1. We want to select $t$ so that a full node uses as much of a disk block as possible. In a full node there are $2t - 1$ keys (4 bytes each), $2t - 1$ data record references (8 bytes each), $2t$ child pointers (4 bytes each), a parent pointer (4 bytes), the number of keys (4 bytes) and the leaf bit (which we’ll go ahead and assume takes 4 bytes though 1 bit would do). Hence we want to pick $t$ as large as we can so that $12(2t - 1) + 4(2t) + 12 = 32t \leq 4096$. Solving for $t$ yields that we need $t = 128$. In class, we argued that the number of disk pages that must be read ($d-1$ using the notation from class) is at most $\log_t(n + 1)/2$. Since $\log_{128}(n + 1)/2 = \log_{128} \approx 2.7$ and the number of levels below the root must be an integer, at most 2 disk pages will need to be brought into main memory during a search.

2. (i): Not legal since the height is not balanced. More specifically, both the node with “BD” and “KS” are at the same level but “BD” is a leaf and “KS” is not.
   
   (ii): This is legal. Remember, that the root can have just a single key.
   
   (iii): Not legal – the key “D” has less than the minimum allowable size of 2 keys.
   
   (iv): This is legal.
   
   (v): Not legal – there’s no leaf node corresponding to the keys between G and L.

3. B-tree insertion problem
4. B-tree deletion problem

- Deleting A:
  - Apply case 3b

- Deleting V:
  - Go to PT then apply case 3b and case 1

- Deleting P:
  - Apply case 3a

  - Apply case 2b

Diagram:

- Initial tree
  - Deleting A
  - Deleting V
  - Deleting P