**Assignment 10**

**R-3.19**

**Algorithm** removeElement(e)

Input e element to remove

Output out element deleted or No\_such\_element

P 🡨 get the least node in the highest list

y 🡨 after(p)

While e  y.element ∧ y null do

While e> y do

y 🡨 after (y)

if e y then

y 🡨 down(left(y))

if e=y then

tmp 🡨 null

while down(y) null do

tmp 🡨 down(y)

removeNode(y)

return tmp

else

return No\_Such\_Element

**C-4.16**

**Algorithm** containsDuplicate(S)

Input sequence S contains a list of integers

Output Boolean indicating whether there is a duplicate integer in S

H 🡨 create new hashtable

Foreach v Є S do

existingElement 🡨 H.removeElement(v)

if existingElement No\_Such\_Key then

return true

else

H.insertItem(v,v)

Return false

**C-4.18**

**Algorithm** inPlacePartition(S, lo, hi)

Input Sequence S and ranks lo and hi, lo, hi

Output the pivot is now stored at its sorted rank

p 🡨 a random integer between lo and hi

S.swapElements(S.atRank( lo ), S.atRank( p ))

pivot 🡨 S.elemAtRank(lo)

for i🡨 1 to S.size() do

if pivot 🡨 s.elementAtRank(i) then

lo 🡨 lo +1

s.swapElements(S.atRank(lo), S.atRank(i))

j 🡨 lo + 1

k 🡨 hi

while j < k do

while k > j ∧ S.elemAtRank( k ) > pivot do

k 🡨 k – 1

while j < k ∧ S.elemAtRank( j ) < pivot do

j 🡨 j + 1

if j < k then

S.swapElements(S.atRank( j ), S.atRank( k ))

S.swapElements(S.atRank( lo ), S.atRank( k )) {move pivot to sorted rank}

return k

**C-4-19**

**Algorithm** countInversions(S, C, count)

Input sequence S with n elements, comparator C

Output number of inversions in S

if S.size() > 1 then

(S1, S2) 🡨 partition(S, n/2)

countInversions (S1, C, count)

countInversions (S2, C, count)

count 🡨 merge(S1, S2, C, count)

return count

return 0

**Algorithm** merge(A, B, C, count)

Input sequences A and B with n/2 elements each, comparator C

Output number of inversions in S

S 🡨 empty sequence

while A.isEmpty() ∧ B.isEmpty() do

if C.isLessThan( B.first().element(),A.first().element() ) then

S.insertLast(B.remove(B.first()))

Count 🡨 count + 1

else

S.insertLast(A.remove(A.first()))

while A.isEmpty() do

S.insertLast(A.remove(A.first()))

while B.isEmpty() do

S.insertLast(B.remove(B.first()))

return S

**C-4.25**

**Algorithm** matchBolts(A, B)

Input sequence A of n nuts and a sequence B of n bolts

Output sequence contains items of matched nuts and bolts

S 🡨 create new sequence 1

While A.isEmpty() ∧ B.isEmpty() do n

a 🡨 A.removeFirst() n

b 🡨 B.first() n

while a.match(b) do n2

b 🡨 B.next(b) n2

{A match has been found)

b 🡨 B.removeElement(b) n

S.insertItem((a,b)) n

Return S 1

Total running time is O(n2)