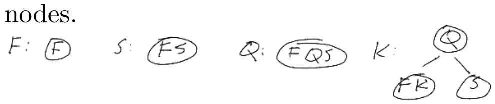
Homework 12
CS 3385

1. For what values of t is the tree of Figure 18.1 a legal B-tree?

2. Show all B-trees of minimum degree 2 (i.e. $t = 2$) that represent $1, 2, 3, 4, 5$.

3. Show the results of inserting the keys $F, S, Q, K, C, L, H, T, V, W, A$ in order into an empty B-tree with minimum degree 2 (i.e. $t = 2$). The first four steps are given. NOTE: Pay particular attention to the point made in the last 70 seconds of the B-tree operations node about splitting on all encountered full nodes.



4. Show that if a `decrement()` operation were included in the k-bit counter example, n operations (either increment or decrement) could cost as much as $\Theta(nk)$ time.

5. Suppose we perform a sequence of n operations on a data structure in which the ith operation costs i if i is an exact power of 2, and 1 otherwise. Determine the amortized cost per operation using the aggregate analysis methods.

6. Dynamic array classes work as follows: the class stores a raw array initialized to some size n and also maintains a counter i for how many elements have been added to the array. Once n elements have been added to the array, on the next `add()` call, a new array of size $2n$ is allocated, n items are copied, and then the new item is added to the new array. As items are added, the raw array continues to double in size when necessary.

   All major C-based languages support a dynamic array, listed here along with part of their online documentation:

C++	`vector<T>`	"Insertion or removal of elements at the end - **amortized constant** $O(1)$" [1]
Java	`ArrayList<E>`	"The add operation runs in **amortized constant time**" [2]
C#	`List<T>`	"If Count is less than Capacity, this method is an O(1) operation. If the capacity needs to be increased to accommodate the new element, this method becomes an O(n) operation, where n is Count." [3] (Apparently Microsoft doesn't give an amortized analysis.)

   Using the result of problem #5, show that the `add()` operation for such a dynamic array really does run in amortized constant time. Assume, for simplicity, that you do not have a `remove()` function.

7. Show the Fibonacci heap that results from calling `FIB-HEAP-EXTRACT-MIN` on the Fibonacci heap shown in Figure 19.4(m).

[1] http://en.cppreference.com/w/cpp/container/vector
[2] http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html
[3] https://msdn.microsoft.com/en-us/library/3wcytfd1.aspx