Homework Assignment #8
Due: November 22, 2024 at 5:00 p.m.

1. Consider a dyamic dictionary implemented by a B-tree, which allows searches, insertions and deletions. Now we would like to make the implementation *persistent*. This means that a new version of the data structure is created each time we update the dictionary by doing an insertion or deletion. Furthermore, we are allowed to do a search in any old version of the tree. Thus, we have 3 operations:

- PERSISTENT-INSERT(element): inserts the element into the most recent version of the dictionary. Returns a version number of the resulting version of the dictionary.
- PERSISTENT-DELETE(element): deletes the element from the most recent version of the dictionary. Returns a version number of the resulting version of the dictionary.
- PERSISTENT-SEARCH(key, version): searches for the given key in the given version of the dictionary (not necessarily the most recently created version).

For example, if we start with an empty dictionary, we could do the following sequence of operations:
PERSISTENT-INSERT(A) $\rightarrow$ returns 1
PERSISTENT-INSERT(B) $\rightarrow$ returns 2
PERSISTENT-SEARCH($B, 1$) $\rightarrow$ returns "not found"
PERSISTENT-SEARCH($B, 2$) $\rightarrow$ returns the element associated with B
PERSISTENT-DELETE(A) $\rightarrow$ returns 3
PERSISTENT-SEARCH($A, 3$) $\rightarrow$ returns "not found"
PERSISTENT-SEARCH($A, 1$) $\rightarrow$ returns the element associated with A.

We could copy the entire B-tree every time we do an update, and keep all the old versions in memory, but this uses a lot of space. Instead, we want to just make new copies of the nodes that change. (See Problem 13-1 in the textbook for a similar description of a persistent red-black tree.) For the purposes of this question, we'll just worry about insertions, not deletions.

[2] **(a)** Describe which nodes in a B-tree could change when we insert a key k. How many nodes does an insert have to write to memory, at most?

[4] **(b)** Describe how to modify the algorithm of Section 18.2 to create PERSISTENT-INSERT.

[2] **(c)** How much time and space is needed for each insertion in the worst case?

(d) [3] **(Bonus)** Is it possible that the worst-case space required for a sequence of n insertions is $O(n)$? (In other words, could the amortized space requirement per insertion be $O(1)$ even if the worst-case space requirement for an individual insertion is bigger?) Explain why or why not.