4.2.9 HW3

maintain 3 indexes into array r, w, b

A [0 ... r-1, r ... w-1, w ... b, b+1 ... n-1]
r's w's unknown b's

Algorithm DutchFlag (A[0..n-1])
r ← 0, b ← n-1, w ← 0
while w ≤ b
    if A[w] = 'R'
        swap(A[r], A[w]);
        r ← r+1;
        w ← w+1
    else if A[w] = 'W'
        w ← w+1
    else // A[w] = 'B'
        swap(A[w], A[b]);
        b ← b-1

4.2.11 use quick sort like matching

1. Partition - randomly select a nut and compare w/ all screws partitioning screws into 2 partially sorted partition

2. select new nut as a new partition, compare w/ first matching screw and repeat above step 1 for relevant subpartition of n/2 screws (if nut smaller than found screw, resort subpartition on left & else resort subpartition on right)

3 repeat until all nuts are selected

Running time $C(n) = \frac{1}{n}\sum_{s=0}^{n-1} (2n-1) + C(s) + C(n-1-s) = \Theta(n \lg n)$

4.3.2 Solve

$C_{worst}(n) = C_{worst}(\lfloor n/2 \rfloor) + 1$ for $n > 1$ ∧ $C_{worst}(1) = 1$
for $n = 2^k$ (by backward substitution)

$C(2^k) = C(2^{k-1}) + 1$ for $k > 0$ $C(2^0) = 1$

sus.

$C(2^k) = C(2^{k-1}) + 1 =$

$= [C(2^{k-2}) + 1] + 1 = C(2^{k-2}) + 2$

$= [C(2^{k-3}) + 1] + 2 = C(2^{k-3}) + 3$

$\vdots$

$= C(2^{k-i}) + i$

$\vdots$

$= C(2^{k-k}) + k \Rightarrow C(2^k) = C(1) + k = 1 + k$

for $n = 2^k$ hence $k = \log_2 n \Rightarrow C(n) = \log_2 n + 1$

4.4.2 Count # of leaves in bin tree

Algo count (T)
if $T = \emptyset$ return 0
else if $T_L = \emptyset \wedge T_R = \emptyset$ return 1
else return count (T_L) + count (T_R)

4.6.1 solve closest pair problem by divide and conquer
/ input - sorted array A

Algol CP ($A[l,..r]$)
if $l = r$ return ∞
else if $r - l = 1$ return $A[r] - A[l]$
else if $r - l = 2$ return $\min\{A[r] - A[l+1], A[l+1] - A[l]\}$
else return $\min\{$ CP ($A[l \ldots \lfloor (l+r)/2 \rfloor]$),
CP ($A[\lfloor (l+r)/2 \rfloor + 1 \ldots r]$),
$A[\lfloor (l+r)/2 \rfloor + 1] - A[\lfloor (l+r)/2 \rfloor]\}$

$T(n) = 2T(n/2) + c$
$= \Theta(n \lg n)$