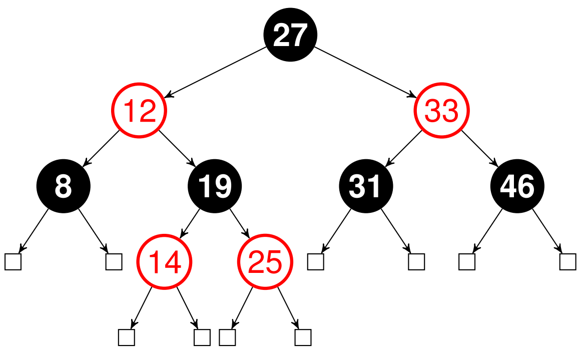
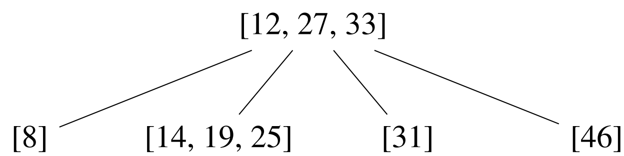
CS 61B Discussion 11: Review Fall 2015

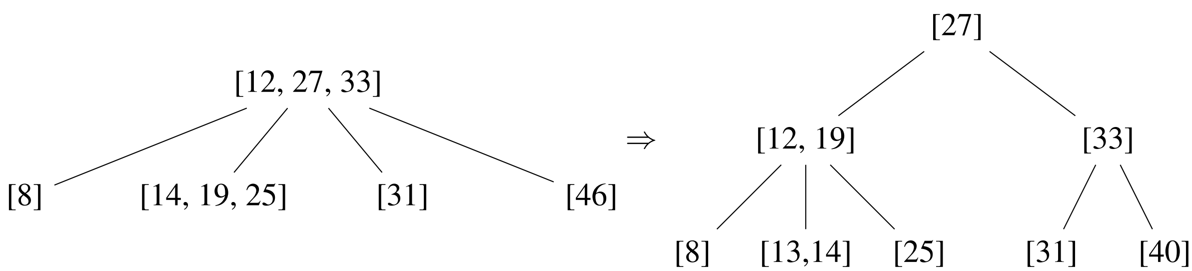
1 Balanced Search Trees

(a) Convert the red-black tree into a 2-4 tree.

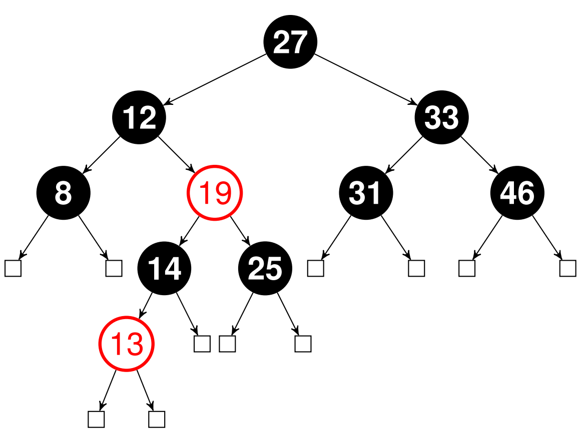




(b) Insert 13 into the 2-4 tree.

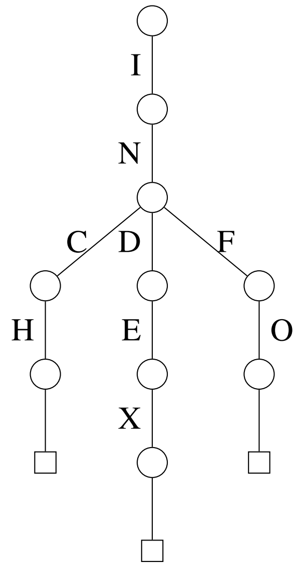


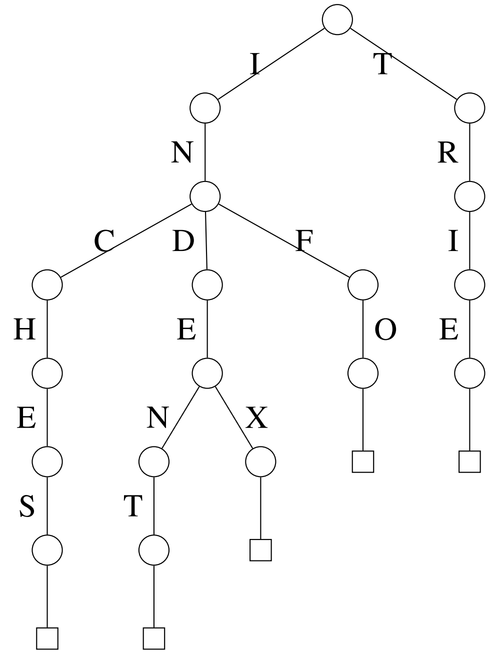
(c) Convert the resulting 2-4 tree into a valid red-black tree.



2 Tries

First, list the words encoded by the trie. Then draw the trie after inserting the words *indent*, *inches*, and *trie*. Ignore case.





Encoded words: index, info, inch

3 Runtime Analysis

(a) Give the best and worst case runtimes for method A in $\Theta(\cdot)$.

```
public boolean A(int[] arr, int x) {
    //Assume arr is sorted; N is arr.length
    return A(arr, x, 0 , arr.length-1);
}

public boolean A(int[] arr, int x, int low, int high) {
    if (low > high) return false;
    int mid = (low + high) / 2;
    if (arr[mid] == x) return true;
    return A(arr, x, low, mid-1) || A(arr, x, mid+1, high);
}
```

This is almost binary search, except that both halves are recursed on.
Best case: $\Theta(1)$. Worst case: $\Theta(N)$.

(b) Give the best and worst case runtimes for method B in $\Theta(\cdot)$.

```
public boolean B(int[] arr) {
    //N is arr.length
    int count = arr.length - 1;
    while(count > 0) {
        count = count - arr.length / 50;
    }
    return count;
}
```

No matter how big the input array is, the loop will only execute about 50 times.
Best case: $\Theta(1)$. Worst case: $\Theta(1)$.