Hopcroft's algorithm

- We want to identify and merge any equivalent states in the DFA: states produce identical behaviour across all inputs
- It takes the set of all states, and repeatedly partitions them into smaller and smaller subsets by identifying ways in which some states in a subset behave differently than others in the subset
- First: split the set of all states into accept/non-accept
- Repeat: pick an input character, check if each state in a partition takes you to a common partition (e.g. on input x, do all states in partition P transition to states in partition Q)

The algorithm

Current = { AcceptStates, NonAcceptStates }

P = { }

Repeat until P and Current are the same:

    P = Current

    Current = P

    For each set, s, in P

        Current = Union of Current and Split(s)

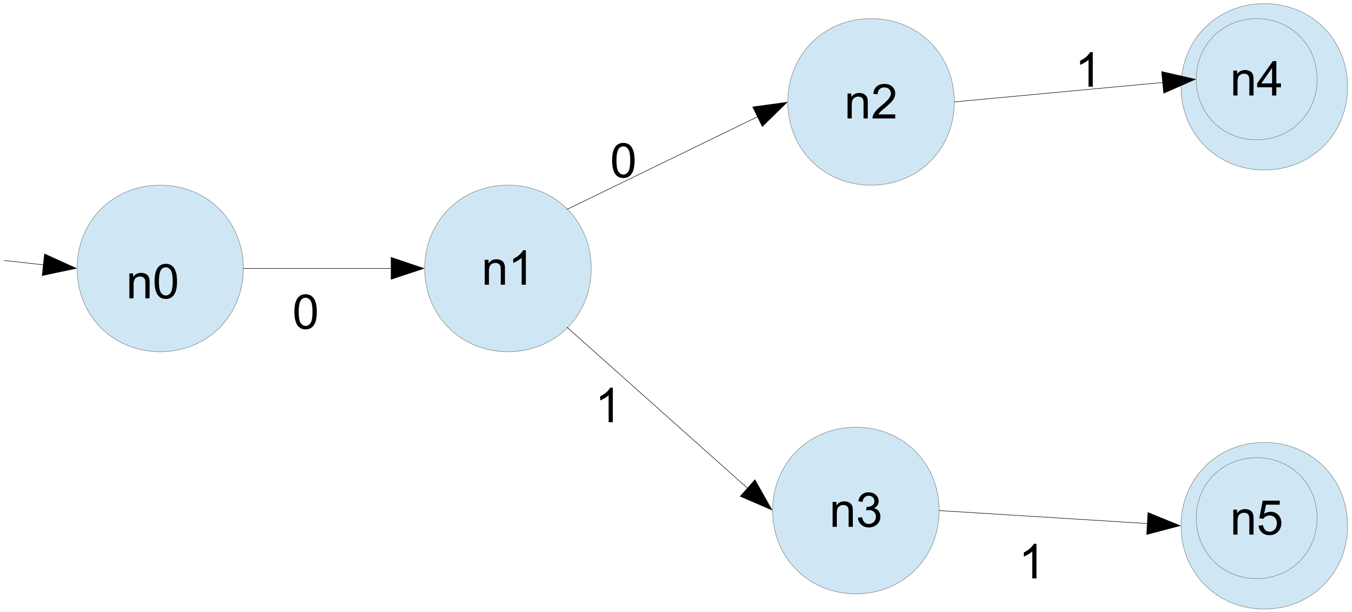
(split tries to break s into two sets of “distinguishable” states)

Split(s)

```
for each input character, c:
    for each state in s
        determine which partition T[s][c] leads to
        if some states of s lead to one partition
            and other states do not, use that to
            divide them into two sets, s1 and s2
            and return { s1, s2 }
return S
```

Example 1

- Will try to minimize the following DFA



Example 1

First, divide into accept/non { {n0,n1,n2,n3}, {n4,n5} }

Make first pass through repeat loop

- in first pass of for loop, try Split({n0,n1,n2,n3})
  - on input 0, n2/n3 go to states in p1, n0/n1 do not
  - Current = { {n0,n1}, {n2,n3} }
- in second pass of for loop, try Split({n4,n5})
  - no differences found between n4/n5 on either input
  - Current = { {n0,n1}, {n2,n3}, {n4,n5})

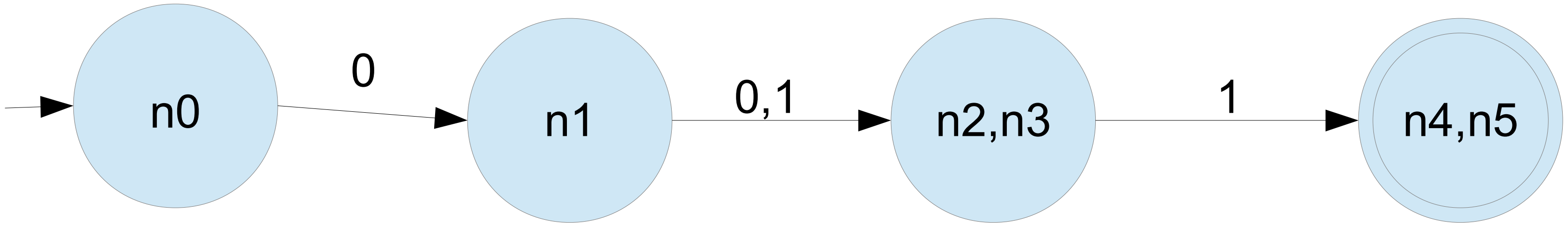
Example 1 continued

Second pass of repeat loop

- First for loop pass, try Split({n0,n1})
  - On input 1, n0->n1, n1->n2, in different partitions
  - Current = { {n0}, {n1} }
- Second for loop pass, try Split({n2,n3})
  - No changes found
  - Current = { {n0}, {n1}, {n2,n3} }
- Third for loop pass, try Split({n4,n5})
  - No changes found
  - Current = { {n0}, {n1}, {n2,n3}, {n4,5} }

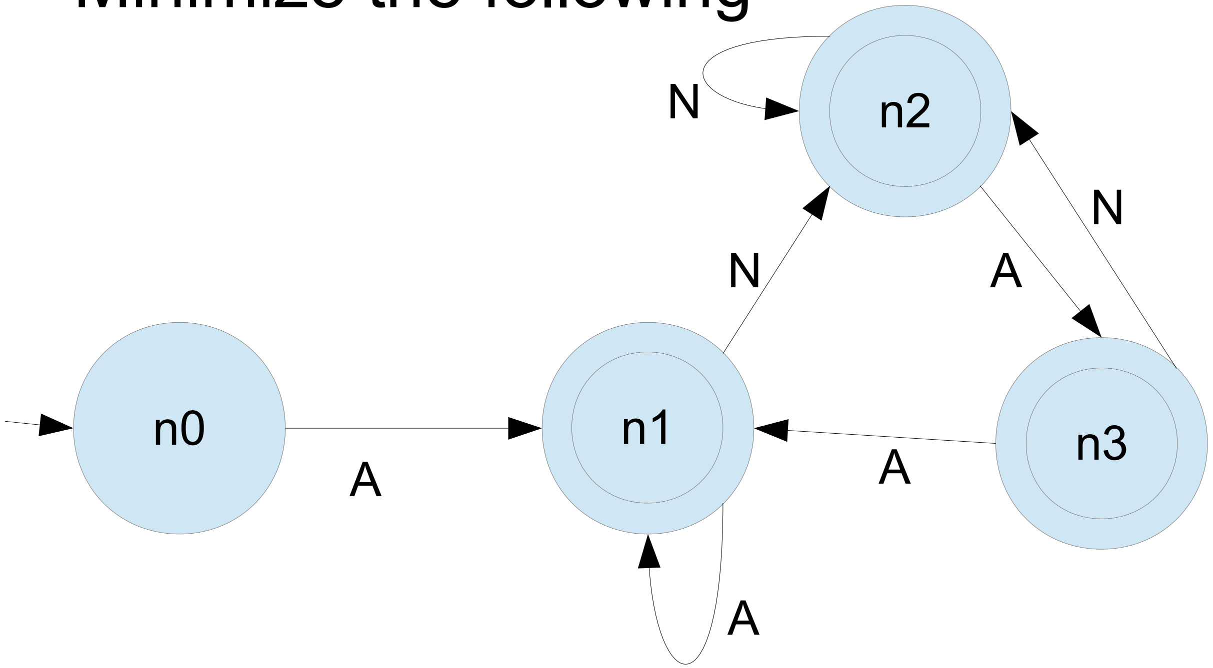
Example 1: final

- Can end now that P and Current are the same, final partitioning realizes n2,n3 are equivalent, and that s4,s5 are equivalent, giving minimized DFA



Example 2

- Minimize the following



Example 2 continued

Current = { {n0}, {n1,n2,n3} }

Split({n0}) can't possibly split further

Split({n1,n2,n3})

For every state/input combination we get one of n1,n2,n3

Can't subdivide further, current is actually the right partition