**Assignment 6**

C-4.13 Suppose we are given two sequences A and B of *n* elements, possibly containing duplicates, on which a total order relation is defined. Describe an efficient algorithm for determining if A and B contain the same set of elements (possibly in different orders). What is the running time of this method?

Answer:

Algorithm is\_same\_set(A, B)

if A.isEmpty() and B.isEmpty() then

return T

else if A.size() != B.size() then

return F

i🡨1

b🡨T

while b=T and i<=n do {

e🡨A.elementAtRank(i)

j🡨i

b 🡨 F

p🡨B.atRank(i)

while !b and j<=n do {

q🡨B.atRank(j)

if q.element()=e then {

b🡨T

B.swapElements(p,q)

}

j++

}

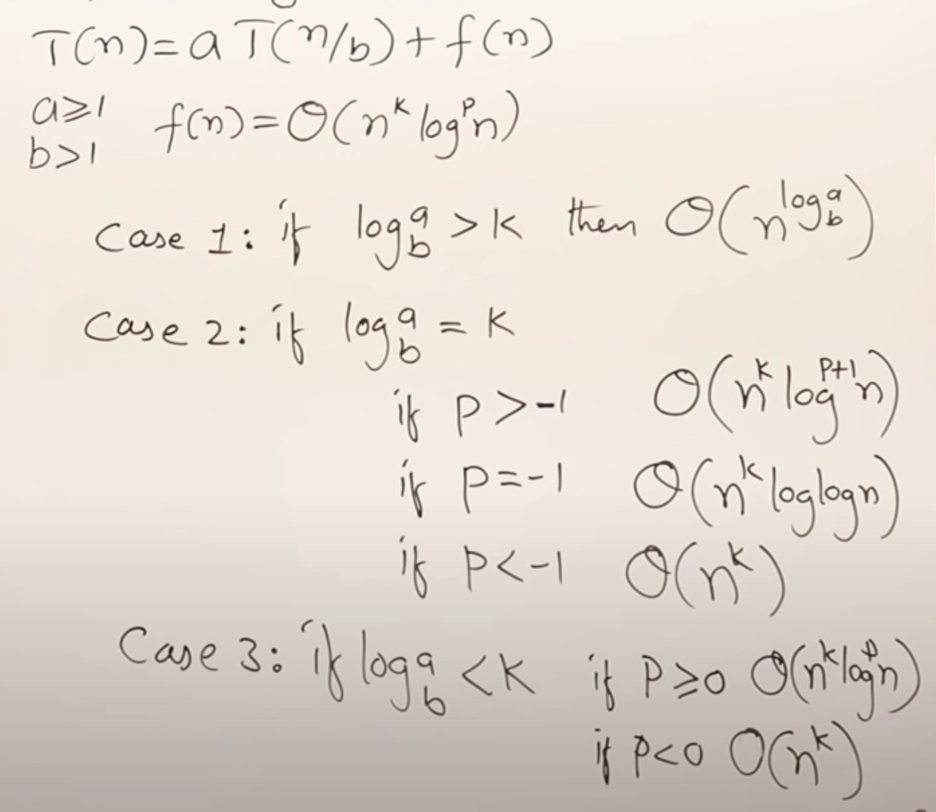
i++

}

return b

R-5.4 Characterize each of the following recurrence equations using the master method (assuming that *T(n)* = *c* for *n < d*, for constant *c > 0* and *d* ≥ *1*).

The form:



a. T(n) = 2T(n/2) + logn

f(n) = Θ(n0log1n)

logba = 1 > k = 0

so, case 1 says T(n) is Θ(n)

b. T(n) = 8T(n/2) + n2

f(n) = Θ(n2log0n)

logba = 3 > k = 2

so, case 1 says T(n) is Θ(n3)

c. T(n) = 16T(n/2) + (nlogn)4

f(n) = Θ(n4log4n)

logba = 4 equals k = 4

so, case 2 says T(n) is Θ(n4log5n)

d. T(n) = 7T(n/3) + n

f(n) = Θ(n1log0n)

logba = 1 equals k = 1

so, case 2 says T(n) is Θ(nlogn)

e. T(n) = 9T(n/3) + (n3logn)

f(n) = Θ(n3log1n)

logba = 2 < k = 3

so, case 3 says T(n) is Θ(n3logn)