for credit. Books and notes are closed. No questions are allowed during the exam. Good Luck!*

Q1) [20 pts] Red-black trees

a) [4 pts]What are the four properties of red-black trees (RBT)?

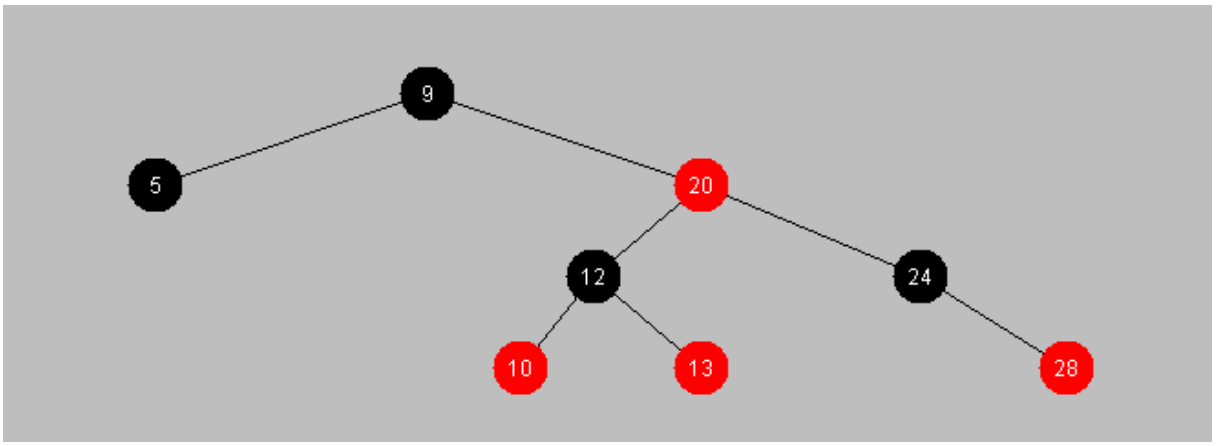
1. Every node is either red or black.
2. The root and leaves (NIL's) are black.
3. If a node is red, then its parent is black.
4. All simple paths from any node x to a descendant leaf have the same number of black nodes = black-height(x).

b) [2 pts] What are the two modifying operations in RBT to update the tree after insert or delete operation?

The operations INSERT and DELETE cause modifications to the red-black tree:

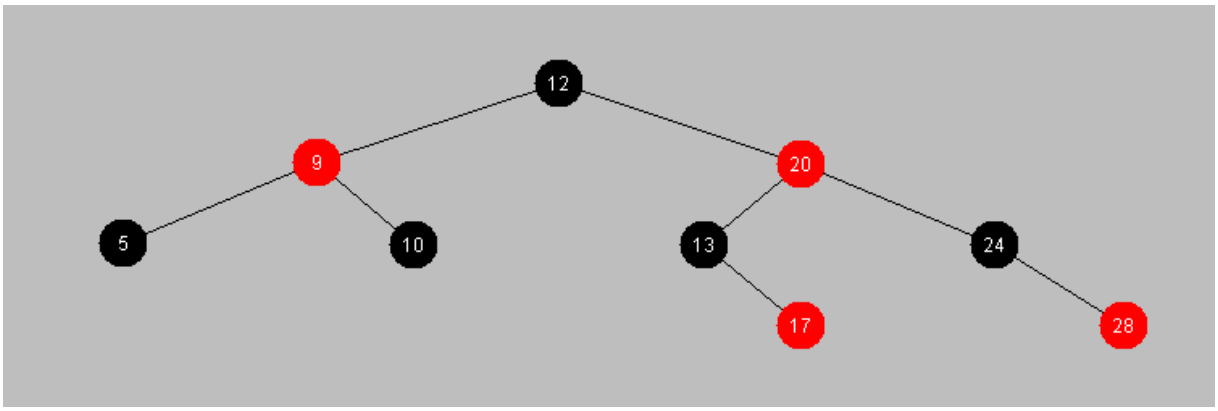
- the operation itself,
- color changes,
- restructuring the links of the tree via "rotations".

c) [6pts] Build a RBT by inserting the following numbers in the given order: [9,5,20,12,24,10,13,28]. (Label the node as R(red) or B(black) to represent the color)



Q1) Continued

Q1d) [8 pts] Insert 17 to the tree which you build in **(b)**. Explain clearly the processing steps.



Q2) [10 pts] Augmenting data structures

a) [2 pts]What is the purpose of augmenting a data structure?

By augmenting already existing data structures one can build new data structures. The purpose of augmenting data structures is fairly simple. It is used to build new data structure that help us quickly get some type of information by augmenting already existing data structures.

b) [4 pts]What are the four steps of augmenting data structure?

1. choosing an underlying data structure,
2. determining additional information to be maintained in the underlying data structure,
3. verifying that the additional information can be maintained for the basic modifying operations on the underlying data structure, and
4. developing new operations.

c) [4 pts]Tell briefly how you would augment a data structure to perform fast order-statistic operations following the steps in **(b)**.

For example:

1. Choose red black trees.
2. Determine sub-tree size as an additional information to be stored in the data structure.
3. Verify that sub-tree size can be maintained for the RB-INSERT, RB-DELETE and rotations.
4. Develop OS-SELECT and OS-RANK operations.

Q3) [10 pts] B-trees

a) [2 pts]Why do we need/use B-trees?

B-tree is a tree data structure that keeps data sorted and allows searches, sequential access, insertions, and deletions in logarithmic time. The B-tree is a generalization of a binary search tree in that a node can have more than two children. B-trees are balanced search trees designed to work well on magnetic disks or other direct-access secondary storage devices, minimizing disk I/O operations.

b) [5 pts]What are the properties of a B-tree?
(please use the following notation: x is a node, $n[x]$, the number of keys currently stored in node x, $key_i[x]$, ith key of node x, $leaf[x]$, leaf of node x, $c_i[x]$,ith children of node x)

A ***B-tree*** T is a rooted tree (whose root is $root[T]$) having the following properties:

1. Every node x has the following fields:
a. $n[x]$, the number of keys currently stored in node x,
b. the $n[x]$ keys themselves, stored in nondecreasing order, so that $key_1[x] \leq key_2[x] \leq \cdots \leq key_{n[x]}[x]$,
c. $leaf[x]$, a boolean value that is TRUE if x is a leaf and FALSE if x is an internal node.

2. Each internal node x also contains n[x]+1 pointers c1[x], c2[x], . . . , cn[x]+1[x] to its children. Leaf nodes have no children, so their ci fields are undefined.

3. The keys keyi [x] separate the ranges of keys stored in each subtree: if ki is any key stored in the subtree with root ci [x], then
$$k1 \leq key1[x] \leq k2 \leq key2[x] \leq \cdots \leq keyn[x][x] \leq kn[x]+1 .$$
4.All leaves have the same depth, which is the tree's height h.
5.There are lower and upper bounds on the number of keys a node can contain. These bounds can be expressed in terms of a fixed integer $t \geq 2$ called the minimum degree of the B-tree:
a. Every node other than the root must have at least $t - 1$ keys. Every internal node other than the root thus has at least t children. If the tree is nonempty, the root must have at least one key.
b. Every node can contain at most $2t - 1$ keys. Therefore, an internal node can have at most 2t children. We say that a node is full if it contains exactly $2t - 1$ keys.1

c) [3 pts]How do you insert a key into a full node?

we insert the new key into an existing leaf node. Since we cannot insert a key into a leaf node that is full, we introduce an operation that ***splits*** a full node y (having $2t - 1$ keys) around its ***median key*** $key_t[y]$ into two nodes having $t - 1$ keys each. The median key moves up into y's parent to identify the dividing point between the two new trees. But if y's parent is also full, it must be split before the new key can be inserted, and thus this need to split full nodes can propagate all the way up the tree.

Q4) [10 pts] The Hiring Problem: Suppose that you need to hire a new office assistant. The employment agency will send you one candidate each day. You will interview that person and then decide to either hire that person or not. You must pay the employment agency a small fee to interview an applicant. To actually hire an applicant is more costly, however, since you must fire your current office assistant and pay a large hiring fee to the employment agency. You are committed to having, at all times, the best possible person for the job. Therefore, you decide that, after interviewing each applicant, if that applicant is better qualified than the current office assistant, you will fire the current office assistant and hire the new applicant. You are willing to pay the resulting price of this strategy, but you wish to estimate what that price will be.
It assumes that the candidates for the office assistant job are numbered 1 through *n*. Interviewing has a low cost, say *ci* , whereas hiring is expensive, costing *ch*.

a) [2 pts]What is the worst-case hiring cost?

In the worst case, we actually hire every candidate that we interview. This situation occurs if the candidates come in increasing order of quality, in which case we hire n times, for a total hiring cost of $O(nc_h)$.

b) [8 pts]What is the expected number of times of hiring a new office assistant?
Hint: Use indicator random variables
we let X_i be the indicator random variable associated with the event in which the ith candidate is hired. Thus,

$$X_i = I\{\text{candidate i is hired}\} = \begin{cases} 1 \text{ if candidate } i \text{ is hired}, \\ 0 \text{ if candidate } i \text{ is not hired}, \end{cases}$$

and

$$X = X_1 + X_2 + \cdots + X_n .$$

Any one of these first i candidates is equally likely to be the best-qualified so far. Candidate i has a probability of $1/i$ of being better qualified than candidates 1 through $i - 1$ and thus a probability of $1/i$ of being hired. Thus, $E[X_i] = 1/i$.
Now we can compute $E[X]$:

$$E[X] = E$$

$$E[X] = E\left[\sum_{i=1}^{n} X\right],$$

$$= \sum_{i=1}^{n} E[X_i]$$

$$= \sum_{i=1}^{n} 1/i$$

$$= \ln n + O(1)$$

Assuming that the candidates are presented in a random order, algorithm HIREASSISTANT has a total hiring cost of $O(c_h \ln n)$.

THIS PAGE LEFT BLANK INTENTIONALLY, USE ONLY FOR Q1-Q4

BLG 381E Final Fall 2011 Name: _____

ClassRoom:____ Student ID:_____

BLG 381E ADVANCED DATA STRUCTURES ANSWERS

FINAL EXAM – JANUARY 9, 2012 12:00-2:00 PM (PART II)

1 (20pt)	2 (10 pt)	3 (10 pt)	4 (10 pt)	5 (15 pt)	6 (15 pt)	7 (20 pt)	Total (100 pt)

Q5. [15 Points] Binomial and Fibonacci Heaps

Q5a) [5pts]

What are the main advantages of Binomial and Fibonacci Heaps compared to the ordinary Heap? How do Binomial and Fibonacci Heaps achieve that advantage?

Binomial and Fibonacci heaps have much faster UNION operations than ~~regular~~ ordinary heaps.
They achieve that advantage by means of "lazy" operations, structural differences (a linked list of trees, for example)

State one difference between Binomial and Fibonacci Heaps

---in terms their complexity of operations.

Any one of is accepted

Union: Fibonacci $O(1)$
Binomial $O(\log n)$

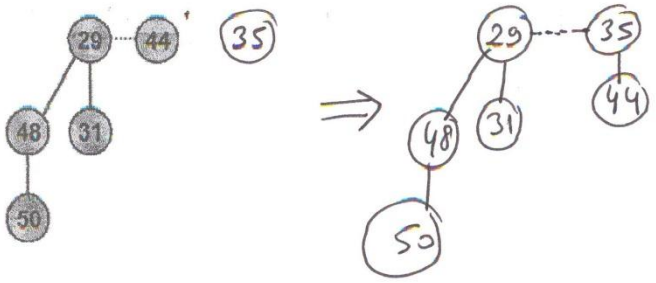
---in terms of their data structure properties.

any one of these is accepted

Fib: doubly linked list of roots, min pointer, marked nodes

Bin: singly linked list, no min pointer, Binomial trees.

Q5c) [10pts] Consider the following min-heap ordered Binomial Heap. Show (graphically) how you would insert an item with key 35 into this Binomial Heap.



$$\begin{array}{r} (101)_2 \\ +\ (\ 1)_2 \\ \hline (110)_2 \end{array}$$

Q6)15pts [Amortized Analysis]
An array *A[0..k−1]* of bits (each array element is 0 or 1) stores a binary number $x = \sum_{i=0}^{k-1} A[i]2^i$. To add 1 (modulo 2^k) to x, we use the following procedure:

```
INCREMENT(A,k)
1.  i ← 0
2.  while i<k and A[i]=1 do
3.      A[i] ← 0
4.      i ← i+1
5.  if i<k then
6.      A[i] ← 1
```

Given a number x, define the potential Φ(x) of x to be the number of 1's in the binary representation of x. For example, Φ(19) = 3, because 19 = 100112. Use a potential-function argument to prove that the amortized cost of an increment is O(1), where the initial value in the counter is x = 0.

Hint:
If cost of operation i is c_i, Amortized cost $\hat{c}_i$ with respect to Φ is: $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$

Solution Q6

$\Phi(x)$ is a valid potential function the number of 1's in the binary representation of x is always nonnegative, i.e., $\Phi(x) \geq 0$ for all x. Since the initial value of the counter is 0, $\Phi_0 = 0$.

Let c_k be the real cost of the operation INCREMENT(A, k), and let d_k be the number of times the while loop body in Lines 3 and 4 execute (i.e., d_k is the number of consecutive 1's counting from the least-significant bit of the binary representation of x). If we assume that executing all of Lines 1, 2, 5, and 6 require unit cost, and executing the body of the while loop requires unit cost, then the real cost is $c_k = 1 + d_k$,

The potential decreases by one every time the while loop is executed. Therefore, the change in potential, $\Delta\Phi = \Phi_k - \Phi_{k-1}$ is at most $1 - d_k$. More specifically, $\Delta\Phi = 1 - d_k$ if we execute Line 6 , and $\Delta\Phi = -d_k$ if we do not.

Thus, using the formula for amortized cost, we get

$$\begin{aligned} \hat{c}_k &= c_k + \Delta\Phi \\ &= 1 + d_k + \Delta\Phi \\ &\leq 1 + d_k + 1 - d_k \\ &= 2. \end{aligned}$$

Therefore, the amortized cost of an increment is $O(1)$.

BLG 381E Final Fall 2011 Name: ANSWERS

ClassRoom:____ Student ID:_____

Q7) [20points] Hashing
Insert numbers B={6, 1, 13, 18} into an empty hash table of size 6. Clearly indicate your hash functions. Show the hash table for all insertions.

Note: You could also use, for example: $h(x,i) = ((x \bmod 11) + i) \bmod 6$

Q7a) [5 points] Use open addressing and linear probing. What is the number of collisions?

$h(x,i) = (x + i) \bmod 6$ (where $i = 0, 1, 2, \ldots$ probe sequence index)

insert: (6) $h(x,0)=0$ | (1) $h(x,0)=1$ | (13) $h(x,0)=1$ X, $h(x,1)=2$ | (18) $h(x,0)=0$ X, $h(x,1)=1$ X, $h(x,2)=2$ X, $h(x,3)=3$

#collisions: 0 | 0 | 1 | 3 collisions

HashTable

	0	1	2	3
0	6	6	6	6
1		1	1	1
2			13	13
3				18
4				
5				

Total # collisions = 1+3 = 4

Q7b) [7 points] Use open addressing and double hashing. What is the number of collisions?

$h(x,i) = (h_1(x) + i\,h_2(x)) \bmod 6$

$h_1(x) = x \bmod 5$
$h_1(x) = x \bmod 11$
} you could choose any other $h_1()$ $h_2()$

insert: (6) | 1 | 13 | 18

$h(x,i)$: $(0 + 0*6) \bmod 6 = 1$ | $(1+0\times1) \bmod 6 = 1$ X, $(1+1\times1) \bmod 6 = 2$ | $(3+0\times2) \bmod 6 = 3$ | $(3+0\times7) \bmod 6 = 3$ X, $(3+1\times7) \bmod 6 = 4$

#collisions: 0 | 1 | 0 | 1

	6	1	13	18
0		6	6	
1	6	1	1	6
2			13	1
3				13
4				18
5				

Q7b) [8pts] Suppose we use a hash function ***h()*** to hash n distinct keys into an array T of length $m=c*n$, where $c>2$ is an integer constant.
Assuming simple uniform hashing (i.e. the probability of element i hashing to any slot k is 1/m), show that the expected number of colliding pairs of elements is $\theta(n/c)$.

Solution Q7b

Let $X_{i,j}$ be an indicator random variable equal to 1 if elements i and j collide, and equal to 0 otherwise. Simple uniform hashing means that the probability of element i hashing to slot k is $1/m$. Therefore, the probability that i and j both hash to the same slot $\Pr(X_{i,j}) = 1/m$. Hence, $\mathrm{E}[X_{i,j}] = 1/m$. We now use linearity of expectation to sum over all possible pairs i and j:

$$\begin{aligned}
\mathrm{E}[\text{number of colliding pairs}] &= \mathrm{E}\left[\sum_{i=1}^{n}\sum_{j=i+1}^{n} X_{i,j}\right] \\
&= \sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[X_{i,j}] \\
&= \sum_{i=1}^{n}\sum_{j=i+1}^{n} 1/m \\
&= \frac{n(n+1)}{2m} \\
&= \Theta(n^2/m)
\end{aligned}$$

Since we know $m = c*n \Rightarrow = \Theta(n^2/cn) = \Theta(n/c)$ ✓