Peer Review

- Drafts will go out this afternoon
- Review will be due on Friday
- You will submit feedback on Moodle and then I will distribute it
    - Reviewers will be anonymous
    - The review of your paper will not affect your grade
    - I will give you a grade on your review

- What makes a good review?

  - Constructive feedback

    - Focus on the writing, not the writer

    - Identify what can be improved, and provide recommendations

  - Point out what background information is missing in order to understand the work

- Identify superfluous content that can be eliminated
- Help improve flow by pointing out awkward transitions

Dijkstra's Algorithm

- Negative weights are not allowed
- Weighted version of breadth-first search
  - Uses a priority queue instead of a FIFO queue
  - S = set of vertices whose final shortest path values are computed
  - Q = priority queue representing V - S

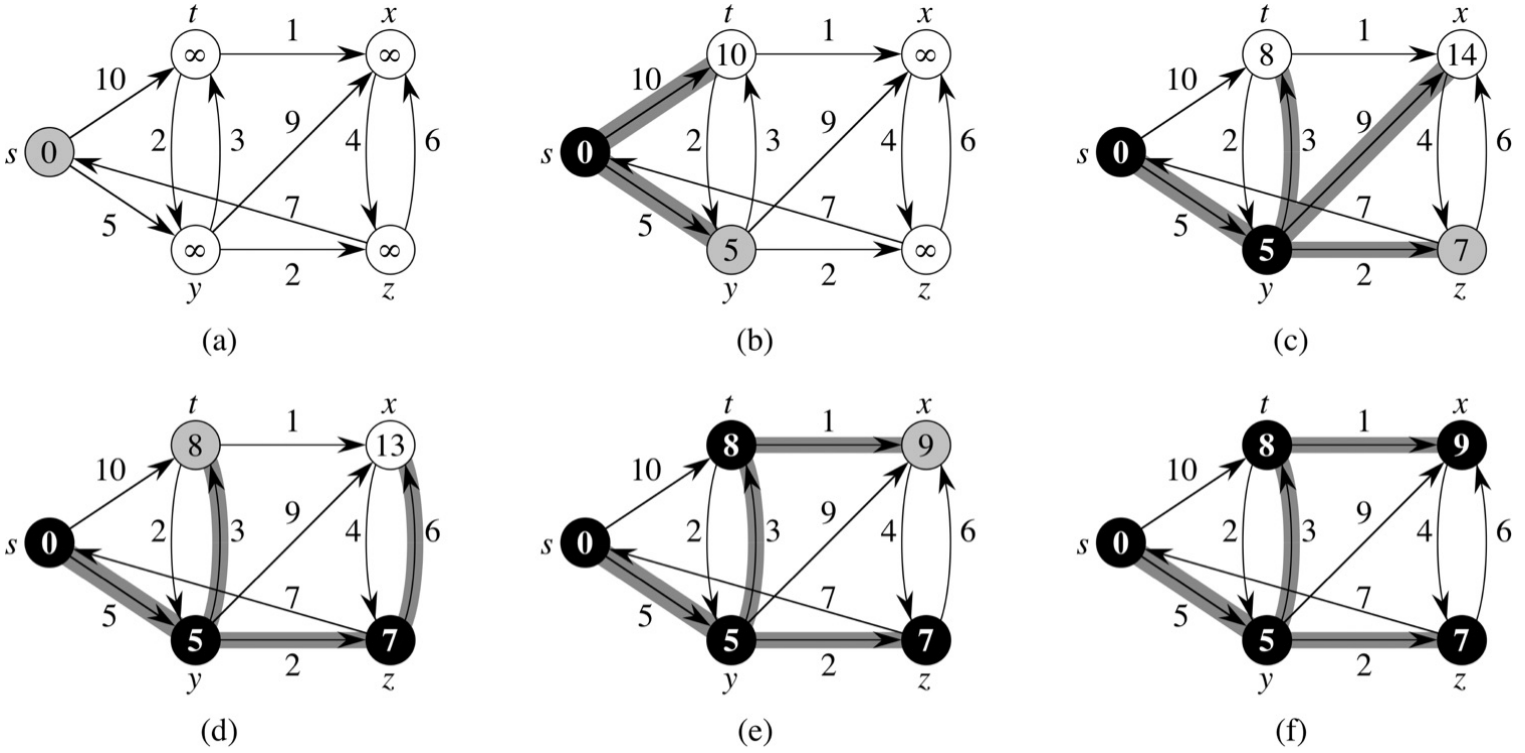
- Repeatedly select the vertex u in $V - S$ with the minimum shortest path estimate $u.d$, add u to S, and relax all edges leaving u

- Greedy!

```
DIJKSTRA(G, w, s)
  INIT-SINGLE-SOURCE(G, s)
  S = ∅
  for each vertex u ∈ G.V
      INSERT(Q, u)
  while Q ≠ ∅
      u = EXTRACT-MIN(Q)
      S = S ∪ {u}
      for each vertex v ∈ G.Adj[u]
          RELAX(u, v, w)
          if v.d changed
              DECREASE-KEY(Q, v, v.d)
```

Q is a priority queue

Extract-Min returns the vertex with min d value


t
x
s
y
z
10
1
9
2
3
4
6
5
7
2
∞
0
(a)
10
5
(b)
8
14
7
(c)
13
(d)
9
(e)
(f)

- Why does it work?

  - Loop invariants

    $Q = V - S$

    $v.d = \delta(s, v)$ for each vertex $v \in S$

  - If the second invariant holds, then we have found the shortest path to a vertex when it is removed from the queue

  - This is trivially true for S

- Adding u to S means we have found a shortest path to u using only vertices in S

If the invariant did not hold, that would mean there could be a shorter path to u through vertices not yet in S. But because we chose u based on its low d value, all of those paths would necessarily be as long or longer

- Time complexity
  - Every vertex is added to Q exactly once
  - Every edge is examined exactly once
  - The algorithm calls Decrease-key at most $|E|$ times
  - Time complexity depends on the implementation of the queue

Using an array for Q

- Insert $O(1)$, append to array
- Decrease-key is $O(1)$
- Extract-Min is $O(V)$ since we have to search the array
- Leads to $O(V^2 + E) = O(V^2)$

Using a Fibonacci Heap (chapter 19)

- Amortized cost of Extract-Min is $O(\lg V)$
- Amortized cost of Decrease-key is $O(1)$
- Overall complexity $O(V \lg V + E)$
- Fibonacci heaps were created for Dijkstra's Algorithm