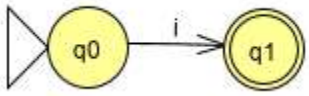
2. Assume $\Sigma = \{\mathtt{i}, \mathtt{f}, \mathtt{n}, \mathtt{t}, \mathtt{x}\}$ ($\mathtt{x}$ stands for any character different from $\mathtt{i}, \mathtt{f}, \mathtt{n}, \mathtt{t}$.) Construct the minimal DFSM to solve the multi-pattern searching problem for the patterns $p_1 = \mathtt{if}, p_2 = \mathtt{int}$. (This is used for keyword identification.) Show your work. You are allowed to use Thomson's construction or directly build an NDFSM.

First, define the language expressed by a regular expression that will accept the patterns. If we input a string letter by letter, this language will be in an accepting state whenever the current character marks the end of an occurrence of a pattern.

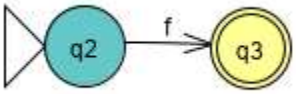
$$L\left(\Sigma^*(if\ U\ int)\right)$$

Then, create NFA by using Thompson's Construction (ignore state names please):

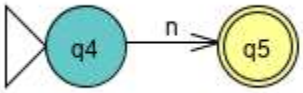
- Step 1 – 5 - Build:
  - NFA A for L(i)



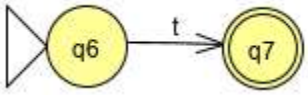
  - NFA B for L(f)



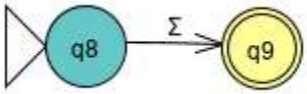
  - NFA C for L(n)



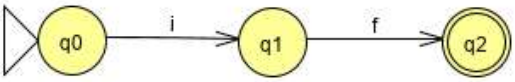
  - NFA D for L(t)



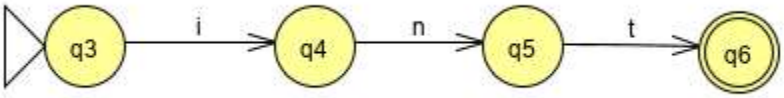
  - NFA E for Σ



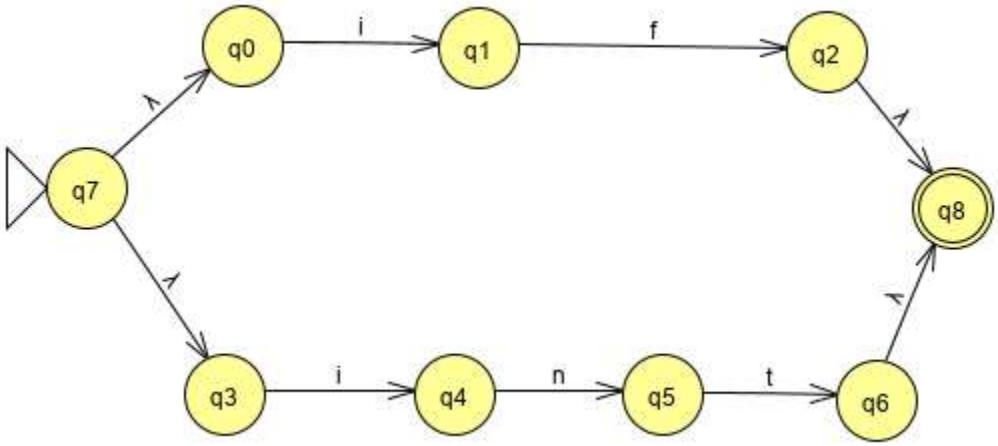
- Step 6-7 - Build:
  - NFA F for L(if) using NFAs A, B



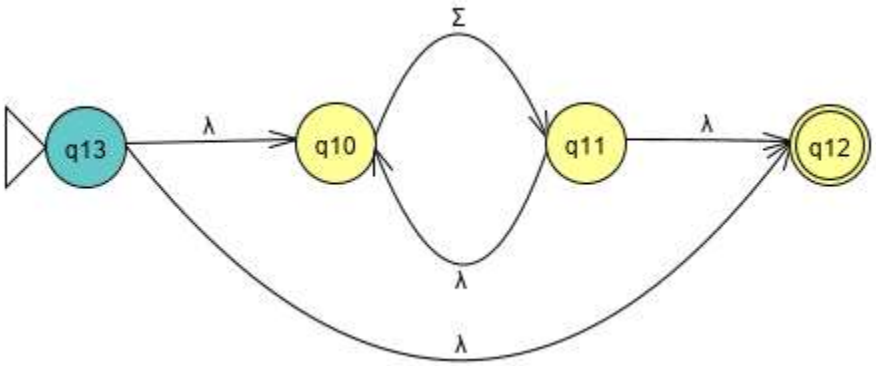
  - NFA G for L(int) using NFAs A, C, D



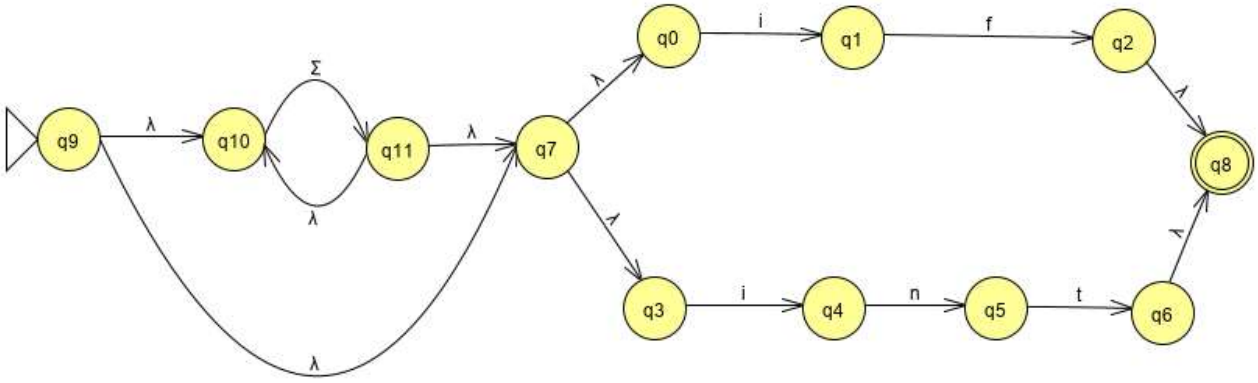
- Step 8 – Build NFA H for $L(if \cup int)$ using NFAs F, G



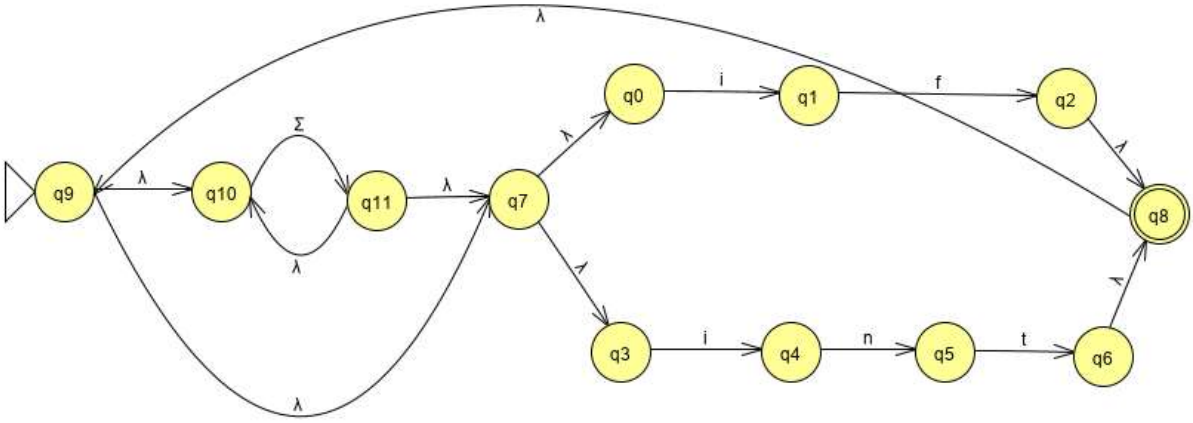
- Step 9 – Build NFA I for $L(\Sigma^*)$ using NFA E



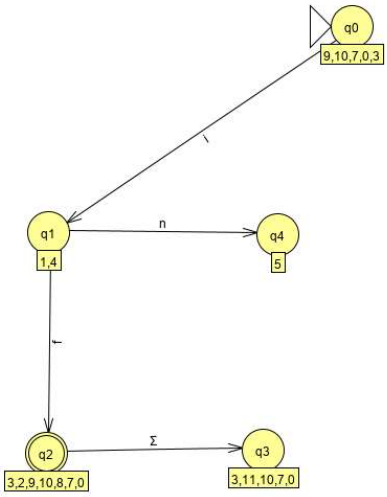
- Step 10 – Build NFA J for $L\left(\Sigma^*(if\ U\ int)\right)$ using NFAs H, I



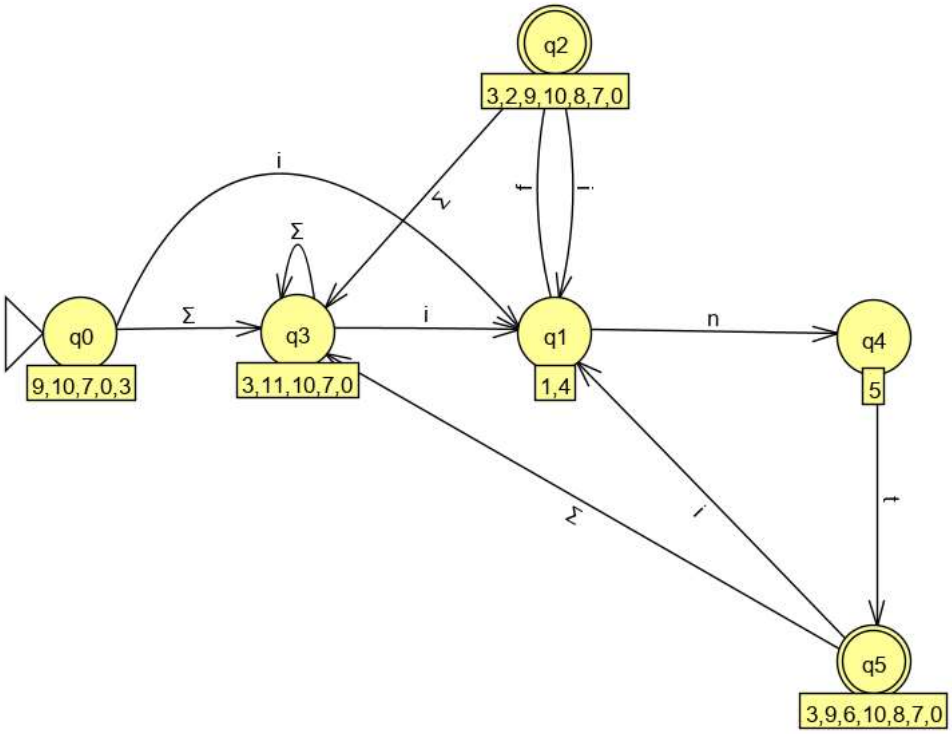
- Step 11. Since we want to find all occurrences of the patterns, we will modify the above NFA so that we can continue searching after finding an appearance. Whenever the NFA is in an accepting state, that means we have found a occurrence. In this case, we can enable this repeat search by creating an empty transition from the final state to the initial state.



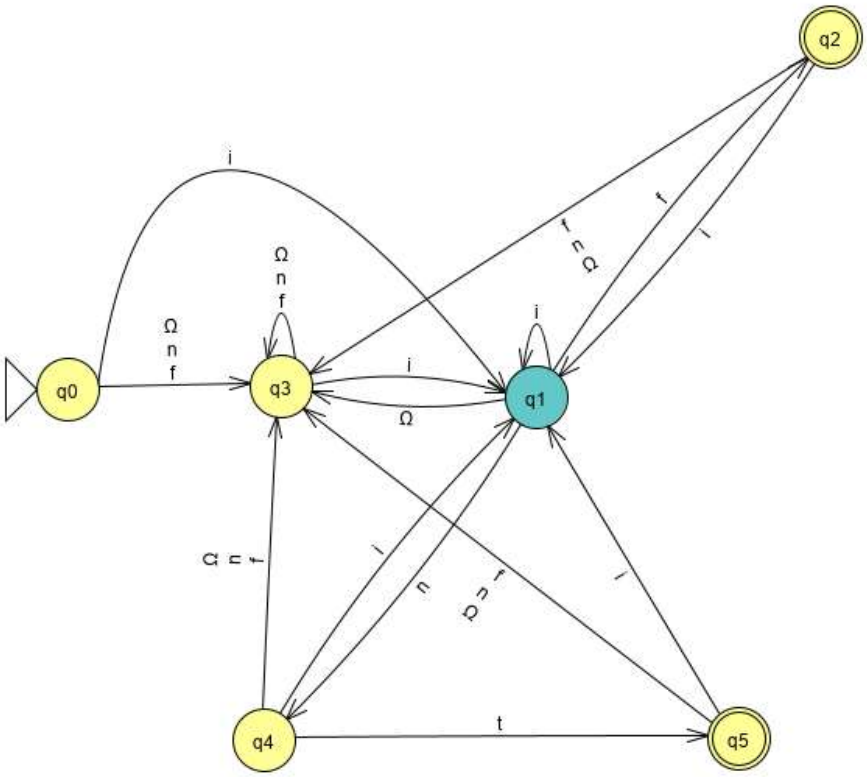
- Step 12. Convert to DFSM.
  - Start with s' = eps(s) = = eps(9) = {9, 10, 7, 0, 3}.
  - Compute DFSM transition states δ', $initially = \emptyset$.
    - active-states = {9, 10, 7, 0 ,3}. Create state 0 that represents the union of (9 u 10 u 7 u 0 u 3)
    - For each active state, create or connect states by tracking where each terminal takes them (ignoring dead states). Example:
      - Expand state 0 on transition through terminal 'i' to create state 1 (1 u 4).
      - Expand state 1 on transition through terminal 'f' to create state 2 (3 u 2 u 9 u 10 u 8 u 7 u 0) (2 is a final state because 8 was the final state in NFA)
      - Expand state 2 on transition through terminal 'Σ' to create state 3 (3 u 11u 10 u 7 u 0)
      - Expand state 1 on transition through terminal 'n' to create state 4 (5)



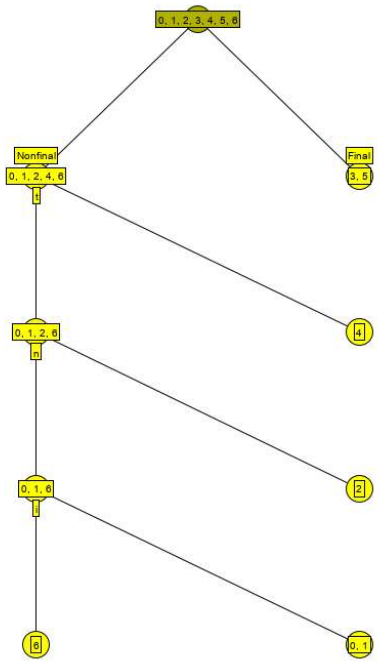
    - Continue doing this until we have no more active states such that $\delta'(State, terminal)$ is unknown.



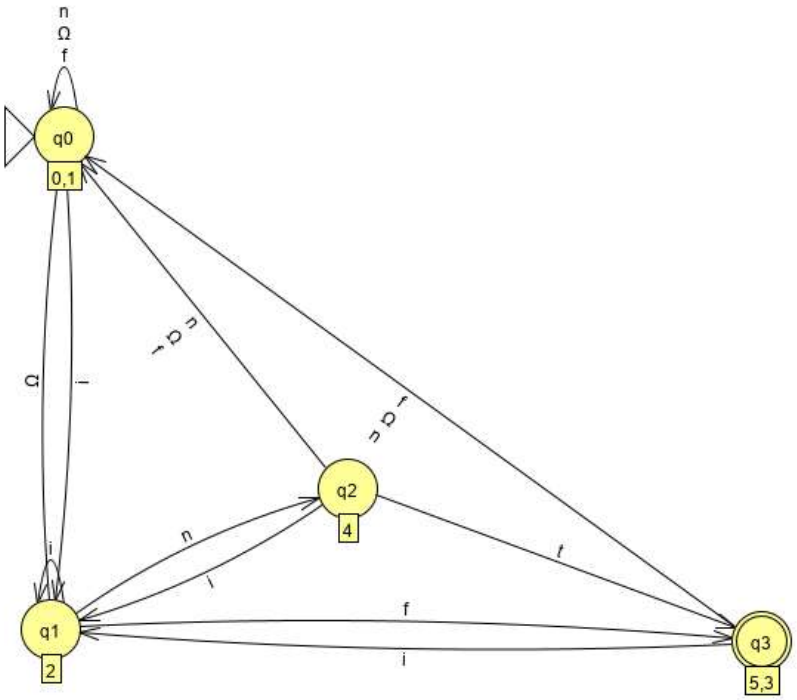
- Step 13: Before we minimize this DFSM, we must first fix another problem. Since Σ includes i, f, and t, we must recognize that Σ, i, f, t overlap. Therefore, this is **NOT determinant and this is NOT A DFSM.** We fix this by letting $\Omega = \Sigma - i - f - t$. Then, we input Ω *to replace* Σ and add the appropriate transitions:



- Step 14: Minimize DFSM. By using the overclustering method, we can minimize this tree. First, we split the states into two groups (A, K-A).
    - Then, for each group, we attempt to split them into groups with a terminal c, such that each group enters a distinct existing group after transitioning via c (including group we're splitting).
    - We do this until we can't split any group any further. After this process, we see that there are 2 groups that have more than 1 state, meaning that we have 2 pairs of redundant states. This can be seen in this diagram:



    - Remove states 1 and 5 and rewire their transitions:



Now we have a minimal DFSM for the multi-pattern searching problem where $\Sigma = \{i, f, n, t, x\} and\ p1 = if, p2 = int$

The following traceback demonstrates how DFSM will be in a accepting state iff we are at the end of an occurrence of p1 or p2.