**WEEK - 6**

**R-10.1**

def pop (self, last=True):

if not self:

raise KeyError("dictionary is empty")

key = next(reversed(self) if last else iteration(self))

return key, MutableMapping.pop(self, key)

**R-10.4**

O(n2)

**R-10.9**

0->

1->20

2->

3->

4 -> 16,5

5->44,88,11

6->94,39

7->12,13

8->

9->13

**C-10.42**

SUM = 0

j = 0

for i from n-1 to 0 step -1 do {

while (j<n)and(A[i,j]==1) do {

j = j+1

} SUM = SUM+j

}

**R-11.2**

30

40

30

20 40

58

30

24 40

26 58

48

30

24 40

11 26 58

13 48

**R-11.3**

5 (2 w/ 1 as root, 1 w/ 2 as root, 2 w/ 3 as root).

**R-11.29**

Quicksort (the one where you select the pivot randomly/just pick one fixed, making worst case Omega(n^2)) might be better than Red-Black Trees because (not necessarily in order of importance)

* Quicksort is in-place. The keeps your memory footprint low. Say this quicksort routine was part of a program which deals with a lot of data. If you kept using large amounts of memory, your OS could start swapping your process memory and trash your perf.
* Quicksort memory accesses are localized. This plays well with the caching/swapping.
* Quicksort can be easily parallelized (probably more relevant these days).
* If you were to try and optimize binary tree sorting (using binary tree without balancing) by using an array instead, you will end up doing something like Quicksort!
* Red-Black trees have memory overheads. You have to allocate nodes possibly multiple times, your memory requirements with trees is doubles/triple that using arrays.
* After sorting, say you wanted the 1045th (say) element, you will need to maintain order statistics in your tree (extra memory cost because of this) and you will have O(logn) access time!
* Red-black trees have overheads just to access the next element (pointer lookups)
* Red-black trees do not play well with the cache and the pointer accesses could induce more swapping.
* Rotation in red-black trees will increase the constant factor in the O(nlogn).
* Perhaps the most important reason (but not valid if you have lib etc available), Quicksort is very simple to understand and implement. Even a school kid can understand it!