```
Algorithm sortRR(L).
    if L.isEmpty() then
        return L.

    cur ← L.first().

    while !L.isLast(cur) do
        cur ← L.after(cur).
        b ← cur
        while !L.isFirst(b). do {
            c ← L.before(b).
            if C.color = RED ∧ b.color = BLUE' then.
            L.swapElement(c, b).
            b ← c
        }
```

2

Algorithm sortRGB(L)

if L.isEmpty() then
    return L.
cur ← L.first()
while !L.isLast(cur) do
    cur ← L.after(cur).
    b ← [illegible](cur)
    while !L.isFirst(b)
        c ← L.before(b).
        if b.color() != c.color() ∧ b.color() = RED then
            a.swapElement(b, c)
        if b.color() = BLUE ∧ c.color() = GREEN
            L.swapElemt(b, c).
        b ← c.

∧ and. conjunction
∨ or disjunction

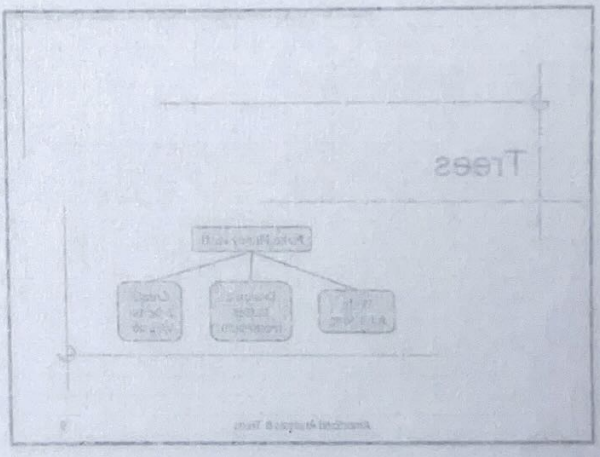
∩ intersection.
∪ union

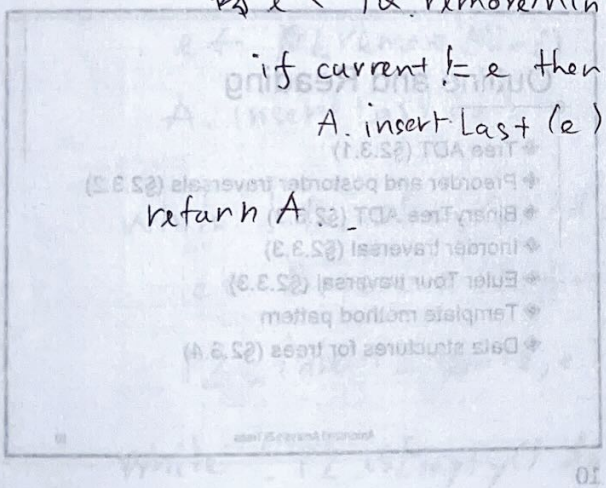
Algorithm noduplicate (A, B)

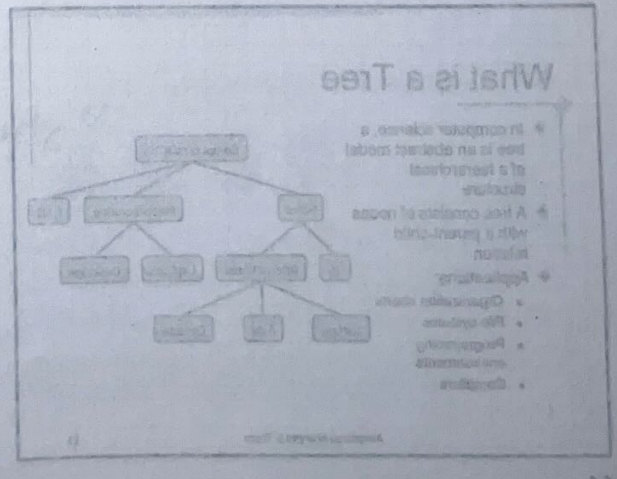
PQ ← new priority queue using C.
while ! A.isEmpty() do.
PQ.insertLast(A.remove(A.first()); e)
While ! B.isEmpty() do.
PQ.insertLast(B.remove(B.first()), e).

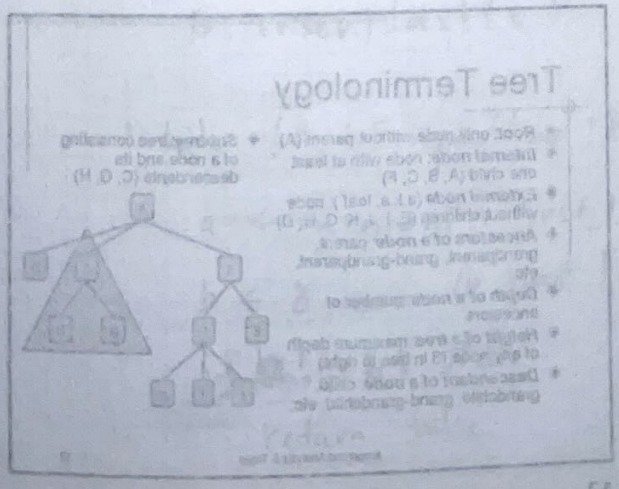
currentKe ← PQ.removeMin()
while ! PQ.isEmpty().do. → A.InsertLast.(current)

e ← PQ.removeMin()
if current != e then.
A.insertLast(e)
return A.









3