Shortest Paths

- Shortest Paths
- Properties of Shortest Paths
- Dijkstra's Algorithm
- Shortest Paths on DAGs

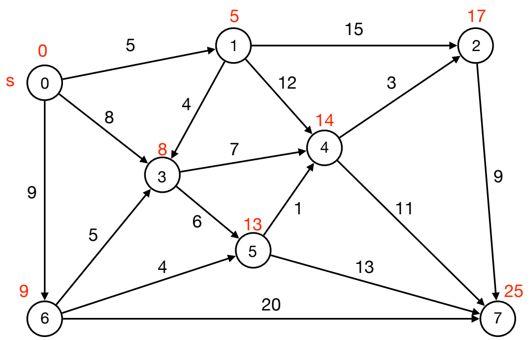
Philip Bille

Shortest Paths

- Shortest Paths
- Properties of Shortest Paths
- Dijkstra's Algorithm
- Shortest Paths on DAGs

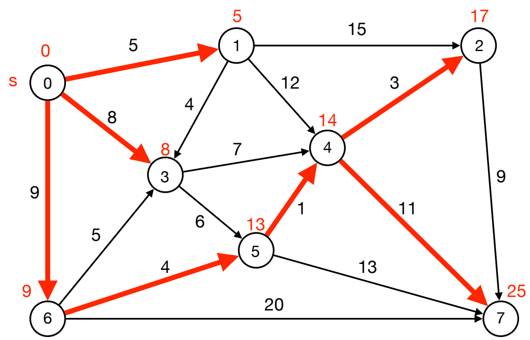
Shortest Paths

- Shortest paths. Given a directed, weighted graph G and vertex s, find shortest path from s to all vertices in G.



Shortest Paths

- Shortest paths. Given a directed, weighted graph G and vertex s, find shortest path from s to all vertices in G.
- Shortest path tree. Represent shortest paths in a tree from s.



Applications

- Routing, scheduling, pipelining, ...

Shortest Paths

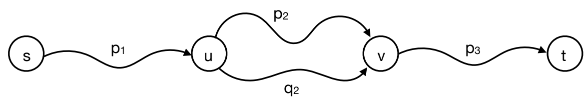
- Shortest Paths
- Properties of Shortest Paths
- Dijkstra's Algorithm
- Shortest Paths on DAGs

Properties of Shortest Paths

- Assume for simplicity:
  - All vertices are reachable from s.
- $\Longrightarrow$ a (shortest) path to each vertex always exists.

Properties of Shortest Paths

- Subpath property. Any subpath of a shortest path is a shortest path.
- Proof.
  - Consider shortest path from s to t consisting of p_1, p_2 and p_3.



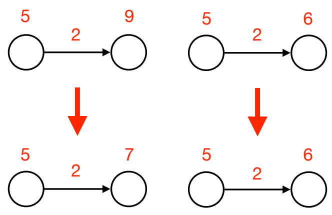
  - Assume q_2 is shorter than p_2.
  - $\Longrightarrow$ Then p_1, q_2 and p_3 is shorter than p.

Shortest Paths

- Shortest Paths
- Properties of Shortest Paths
- Dijkstra's Algorithm
- Shortest Paths on DAGs

Dijkstra's Algorithm

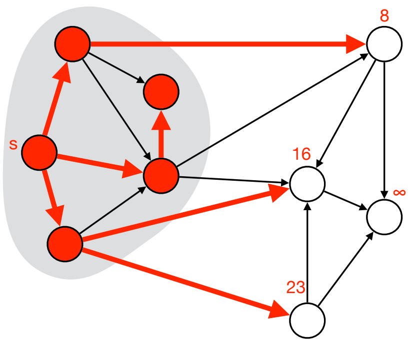
- Goal. Given a directed, weighted graph with non-negative weights and a vertex s, compute shortest paths from s to all vertices.
- Dijkstra's algorithm.
  - Maintains distance estimate v.d for hver knude v = length of shortest known path from s to v.
  - Updates distance estimates by relaxing edges.

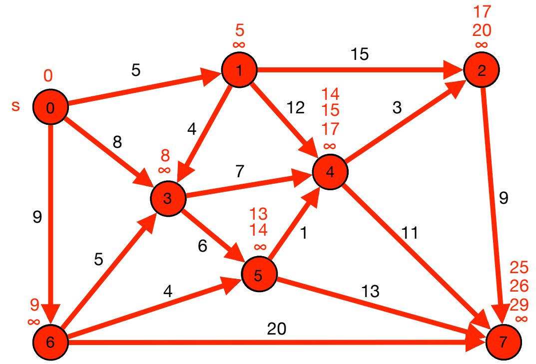


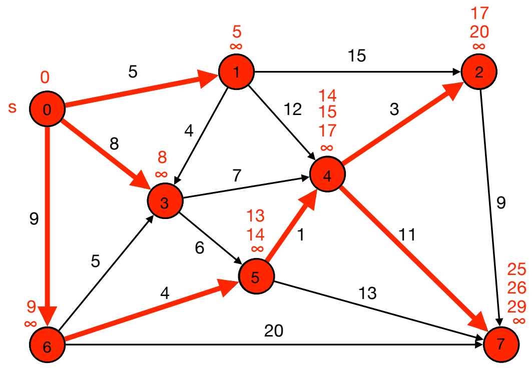

```
RELAX(u,v)
   if (v.d > u.d + w(u,v))
      v.d = u.d + w(u,v)
```

Dijkstra's Algorithm

- Initialize s.d = 0 and v.d = ∞ for all vertices v ∈ V\{s}.
- Grow tree T from s.
- In each step, add vertex with smallest distance estimate to T.
- Relax all outgoing edges of v.

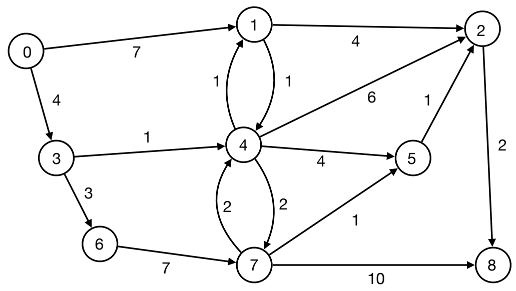






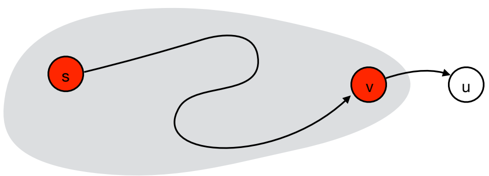
Dijkstra's Algorithm

- Initialize s.d = 0 and v.d = ∞ for all vertices v ∈ V\{s}.
- Grow tree T from s.
- In each step, add vertex with smallest distance estimate to T.
- Relax all outgoing edges of v.
- Exercise. Show execution of Dijkstra's algorithm from vertex 0.



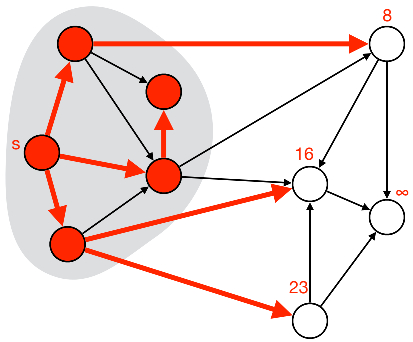
Dijkstra's Algorithm

- Lemma. Dijkstra's algorithms computes shortest paths.
- Proof.
  - Consider some step after growing tree T and assume distances in T are correct.
  - Consider closest vertex u of s not in T.
  - Shortest path from s to u ends with an edge (v,u).
  - v is closer than u to s $\Longrightarrow$ v is in T. (u was closest not in T)
  - $\Longrightarrow$ shortest path to u is in T except last edge (u,v).
  - Dijkstra adds (u,v) to T $\Longrightarrow$ T is shortest path tree after n-1 steps.



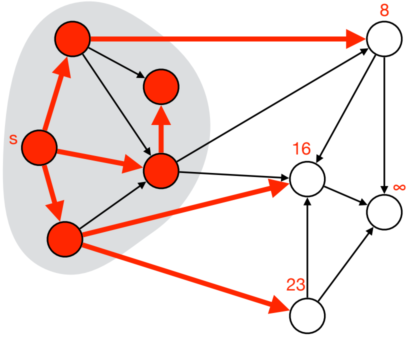
Dijkstra's Algorithm

- Implementation. How do we implement Dijkstra's algorithm?
- Challenge. Find vertex with smallest distance estimate.



Dijkstra's Algorithm

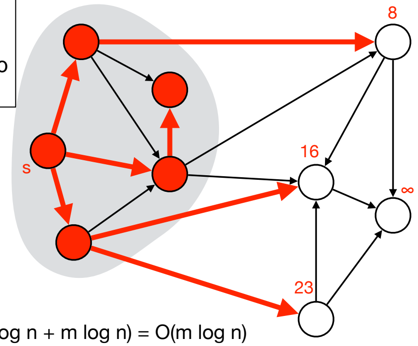
- Implementation. Maintain vertices outside T in priority queue.
  - Key of vertex v = v.d.
  - In each step:
    - Find vertex u with smallest distance estimate = Extract-Min
    - Relax edges that u point to with Decrease-Key.



Dijkstra's Algorithm

```
Dijkstra(G, s)
   for all vertices v∈V
      v.d = ∞
      v.π = null
      Insert(P,v)
   Decrease-Key(P,s,0)
   while (P ≠ ∅)
      u = Extract-Min(P)
      for all v that u point to
         Relax(u,v)
```

```
Relax(u,v)
   if (v.d > u.d + w(u,v))
      v.d = u.d + w(u,v)
      Decrease-Key(P,v,v.d)
      v.π = u
```



- Time.
  - n Extract-Min
  - n Insert
  - < m Decrease-Key
- Total time with min-heap. O(n log n + n log n + m log n) = O(m log n)

Dijkstra's Algorithm

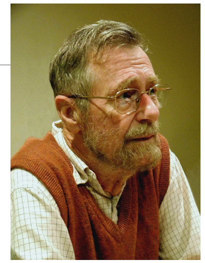
- Priority queues and Dijkstra's algorithm. Complexity of Dijkstra's algorithm depend on priority queue.
  - n Insert
  - n Extract-Min
  - < m Decrease-Key

Priority queue	Insert	Extract-Min	Decrease-Key	Total
array	O(1)	O(n)	O(1)	$O(n^2)$
binary heap	O(log n)	O(log n)	O(log n)	O(m log n)
Fibonacci heap	O(1)†	O(log n)†	O(1)†	O(m + n log n)

† = amortized

- Greed. Dijkstra's algorithm is a greedy algorithm.

Edsger W. Dijkstra



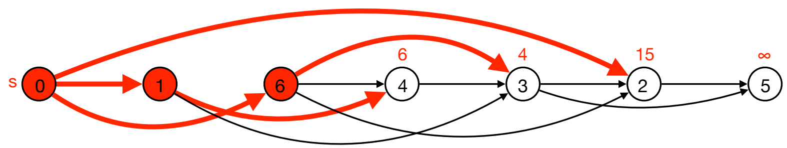
- Edsger Wybe Dijkstra (1930-2002)
- Dijkstra algorithm. "A note on two problems in connexion with graphs". Numerische Mathematik 1, 1959.
- Contributions. Foundations for programming, distributed computation, program verifications, etc.
- Quotes. “*Object-oriented programming is an exceptionally bad idea which could only have originated in California.*”
- “*The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence.*”
- “*APL is a mistake, carried through to perfection. It is the language of the future for the programming techniques of the past: it creates a new generation of coding bums.*”

Shortest Paths

- Shortest Paths
- Properties of Shortest Paths
- Dijkstra's Algorithm
- Shortest Paths on DAGs

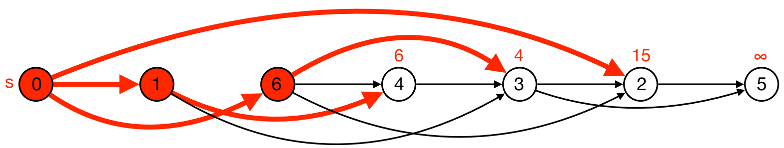
Shortest Paths on DAGs

- Challenge. Is it computationally easier to find shortest paths on DAGs?
- DAG shortest path algoritme.
  - Process vertices in topological order.
  - For each vertex v, relax all edges from v.
- Also works for negative edge weights.



Shortest Paths on DAGs

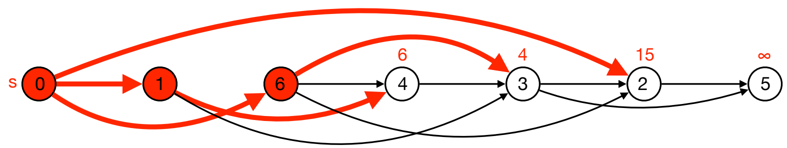
- Lemma. Algorithm computes shortest paths in DAGs.



- Proof.
  - Consider some step after growing tree T and assume distances in T are correct.
  - Consider next vertex u of s not in T.
  - Any path to u consists vertices in T + edge e to u.
  - Edge e is relaxed $\Longrightarrow$ distance to u is shortest.

Shortest Paths on DAGs

- Implementation.
  - Sort vertices in topological order.
  - Relax outgoing edges from each vertex.
- Total time. O(m + n).



Shortest Paths Variants

- Vertices
  - Single source.
  - Single source, single target.
  - All-pairs.
- Edge weights.
  - Non-negative.
  - Arbitrary.
  - Euclidian distances.
- Cycles.
  - No cycles
  - No negative cycles.

Shortest Paths

- Shortest Paths
- Properties of Shortest Paths
- Dijkstra's Algorithm
- Shortest Paths on DAGs