**Data**: course registrations $T$
**Result**: Sequential patterns $F$
$k := 1$;
$F_k := \{i | i \in I \wedge \frac{\sigma(i)}{N} \geq minsup\}$;
**repeat**
  $k := k + 1$;
  $C_k := apriori\_gen(F_{k-1})$;
  **foreach** *student registration* $t \in T$ **do**
    **foreach** *candidate* $c \in C_k$ **do**
      **if** *contains(t, c, 0, 1)* **then**
        $\sigma(c) := \sigma(c) + 1$
      **end**
    **end**
  **end**
  $F_k := \{c | \frac{\sigma(c)}{N} \geq minsup\}$
**until** $F_k = \emptyset$;
**return** $\cup F_k$

**Algorithm 1:** Apriori-like algorithm

Each element is allowed to have 1 event, so the second case in the book is applied when merging 2 sequences.

**Function** *Apriori_gen*($F_k$) **is**
    **Data**: $F_k$: $k$-length sequences
    **Result**: $k+1$-length candidates $C_{k+1}$
    $C_{k+1} := \emptyset$;
    **foreach** $(a, b) \in P(|F_k|, 2)$ **do**
        **if** $a[2 : end] = b[1 : end - 1]$ **then**
            $C_{k+1} := C_{k+1} \cup concat(a, b[end])$
        **end**
    **end**
    **return** $C_{k+1}$
**end**

**Algorithm 2:** Apriori-gen algorithm

```
Function contains(t, c, j, depth) is
    Data: t: registration sequence, c: sequential pattern
          candidate, i: index of element in t that previous
          event e = c[depth − 1] in c belongs to, depth:
          current depth of the recursion
    Result: true if c is in t, false otherwise
    foreach j ∈ [i + mingap : i + maxgap] do
        if c[depth] ∈ t[j] then
            if depth = |c| then
                return true
            else
                if contains(t, c, j, depth+1)=true then
                    return true
                end
            end
        end
    end
    if depth = 1 then
        return false
    end
end
```

**Algorithm 3:** Checking if candidate $c$ is in registration $t$ with constraints