算法设计与分析
Algorithms Design & Analysis

第十二讲：单源最短路径

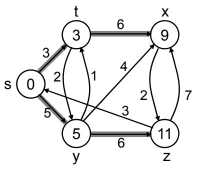
最短路径问题(Shortest Path Problems)

- How can we find the shortest route between two points on a map?(如何确定地图上两个地点的最短路径?)
- Model the problem as a graph problem(模型:图的算法问题):
  - Road map is a weighted graph: (路线地图→有权图)
    - vertices = cities (城市→顶点)
    - edges = road segments between cities(路线→边)
    - edge weights = road distances(路程→边的权)
  - Goal: find a shortest path between two vertices (cities) (目标:确定两顶点之间的最短路径)

最短路径问题(Shortest Path Problems)

- **Input:(输入)**
  - Directed graph G = (V, E) (有向图G)
  - Weight function w : E → **R (权)**
- **Weight of path** p = $\langle v_0, v_1, \ldots, v_k \rangle$ (路径权的定义)

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$



- **Shortest-path weight** from u to v: (最短路径的权)

$$\delta(u, v) = \begin{cases} \min\ w(p) : u \overset{p}{\leadsto} v & \text{if there exists a path from u to v} \\ \infty & \text{otherwise} \end{cases}$$

- Shortest path u to v is any path p such that w(p) = δ(u, v)

(u 到 v 的最短路径是满足w(p) = δ(u, v)的路径，不唯一)

最短路径的不同形式(Variants of Shortest Paths)

- **Single-source shortest path(单源最短路径)**
  - G = (V, E) ⇒ find a shortest path from a given source vertex *s* to each vertex v ∈ V (从给定的源顶点s到图中其他顶点v ∈ V的最短路径)
- **Single-destination shortest path(单目的地最短路径)**
  - Find a shortest path to a given destination vertex **t** from each vertex v (图中的顶点v ∈ V到某个给定的目的顶点t的最短路径)
  - Reverse the direction of each edge ⇒ single-source (反向图，演变为单源最短路径问题)
- **Single-pair shortest path（单点对最短路径问题）**
  - Find a shortest path from u to v for given vertices u and v （给定两顶点u和 v，求u 到 v的最短路径）
  - Solve the single-source problem （是单源最短路径问题的一个子问题）
- **All-pairs shortest-paths （全对点最短路径问题）**
  - Find a shortest path from u to v for every pair of vertices u and v（图中所有点对之间的最短距离）

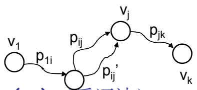
最短路径的优化基础（Optimal Substructure of Shortest Paths）

Given:给定
- A weighted, directed graph G = (V, E) （有权有向图）
- A weight function w: E → R, （权函数）
- A shortest path p = $\langle v_1, v_2, \ldots, v_k \rangle$ from v_1 to v_k （ v_1到v_k的最短路径p ）
- A subpath of p: $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$, with $1 \le i \le j \le k$ （ p的部分路径p_{ij} ）

Then: p_{ij} is a shortest path from v_i to v_j （部分路径是最短路径）

Proof: $p = v_1 \overset{p_{1i}}{\leadsto} v_i \overset{p_{ij}}{\leadsto} v_j \overset{p_{jk}}{\leadsto} v_k$

$w(p) = w(p_{1i}) + w(p_{ij}) + w(p_{jk})$



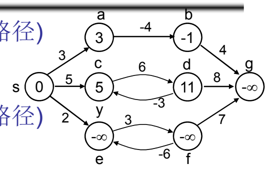
Assume ∃ p_{ij}' from v_i to v_j with $w(p_{ij}') < w(p_{ij})$ （反证法）

⇒ $w(p') = w(p_{1i}) + w(p_{ij}') + w(p_{jk}) < w(p)$ contradiction!

权值为负的边(Negative-Weight Edges)

- s → a: only one path (唯一路径)

δ(s, a) = w(s, a) = 3



- s → b: only one path (唯一路径)

δ(s, b) = w(s, a) + w(a, b) = -1

- s → c: infinitely many paths (无穷路径)

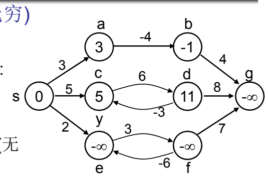
⟨s, c⟩, ⟨s, c, d, c⟩, ⟨s, c, d, c, d, c⟩

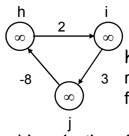
cycle has positive weight (6 - 3 = 3) (回路的权>0)

⟨s, c⟩ is shortest path with weight δ(s, c) = w(s, c) = 5 (最短路径)

权值为负的边(Negative-Weight Edges)

- s → e: infinitely many paths:(无穷)
  - ⟨s, e⟩, ⟨s, e, f, e⟩, ⟨s, e, f, e, f, e⟩
  - cycle ⟨e, f, e⟩ has negative weight:
    3 + (- 6) = -3 (回路的权<0)
  - can find paths from *s* to *e* with arbitrarily large negative weights (无穷大负权)
  - δ(s, e) = - ∞ ⇒ no shortest path exists between *s* and *e* (不存在最短路径)
  - Similarly: δ(s, f) = - ∞,
    δ(s, g) = - ∞



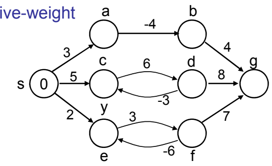


h, i, j not reachable from s

δ(*s*, h) = δ(*s*, i) = δ(*s*, j) = ∞

权值为负的边(Negative-Weight Edges)

- Negative-weight edges may form negative-weight cycles(负权边可能形成负权回路)
- If such cycles are reachable from the source: δ(s, v) is not properly defined



(如果从源顶点s能够抵达负权回路的顶点v,则有: w(s, v) = - ∞)

- Keep going around the cycle, and get
  w(s, v) = - ∞ for all v on the cycle

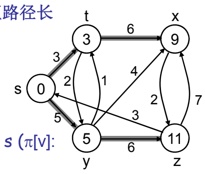
回路(Cycles)

- Can shortest paths contain cycles? (最短路径可否含有回路)
- Negative-weight cycles(负权回路) No!
- Positive-weight cycles:(正权回路) No!
  - By removing the cycle we can get a shorter path (如有移除，将产生更短的路径)
- Zero-weight cycles(权为0的回路)
  - No reason to use them (没有合理的理由在最短路径中包含它们)
  - Can remove them to obtain a path with similar weight(移除不会对最短路径的权产生任何影响)
- We will assume that when we are finding shortest paths, the paths will have no cycles(所以,最短路径不含有任何回路)

最短路径表示法(Shortest-Path Representation)

For each vertex v ∊ V:(对于任何一个顶点v ∊ V)

- d[v] = δ(s, v): a **shortest-path estimate (对最短路径长度的估计d[v] = δ(s, v):)**
  - Initially, d[v]=∞ (初值)
  - Reduces as algorithms progress
    (算法过程中逐步减少)



- π[v] = **predecessor** of v on a shortest path from *s* (π[v]: 从源顶点s到顶点v的最短路径上v的前辈顶点)
  - If no predecessor, π[v] = NIL (如果没有前辈顶点, π[v] = NIL)
  - π induces a tree—**shortest-path tree (形成树)**
- Shortest paths & shortest path trees are not unique (最短路径和最短路径树不是唯一的)

初始化算法(Initialization)

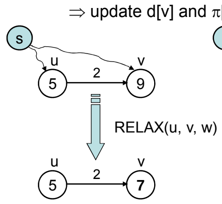
Alg.: INITIALIZE-SINGLE-SOURCE(V, s)

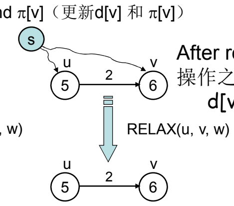
1. **for each** v ∊ V
2. **do** d[v] ← ∞
3. π[v] ← NIL
4. d[s] ← 0

- All the shortest-paths algorithms start with INITIALIZE-SINGLE-SOURCE (所有的最短路径算法都以初始化开始)

缩短法(Relaxation)

- **Relaxing** an edge (u, v) = testing whether we can improve the shortest path to v found so far by going through u (所谓对边(u, v)的缩短，即是检查能否通过顶点u，改善已有的到达v的最短路径)
  If d[v] > d[u] + w(u, v) （如果满足 d[v] > d[u] + w(u, v)）
  we can improve the shortest path to v （则可以改进抵达v的最短路径）
  ⇒ update d[v] and π[v] （更新d[v] 和 π[v]）





After relaxation:（缩短操作之后满足：）
d[v] ≤ d[u] + w(u, v)

RELAX算法（RELAX(u, v, w)）

1. **if** d[v] > d[u] + w(u, v)
2. **then** d[v] ← d[u] + w(u, v)
3. π[v] ← u

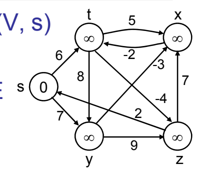
- All the single-source shortest-paths algorithms （所有单源最短路径算法）
  - start by calling INIT-SINGLE-SOURCE（从初始化算法开始）
  - then relax edges（然后是缩短算法）
- The algorithms differ in the order and how many times they relax each edge （算法之间的差别在于缩短算法的执行顺序和次数）

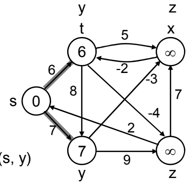
Bellman-Ford 算法

- Single-source shortest paths problem（单源最短路径问题）
  - Computes d[v] and π[v] for all v ∈ V （计算顶点v ∈ V 的d[v] 和 π[v] ）
- Allows negative edge weights（允许负权值边）
- Returns: （返回值）
  - TRUE if no negative-weight cycles are reachable from the source s （如果从源顶点s没有可抵达的负权值回路，返回'真'）
  - FALSE otherwise ⇒ no solution exists （其余的返回'假'，无解）
- Idea: （思想）
  - Traverse all the edges |V – 1| times, every time performing a relaxation step of each edge（遍历所有的边|V – 1|次，每次对每条边执行一次缩短运算）

BELLMAN-FORD(V, E, w, s)

1. INITIALIZE-SINGLE-SOURCE(V, s)
2. **for** i ← 1 to |V| - 1
3. **do for** each edge (u, v) ∈ E
4. **do** RELAX(u, v, w)
5. **for** each edge (u, v) ∈ E
6. **do if** d[v] > d[u] + w(u, v)
7. **then return** FALSE
8. **return** TRUE



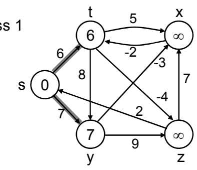


E: (t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)

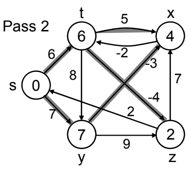
|V| - 1次边搜索的顺序，

(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)

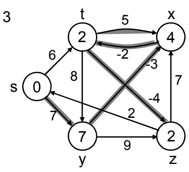
Pass 1



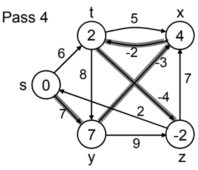
Pass 2



Pass 3

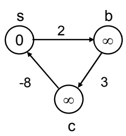


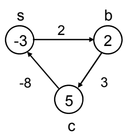
Pass 4

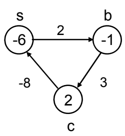


回路检测(Detecting Negative Cycles)

- **for** each edge (u, v) ∈ E
- **do if** d[v] > d[u] + w(u, v)
- **then return** FALSE
- **return** TRUE







执行|V|-1次缩短运算

Look at edge (s, b):(对于边(s, b))

d[b] = -1
d[s] + w(s, b) = -4

⇒ d[b] > d[s] + w(s, b)

分析:BELLMAN-FORD(V, E, w, s)

1. INITIALIZE-SINGLE-SOURCE(V, s) ⟵ Θ(V)
2. **for** i ← 1 to |V| - 1 ⟵ O(V)
3. **do for** each edge (u, v) ∈ E ⟵ O(E)
4. **do** RELAX(u, v, w)
5. **for** each edge (u, v) ∈ E ⟵ O(E)
6. **do if** d[v] > d[u] + w(u, v)
7. **then return** FALSE
8. **return** TRUE

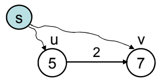
Running time: O(VE) (运行开销)

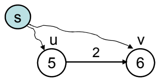
最短路径特性(Shortest Path Properties)

- **Triangle inequality (三角不等式)**

  For all (u, v) ∈ E, we have:

  $\delta(s, v) \le \delta(s, u) + w(u, v)$





- If u is on the shortest path to v we have the equality sign (如果顶点u在抵达v的最短路径上，等号成立）

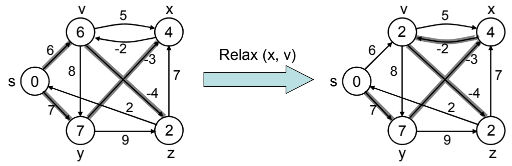
最短路径特性(Shortest Path Properties)

- **Upper-bound property(上限特性)**

  We always have d[v] ≥ δ(s, v) for all v.(对于任何顶点v， d[v] ≥ δ(s, v)成立)

  Once d[v] = δ(s, v), it never changes. (一旦d[v] = δ(s, v)成立, d[v] 便不再改变)

  – The estimate never goes up – relaxation only lowers the estimate (对距离d[v]的估计不会变大，缩短操作只会缩小距离d[v] 的估计）



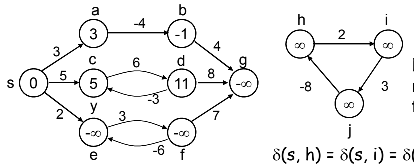
最短路径特性(Shortest Path Properties)

- **No-path property(无通路特性)**

  If there is no path from *s* to v then d[v] = ∞ always.

  如果不存在从*s*到v的通路，则有d[v] = ∞

  – δ(s, h) = ∞ and d[h] ≥ δ(s, h) ⇒ d[h] = ∞ （例）



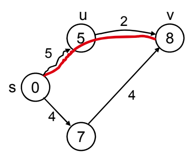
$\delta(s, h) = \delta(s, i) = \delta(s, j) = \infty$

最短路径特性(Shortest Path Properties)

- **Convergence property**

  If s⇝u → v is a shortest path, and if d[u] = δ(s, u) at any time prior to relaxing edge (u, v), then d[v] = δ(s, v) at all times afterward.

  (如果s⇝u → v 是一条最短路径，在对边(u, v)进行缩短操作之前的任何时刻有d[u] = δ(s, u), 那么在对边(u, v)进行缩短操作之后的任何时刻有d[v] = δ(s, v) 。)

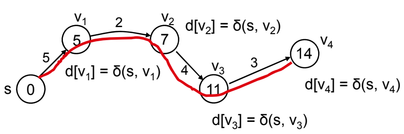


- If d[v] > δ(s, v) ⇒ after relaxation:
  d[v] = d[u] + w(u, v)
  d[v] = 5 + 2 = 7
- Otherwise, the value remains unchanged, because it must have been the shortest path value(保持不变，因为它是最短路径）

最短路径特性(Shortest Path Properties)

- **Path relaxation property**

  Let p = ⟨v_0, v_1, . . . , v_k⟩ be a shortest path from s = v_0 to v_k. If we relax, in order, (v_0, v_1), (v_1, v_2), . . . , (v_{k-1}, v_k), even intermixed with other relaxations, then d[v_k] = δ(s, v_k). (p = ⟨v_0, v_1, . . . , v_k⟩ 是从源顶点v_0到v_k的最短路径，如果缩短操作是按照(v_0, v_1), (v_1, v_2), . . . , (v_{k-1}, v_k)进行的，即使其中有其他缩短操作穿插， d[v_k] = δ(s, v_k).成立）



DAG的单源最短路径(Single-Source Shortest Paths in DAGs)

- Given a weighted DAG: G = (V, E)

  – solve the shortest path problem

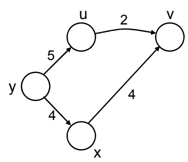
  (给出DAG: G = (V, E) , 求最短路径)

- Idea: (思想)

  – Topologically sort the vertices of the graph (对图进行拓扑排序)

  – Relax the edges according to the order given by the topological sort (依据拓扑排序对边进行缩短操作)

    - for each vertex, we relax each edge that starts from that vertex (对于每一个顶点, 对始于该顶点的每条边进行缩短操作)



- Are shortest-paths well defined in a DAG?

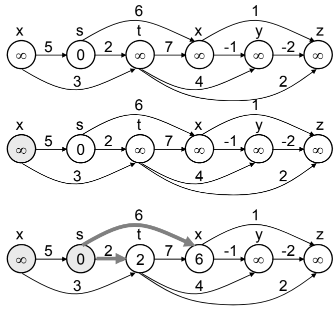
  – Yes, (negative-weight) cycles cannot exist (DGA中没有负权值回路, 因此存在最短路径)

DAG-SHORTEST-PATHS(G, w, s)

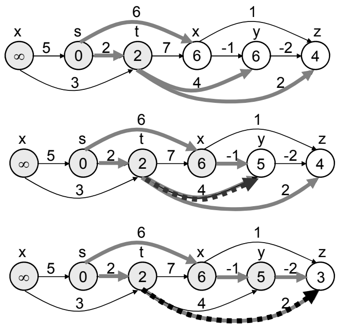
1. topologically sort the vertices of G(拓扑排序) — Θ(V+E)
2. INITIALIZE-SINGLE-SOURCE(V, *s*)(初始化) — Θ(V)
3. **for** each vertex u, taken in topologically sorted order(依据拓扑排序顶点顺序) — Θ(V)
4. **do for** each vertex v ∈ Adj[u] (对邻接边进行缩短操作) — Θ(E)
5. **do** RELAX(u, v, w)

Running time: Θ(V+E)

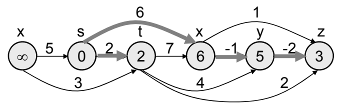
Example



Example (cont.)



Example (cont.)

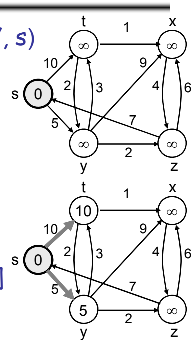


Dijkstra's 算法

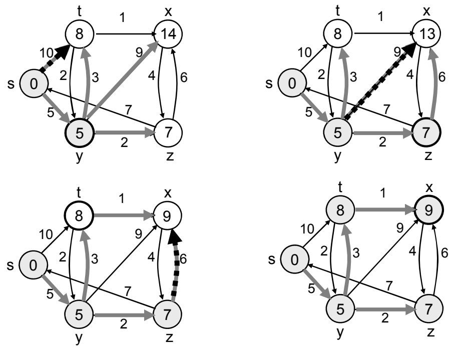
- Single-source shortest path problem: (单源最短路径)
  - No negative-weight edges: w(u, v) > 0 ∀ (u, v) ∈ E (不存在负权值边界)
- Maintains two sets of vertices: (两类顶点的集合)
  - S = vertices whose final shortest-path weights have already been determined (S: 集合中顶点的最短路径已经确定)
  - Q = vertices in V – S: min-priority queue (Q: V – S, 极小优先队列)
    - Keys in Q are estimates of shortest-path weights (d[v]) (Q中的值是最短路径的估计)
- Repeatedly select a vertex u ∈ V – S, with the minimum shortest-path estimate d[v] (重复的从Q中选择具有最短估计距离的顶点进行处理)

Dijkstra (G, w, s)

1. INITIALIZE-SINGLE-SOURCE(V, *s*)
2. S ← ∅
3. Q ← V[G]
4. **while** Q ≠ ∅
5. **do** u ← EXTRACT-MIN(Q)
6. S ← S ∪ {u}
7. **for** each vertex v ∈ Adj[u]
8. **do** RELAX(u, v, w)



Example



31