## Dijkstra's Implementation

- don't modify the Graph ADT!
  - labels like `dist`, `prev` should be stored separately rather than as part of `Vertex`
- `init()` should fully initialize all of the instance variables used in `run()`
  - initialization at the point of declaration is only done when an object is first constructed
  - `init()` is meant to be used each time there's a new start vertex for subsequent `run()`
- there is no notion of equivalence for vertices and edges – "the same" means "the same object"
  - compare using == rather than `equals()`



## Dijkstra's Implementation

- with the exception of implementing `decreaseKey` by removing and re-inserting into the PQ, you should achieve the running time discussed in class
  - use the right Graph implementation
  - `getPathVertices()` and `getPathEdges()` should be O(1) per element – make sure that building up the path to return is also O(1) per element added
- avoid unnecessary operations even if the overall big-Oh isn't affected
  - O(1) operations are cheap, but it's even cheaper to skip them
  - why do two O(n) things when you can do just one?
    - is it necessary to check `contains()` before calling `remove()`?
    - is it necessary to reverse a list? can you build it in the right order to start with?



## Programming Assignment Resubmits

- programming assignments count towards both engagement and mastery
  - mastery grade is based on the final handin + interview
- for any assignment, resubmits (with substantive improvements) earn an additional +1 for engagement
  - (no +1 if the resubmit is your first handin)