> Single-Source Shortest Path 单源最短路径
> All-Pairs Shortest Path 结点对最短路径

SSSP--The Shortest Path Problem

- 权不仅是长度
- 可以有向图
- 允许负边，不允许负环(总权和为负)

> 事实上最短路径不会有环：否则无穷&&零环可删

松弛操作

- u.d表示从s到u的权重上界

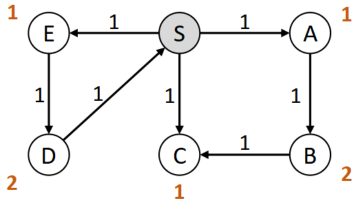
```
RELAX(u,v)
if v.d>u.d+w(u,v)
    v.d=u.d+w(u,v)
    v.parent=u
```

```
UPDATE(u,v)
    v.d=min{v.d,u.d+w(u,v)}
```

Case 1: Unit weight

> 有向，无向

- BFS



Case 2: Arbitrary positive weight

> 有向，无向

unit算法：

- 在权大的边上添加结点，变成unit问题

> 权差异大时太慢

- 三角不等式：v.d<=u.d+w(u,v)
- 递推改进：对于已知的u和（u，v），Tv=min{Tv,Tu+w(u,v)}

Dijkstra's 算法

> 贪心：类似prim，每次加入一个权最小边
>
> 维持一组已找完的结点，集合已知区域R
>
> 每次与R直接相连的.d被更新，并找出最小

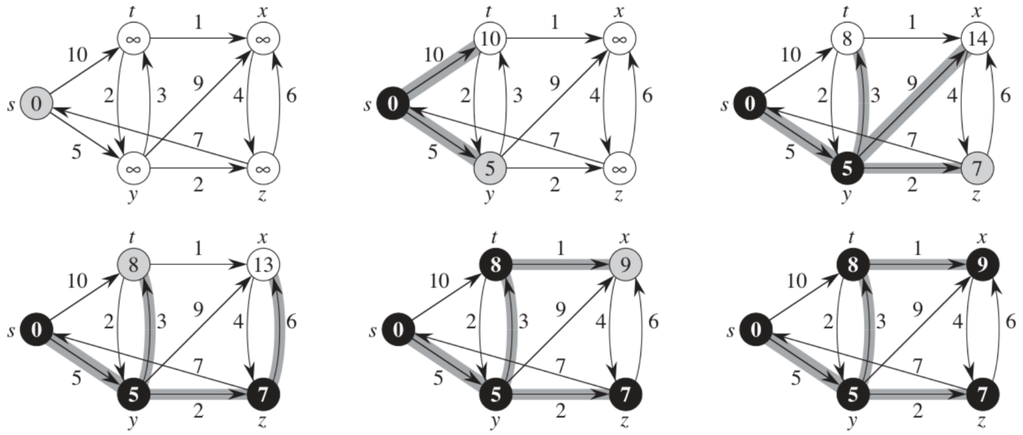
- 更新添加点v

> 与prim的更新算法不同，prim侧重相邻最小
>
> SSSP侧重从源点s出发最小

```
u∈R,v∈V-R
find min{dist(s,u)+w(u,v)}
```

```
DijkstraSSSP(G,s):
for (each u in V)
  u.d=INF, u.parent=NIL
s.d = 0
Build priority queue Q based on dist
while (!Q.empty())
  u = Q.ExtractMin()//O(nlgn)
  for (each edge (u,v) in E)//O(mlgn)
     RELAX(u,v)
    Q.DecreaseKey(v)
```

- O（(m+n)lgn)



DFS, BFS, Prim, Dijkstra, and others…

```
DFSIterSkeleton(G,s):
Stack Q
Q.push(s)
while (!Q.empty())
  u = Q.pop()
  if (!u.visited)
    u.visited = true
    for (each edge (u,v) in E)
      Q.push(v)
```

```
BFSSkeletonAlt(G,s):
FIFOQueue Q
Q.enque(s)
while (!Q.empty())
  u = Q.dequeue()
  if (!u.visited)
    u.visited = true
    for (each edge (u,v) in E)
      Q.enque(v)
```

```
PrimMSTSkeleton(G,x):
PriorityQueue Q
Q.add(x)
while (!Q.empty())
  u = Q.remove()
  if (!u.visited)
    u.visited = true
    for (each edge (u,v) in E)
      if (!v.visited and …)
        Q.update(v,…)
```

```
DijkstraSSSPSkeleton(G,x):
PriorityQueue Q
Q.add(x)
while (!Q.empty())
  u = Q.remove()
  if (!u.visited)
    u.visited = true
    for (each edge (u,v) in E)
      if (!v.visited and …)
        Q.update(v,…)
```

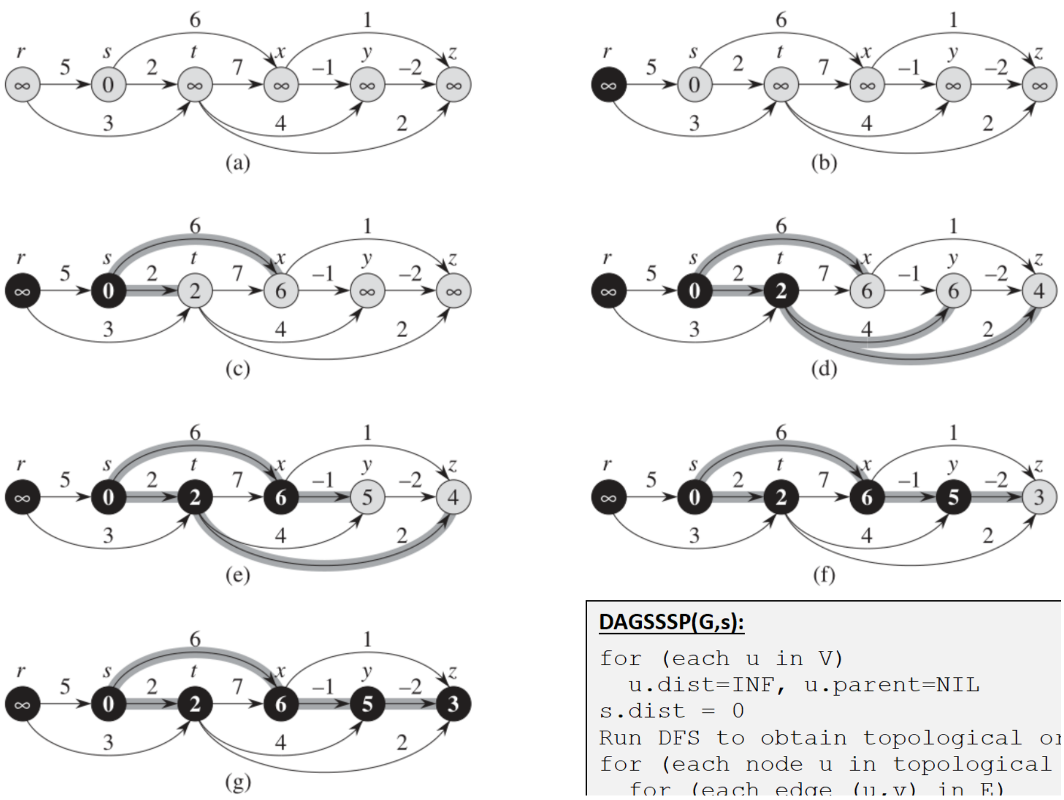
Case 3: without cycle

> 有向
> 可以用Bellman但太慢了

- 拓扑排序O（n+m)

```
DAGSSSP(G,s):
for (each u in V)
  u.dist=INF, u.parent=NIL
s.dist = 0
```

```
Run DFS to obtain topological order
for (each node u in topological order)
  for (each edge (u,v) in E)
    RELAX(u,v)
```


DAGSSSP(G,s):

```
for (each u in V)
  u.dist=INF, u.parent=NIL
s.dist = 0
Run DFS to obtain topological or
for (each node u in topological
  for (each edge (u,v) in E)
```

Case 4: negative weights

> 有向
> Dijkstra失效，因为v的最短路径未必经过已知区域

Bellman-Ford算法

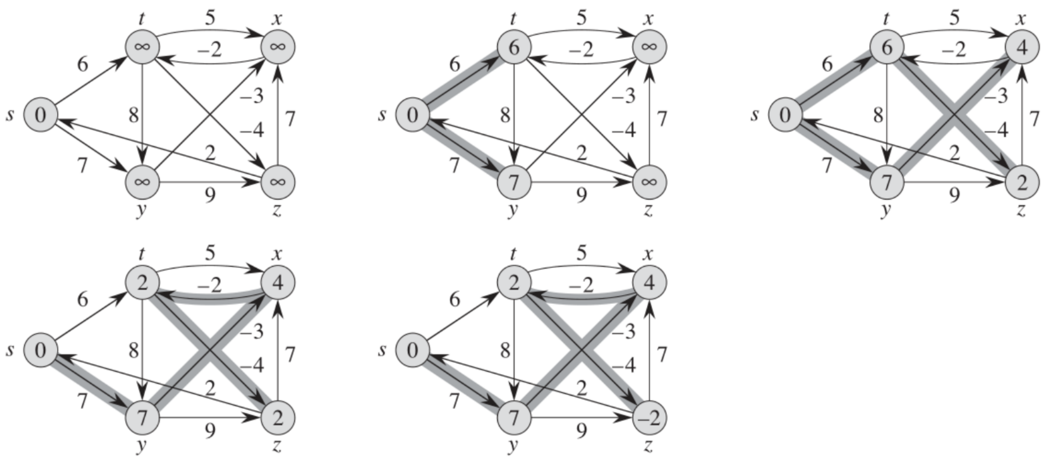
> 可以处理负边，但有点慢
> update所有边，渐近降低s到每个u的估计值
> 重复n-1次，直到所有估计值达到实际值

- 可以顺便检测负圈

> 有负圈则无解，是负无穷

```
BellmanFordSSSP(G,s):
for (each u in V)
  u.dist=INF, u.parent=NIL
s.dist = 0
repeat n-1 times:
  for (each edge (u,v) in E)
    RELAX(u,v)
//负圈检测
for (each edge (u,v) in E)
  if (v.d > u.d + w(u,v))
    return “Negative Cycle”
```

- O(nm)



APSP--All-Pairs Shortest Path

> n次SSSP太慢

Johnson算法

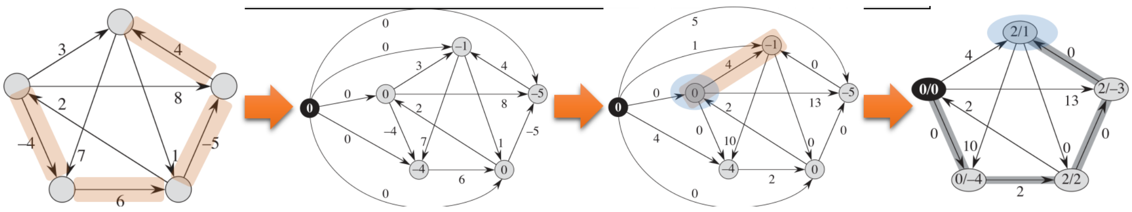
- 思路：改变weights而不是最短路径

> 使u，v的所有路径权和改变相同

```
w'(u,v)=h(u)+w(u,v)-h(v)
w'(u->v)=h(u)+w(u->v)-h(v)
```

> 在无向图中，可以令h（u）=dist（z，u）
> 在有向图中，加结点z和w（z，v）=0，使z直接和所有点相连

- 每个点，每个点按最短路径赋值
-



```
JohnsonAPSP(G):
Create H=(V+{z},E+{(z,v)|v∈V}) with w(z,v)=0
Bellman-FordSSSP(H,z) to obtain distH
for (each edge (u,v) in H.E)
  w’(u,v) = distH(z,u)+w(u,v)-distH(z,v)
for (each node u in G.V)
  DijkstraSSSP(G,u) with w’ to obtain distG,w’
  for (each node v in G.V)
    distG(u,v) = distG,w’(u,v)+distH(z,v)-distH(z,u)
```

- 包含了Dijkstra和Bellmman，O（nnnlgn）

Floyd-Warshall算法

- 遇到环可能出不来，因此限定步数

$$dist(u,v) = \begin{cases} 0 & \text{if } u = v \\ min_{(x,v)\in E}\{dist(u,x) + w(x,v)\} & \text{otherwise} \end{cases}$$

- 不超过l条边的路径dist(u,v,l)

迭代1--l-1

$$dist(u,v,l) = \begin{cases} 0 & \text{if } l = 0 \text{ and } u = v \\ \infty & \text{if } l = 0 \text{ and } u \neq v \\ \min\left\{ \begin{array}{c} dist(u,v,l-1) \\ \min\limits_{(x,v)\in E}\{dist(u,x,l-1) + w(x,v)\} \end{array} \right\} & \text{otherwise} \end{cases}$$

```
RecursiveAPSP(G):
for (every pair (u,v) in V*V)
  if (u=v) then dist[u,v,0]=0
```

```
  else dist[u,v,0]=INF
for (l=1 to n-1)
  for (each node u)
    for (each node v)
      dist[u,v,l] = dist[u,v,l-1]
      for (each edge (x,v) going to v)
        if (dist[u,v,l] > dist[u,x,l-1] + w(x,v))
          dist[u,v,l] = dist[u,x,l-1]+w(x,v)
```

- O (n^4)

迭代l/2--l/2

$$dist(u,v,l)=\begin{cases} w(u,v) & \text{if } l=1 \text{ and } (u,v)\in E \\ \infty & \text{if } l=1 \text{ and } (u,v)\notin E \\ \min_{x\in V}\{dist(u,x,l/2)+dist(x,v,l/2)\} & \text{otherwise} \end{cases}$$

```
FasterRecursiveAPSP(G):
for (every pair (u,v) in V*V)
  if ((u,v) in E) then dist[u,v,1]=w(u,v)
  else dist[u,v,1]=INF
for (i=1 to Ceil(lg(n)))//2^[lgn]
  for (each node u)
    for (each node v)
      dist[u,v,i] = INF
      for (each node x)
        if (dist[u,v,i] > dist[u,x,i-1] + dist[x,v,i-1])
          dist[u,v,i] = dist[u,x,i-1] + dist[x,v,i-1]
```

- O(n^3lgn)

利用上一次

$$dist(u,v,r)=\begin{cases} w(u,v) & \text{if } r=0 \text{ and } (u,v)\in E \\ \infty & \text{if } r=0 \text{ and } (u,v)\notin E \\ \min\begin{Bmatrix} dist(u,v,r-1) \\ dist(u,x_r,r-1)+dist(x_r,v,r-1) \end{Bmatrix} & \text{otherwise} \end{cases}$$

```
FloydWarshallAPSP(G):
for (every pair (u,v) in V*V)
  if ((u,v) in E) then dist[u,v,0]=w(u,v)
  else dist[u,v,0]=INF
for (r=1 to n)
  for (each node u)
    for (each node v)
      dist[u,v,r] = dist[u,v,r-1]
      if (dist[u,v,r] > dist[u,xr,r-1] + dist[xr,v,r-1])
        dist[u,v,r] = dist[u,xr,r-1] + dist[xr,v,r-1]
```

- O(n^3)

变形--判断路径

```
FloydWarshallTransitiveClosure(G):
for (every pair (u,v) in V*V)
  if ((u,v) in E) then t[u,v,0] = TRUE
  else t[u,v,0] = FALSE
for (r=1 to n)
  for (each node u)
    for (each node v)
      t[u,v,r] = t[u,v,r-1]
      if (t[u,xr,r-1] AND t[xr,v,r-1])
        t[u,v,r] = TRUE
```