Edge-weighted DAGs

* Consider vertices in topological order
* Relax all edges pointing from that vertex

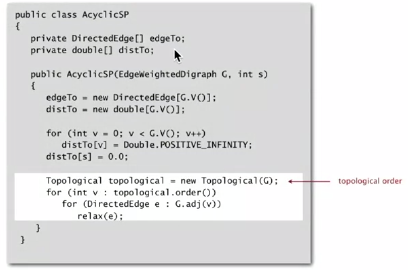
Proposition:

* topological sort algorithm computes SPT in any edge-weighted DAG in time proportional to E + V

Proof:

* Each edge e = v -> w is relaxed exactly once (when v is relaxed),  
  leaving distTo[w] <= distTo[v] + e.weight()
* Inequality holds until algorithm terminates because:
  + distTo[w] cannot increase (because distTo[] values are monotone decreasing
  + distTo[v] cannot change (because of topological order, no edge pointing to v will be relaxed after v is relaxed
* Thus, upon termination, shortest-paths optimality conditions hold

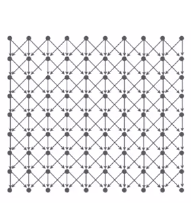
Implementation

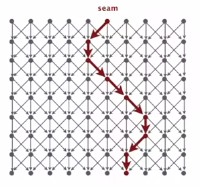


Application: Content-aware resizing

Seam carving: resize an image without distortion for display on cell phones and web-browsers

To find a vertical seam:

* Grid DAG: vertex = pixel; edge = from pixel to 3 downward neighbors
* Weight of pixel = energy function of 8 neighboring pixels
* Seam = shortest path from top to bottom

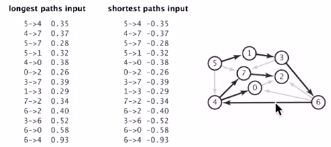


To remove a vertical seam:

* Delete pixels on the seam (one per row)

Longest paths in edge-weighted DAGs

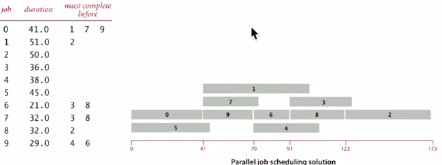
Formulate as a shortest paths problem in edge-weighted DAGs

* Negate all weights
* Find shortest paths
* Negate weights in result  
  **
* *Equivalent: reverse sense of equality in relax()*

Key point: topological sort algorithm works even with negative weights

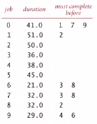
Application of longest paths in DAGs

Parallel job scheduling: given a set of jobs with durations and precedence constraints, schedule the jobs (by finding a start time for each) so as to achieve the minimum completion time, while respecting the constraints

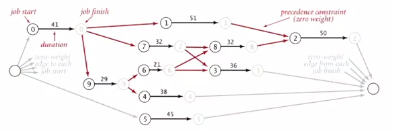


Critical path method

CPM: to solve a parallel job-scheduling problem, create edge-weighted DAG:



* Source and link vertices
* Two vertices (begin and end) for each job
* Three edges for each job
  + Begin to end (weighted by duration)
  + Source to begin (0 weight)
  + End to sink (0 weight)
* One edge for each precedence constraint (0 weight)



Use the longest path from the source to schedule each job: