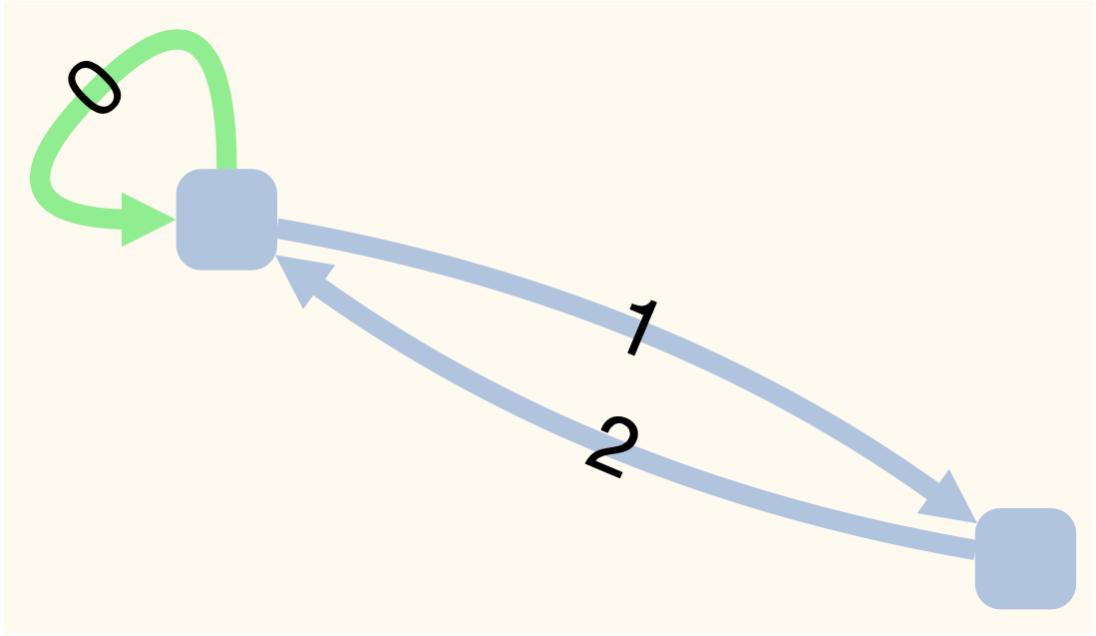
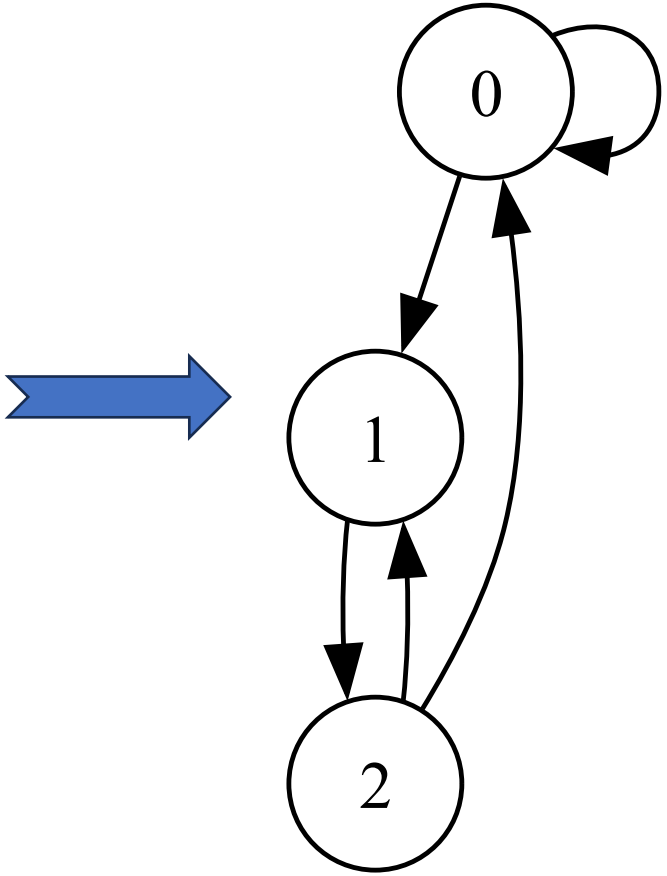
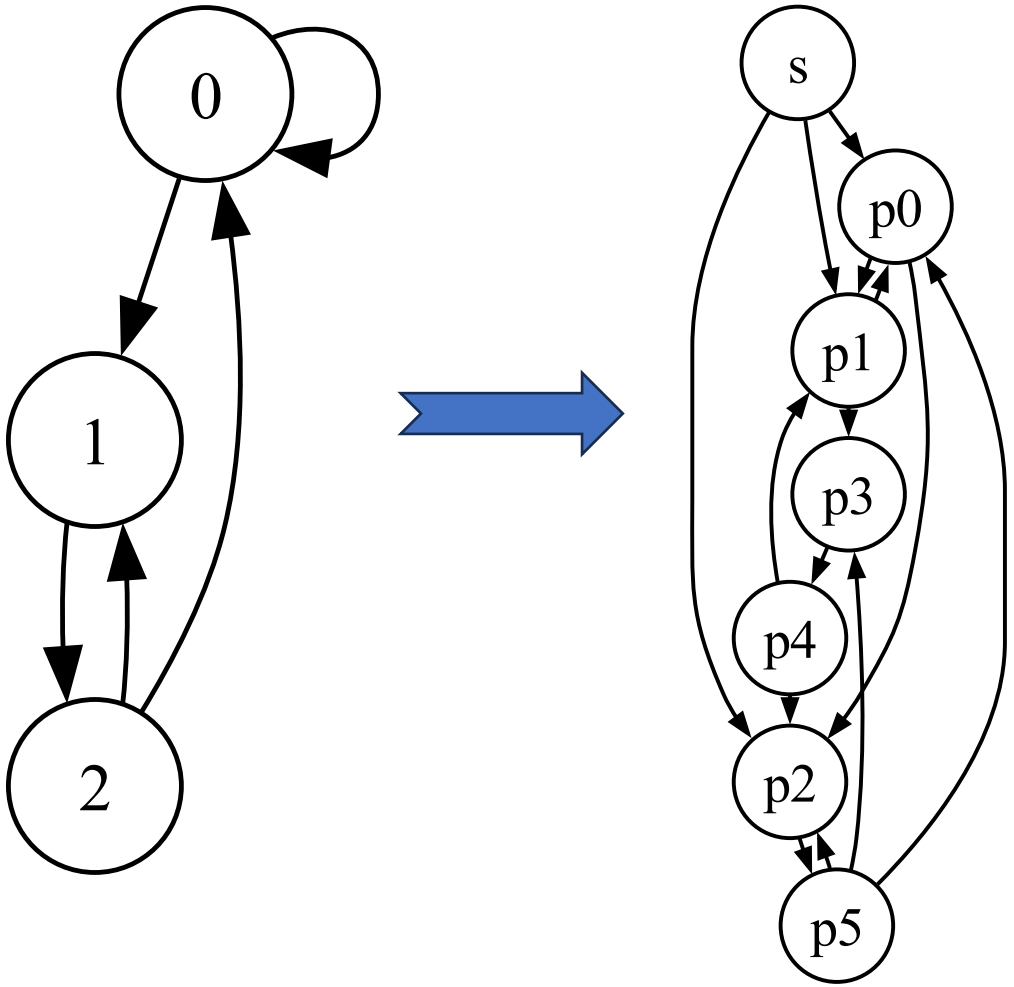
- A GraphWalker model (on the left) is a *Loop-Permitting Directed Multigraph*.
  - There can be multiple edges in the same direction between two vertices (messes up with test gen. algos)
- We must generate a *Simple Graph* (on the right) by converting every edge to a vertex.
  - Vertex Coverage in the Simple Graph = Edge Coverage in the GraphWalker model.
  - Edge Coverage in the Simple Graph = Edge Pair Coverage in the GraphWalker model.
  - For Vertex Coverage in the GraphWalker model => Delete all multiple edges to obtain the Simple Graph.



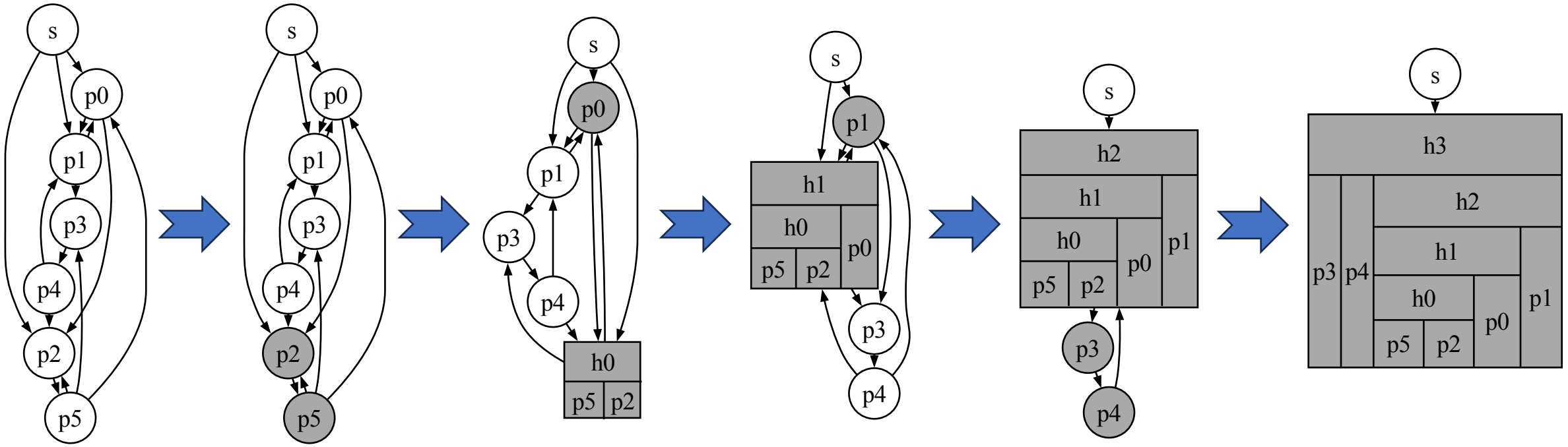


- Test Requirements are a set of vertex paths (or paths, in short).
- We construct a Path Graph of Test Requirements where
  - ”s” represents the start and it is an empty path.
  - Every edge is a splice between two paths that does NOT unintentionally cover another test requirement.
  - We do NOT need a “t” to represent test termination, because we can terminate the test at any vertex.

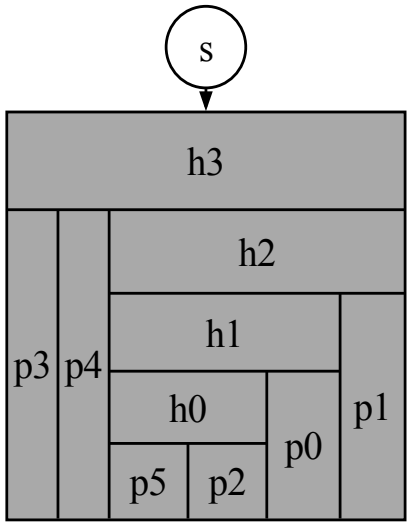


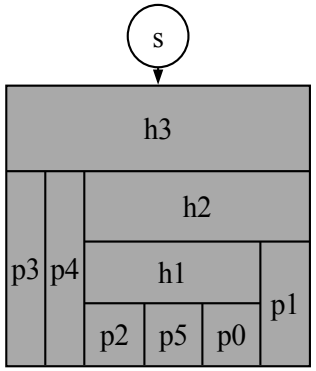
	TEST REQUIREMENTS FOR PRIME PATH COVERAGE
p0	00
p1	0120
p2	121
p3	1201
p4	2102
p5	212

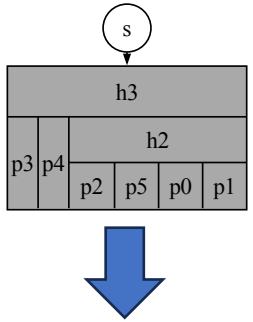
- We must convert the Path Graph into an acyclic Hyper-Path Graph
  - A hyper-path is either a path or a cycle of hyper-paths.
  - We incrementally replace every cycle with a hyper-path.
- The final Hyper-Path Graph has only two vertices!
  - The original GraphWalker model is restartable, i.e., there always exists a path back to the initial element.
  - Therefore, the GraphWalker model is strongly connected.
  - So, we do not need Minimum Flow Algorithm.

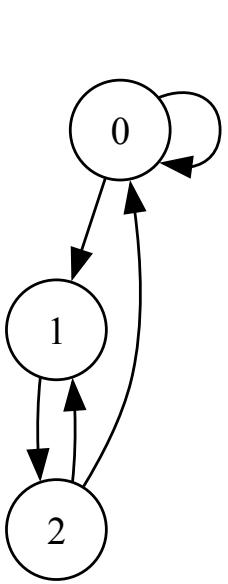


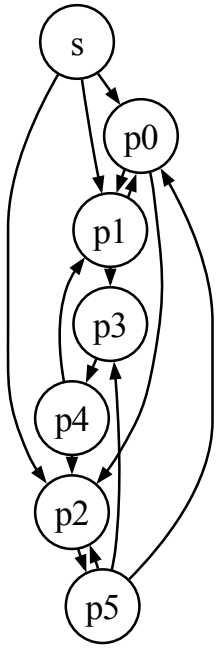
- Remember the original Path Graph.
  - Just rotate and combine hyper-paths until they form a fully splicable path trace.
- We must splice “s” and the final path trace.
- Rotate the final path trace until we get the shortest test.
- To reproduce:
  - `git clone https://github.com/yavuzkoroglu/gwplus.git`
  - `cd gwplus`
  - `make bin/toygraph`
  - `bin/toygraph`



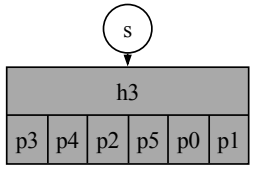




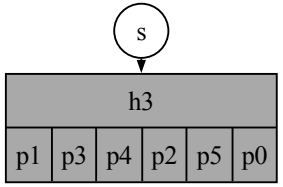




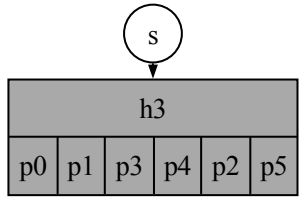
p0	00
p1	0120
p2	121
p3	1201
p4	2102
p5	212



s -> p3 does NOT exist in the Path Graph



Test Path = 0120121200
Test Length = 10



Test Path = 001201212
Test Length = 9