第七讲 图（中）

浙江大学 陈 越

7.1 最短路径问题

嘉定西站
白银路站
嘉定新城站
上海汽车城站
上海赛车场站
马陆站
南翔站
桃浦新村站
武威路站
祁连山路站
李子园站
上海西站站
真如站
枫桥路站
轨道交通11号线
轨道交通7号线
上大路站
场中路站
大场镇站
行知路站
大华三路站
新村路站
岚皋路站
镇坪路站
中潭路站
上海火车站
曹杨路站
金沙江路站
轨道交通3号线
虹桥2号航站楼站
淞虹路站
北新泾站
威宁路站
娄山关路站
中山公园站
隆德路站
江苏路站
长寿路站
昌平路站
静安寺站
虹桥国际机场
延安西路站
虹桥路站
交通大学站
上海图书馆站
常熟路站
陕西南路站
黄陂南路站
紫藤路站
龙柏新村站
龙溪路站
水城路站
伊犁路站
宋园路站
衡山路站
徐家汇站
上海体育馆站
宜山路站
轨道交通9号线
泗泾站
九亭站
中春路站
七宝站
星中路站
合川路站
漕河泾开发区站
桂林路站
上海体育场站
漕宝路站
上海南站站
漕溪路站
龙漕路站
石龙路站
锦江乐园站
莲花路站
外环路站
莘庄站
上海南站
春申路站
银都路站
轨道交通5号线
东安路站
肇嘉浜路站
嘉善路站
打浦桥站
鲁班路站
马当路站
新天地站
船厂路站
大木桥路站
卢浦大桥站
后滩站
世博大道站
长清路站
耀华路站
成山路站
云台路站
轨道交通8号线
杨思站
凌兆新村站
芦恒路站
友谊西路站
宝安公路站
共富新村站
呼兰路站
通河新村站
共康路站
彭浦新村站
汶水路站
上海马戏城站
延长路站
上海火车站站
中山北路站
中兴路站
新闸路站
汉中路站
南京西路站
人民广场站
大世界站
豫园站
老西门站
陆家浜路站
西藏南路站
南浦大桥站
世博园区轨道交通专用线
小南门站
世博园区
黄浦江
西藏北路站
虹口足球场站
曲阳路站
东宝兴路站
宝山路站
曲阜路站
天潼路站
四川北路站
海伦路站
邮电新村站
南京东路站
宝杨路站
水产路站
淞滨路站
张华浜站
淞发路站
长江南路站
殷高西路站
江湾镇站
大柏树站
赤峰路站
轨道交通10号线
新江湾城站
殷高东路站
三门路站
江湾体育场站
五角场站
国权路站
同济大学站
四平路站
鞍山新村站
江浦路站
黄兴路站
市光路站
嫩江路站
翔殷路站
黄兴公园站
延吉中路站
外高桥
轨道交通4号线
临平路站
大连路站
杨树浦路站
浦东大道站
东昌路站
陆家嘴站
轨道交通6号线
源深体育中心站
民生路站
北洋泾路站
德平路站
云山路站
金桥路站
世纪大道站
商城路站
浦电路站
杨高中路站
上海科技馆站
浦电路站
世纪公园站
塘桥站
蓝村路站
上海儿童医学中心站
临沂新村站
高科西路站
东明路站
高青路站
华夏西路站
杨高南路站
锦绣路站
芳华路站
龙阳路站
花木路站
张江高科站
金科路站
广兰路站
磁浮
灵岩
上南路

最短路径问题的抽象

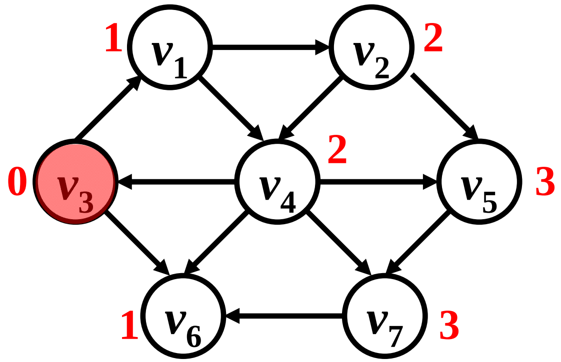
- 在网络中，求两个不同顶点之间的所有路径中，边的权值之和最小的那一条路径
  - 这条路径就是两点之间的最短路径（**Shortest Path**）
  - 第一个顶点为源点（**Source**）
  - 最后一个顶点为终点（**Destination**）

问题分类

- 单源最短路径问题：从某固定源点出发，求其到所有其他顶点的最短路径
  - （有向）无权图
  - （有向）有权图
- 多源最短路径问题：求任意两顶点间的最短路径

无权图的单源最短路算法

- 按照递增（非递减）的顺序找出到各个顶点的最短路



0: ☞ v_3

1: ☞ v_1 and v_6

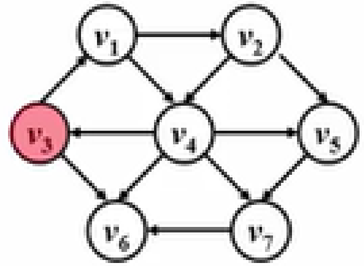
2: ☞ v_2 and v_4

3: ☞ v_5 and v_7

BFS！

James Bond 从孤岛跳上岸，最少需要跳多少步？

无权图的单源最短路算法




```
void BFS ( Vertex S )
{ visited[S] = true;
  Enqueue(S, Q);
  while(!IsEmpty(Q)){
    V = Dequeue(Q);
    for ( V 的每个邻接点 W )
      if ( !visited[W] ) {
        visited[W] = true;
        Enqueue(W, Q);
      }
  }
}
```

```
void Unweighted ( Vertex S )
{ Enqueue(S, Q);
  while(!IsEmpty(Q)){
    V = Dequeue(Q);
    for ( V 的每个邻接点 W )
      if ( dist[W]==-1  ) {
        dist[W] = dist[V]+1;
        path[W] = V;
        Enqueue(W, Q);
      }
  }
}
```

$T = O(|V| + |E|)$

初始化两个数组

下标	1	2	3	4	5	6	7
dist	-1	-1	-1	-1	-1	-1	-1
path	-1	-1	-1	-1	-1	-1	-1

dist[W] = S到W的最短距离

dist[S] = 0

path[W] = S到W的路上经过的某顶点

Unweighted(3)

下标	1	2	3	4	5	6	7
dist	-1	-1	0	-1	-1	-1	-1
path	-1	-1	-1	-1	-1	-1	-1

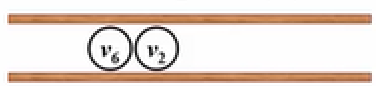
Unweighted(3)

下标	1	2	3	4	5	6	7
dist	1	-1	0	-1	-1	-1	-1
path	3	-1	-1	-1	-1	-1	-1

Unweighted(3)

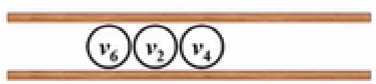
下标	1	2	3	4	5	6	7
dist	1	-1	0	-1	-1	1	-1
path	3	-1	-1	-1	-1	3	-1

Unweighted(3)



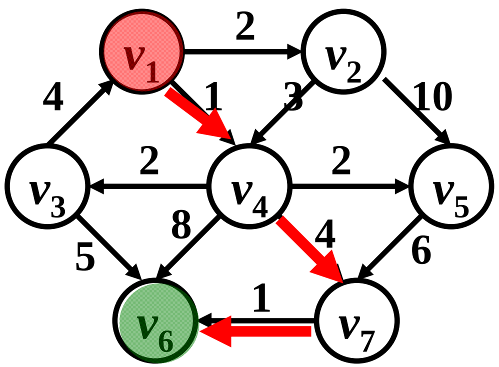
下标	1	2	3	4	5	6	7
dist	1	2	0	2	-1	1	-1
path	3	1	-1	-1	-1	3	-1

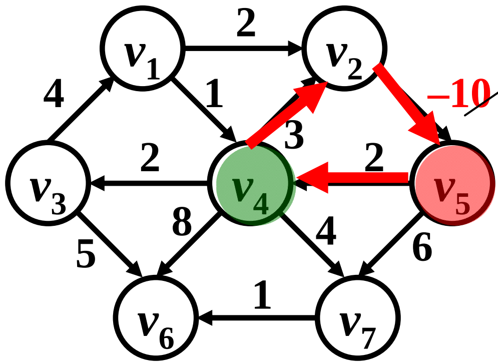
Unweighted(3)



下标	1	2	3	4	5	6	7
dist	1	2	0	2	-1	1	-1
path	3	1	-1	1	-1	3	-1

有权图的单源最短路算法





负值圈 (negative-cost cycle)

- 按照递增的顺序找出到各个顶点的最短路

Dijkstra 算法

有权图的单源最短路算法

- **Dijkstra 算法**
  - 令S={源点s + 已经确定了最短路径的顶点v_i}
  - 对任一未收录的顶点v，定义dist[v]为s到v的最短路径长度，但该路径仅经过S中的顶点。即路径{s→(v_i∈S)→v}的最小长度
  - 若路径是按照递增（非递减）的顺序生成的，则
    - 真正的最短路必须只经过S中的顶点（为什么？）
    - 每次从未收录的顶点中选一个dist最小的收录（贪心）
    - 增加一个v进入S，可能影响另外一个w的dist值！
      - dist[w] = min{dist[w], dist[v] + <v,w>的权重}

有权图的单源最短路算法

```
void Dijkstra( Vertex s )
{ while (1) {
    V = 未收录顶点中dist最小者;
    if ( 这样的V不存在 )
      break;
    collected[V] = true;
    for ( V 的每个邻接点 W )
      if ( collected[W] == false )
       if ( dist[V]+E<V,W> < dist[W] ) {
         dist[W] = dist[V] + E<V,W> ;
         path[W] = V;
       }
  }
} /* 不能解决有负边的情况 */
```

$$T = O(\ ?\ )$$

Dijkstra算法中的dist应该如何初始化?

如果s到w有直接的边，则dist[w]=<s,w>的权重；否则dist[w]定义为

- A. 正无穷
- B. 负无穷
- C. -1
- D. 这三种都可以

有权图的单源最短路算法

- 方法1：直接扫描所有未收录顶点 – $O(|V|)$
  - $T = O(|V|^2 + |E|)$

对于稠密图效果好

- 方法2：将**dist**存在最小堆中 – $O(\log|V|)$
  - 更新**dist[w]**的值 – $O(\log|V|)$
  - $T = O(|V| \log|V| + |E| \log|V|) = O(|E| \log|V|)$

对于稀疏图效果好

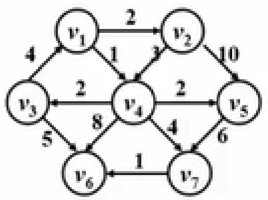
有权图的单源最短路算法

```
void Dijkstra( Vertex s )
{ while (1) {
    V = 未收录顶点中dist最小者;
    if ( 这样的V不存在 )
      break;
    collected[V] = true;
    for ( V 的每个邻接点 W )
      if ( collected[W] == false )
        if ( dist[V]+E<V,W> < dist[W] ) {
          dist[W] = dist[V] + E<V,W> ;
          path[W] = V;
        }
  }
}
```

多源最短路算法

- 方法1：直接将单源最短路算法调用|V|遍
  - $T = O(|V|^3 + |E|\times|V|)$ 对于稀疏图效果好
- 方法2： Floyd 算法
  - $T = O(|V|^3)$ 对于稠密图效果好

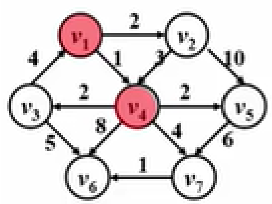
有权图的单源最短路算法



下标	1	2	3	4	5	6	7
dist	∞	∞	∞	∞	∞	∞	∞
path	-1	-1	-1	-1	-1	-1	-1

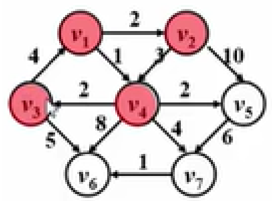
下标	1	2	3	4	5	6	7
dist	0	2	∞	∞	∞	∞	∞
path	-1	1	-1	-1	-1	-1	-1

下标	1	2	3	4	5	6	7
dist	0	2	∞	1	∞	∞	∞
path	-1	1	-1	1	-1	-1	-1



下标	1	2	3	4	5	6	7
dist	0	2	3	1	∞	∞	∞
path	-1	1	4	1	-1	-1	-1

下标	1	2	3	4	5	6	7
dist	0	2	3	1	3	∞	∞
path	-1	1	4	1	4	-1	-1



下标	1	2	3	4	5	6	7
dist	0	2	3	1	3	9	∞
path	-1	1	4	1	4	4	-1

下标	1	2	3	4	5	6	7
dist	0	2	3	1	3	9	5
path	-1	1	4	1	4	4	4

下标	1	2	3	4	5	6	7
dist	0	2	3	1	3	8	5
path	-1	1	4	1	4	3	4

多源最短路算法

参考youtube: https://www.youtube.com/watch?v=I1kA7c09IZE&ab_channel=Bleeptrack

- **Floyd 算法**
  - $D^k[i][j]$ = 路径$\{ i \to \{ l \le k \} \to j \}$的最小长度
  - $D^0, D^1, \ldots, D^{|V|-1}[i][j]$即给出了i到j的真正最短距离
  - 最初的D^{-1}是什么？
  - 当D^{k-1}已经完成，递推到D^k时：
    - 或者$k \notin$最短路径$\{ i \to \{ l \le k \} \to j \}$，则$D^k = D^{k-1}$
    - 或者$k \in$最短路径$\{ i \to \{ l \le k \} \to j \}$，则该路径必定由两段最短路径组成： $D^k[i][j]=D^{k-1}[i][k]+D^{k-1}[k][j]$

D矩阵应该初始化为什么？

A. 带权的邻接矩阵，对角元是0

如果i和j之间没有直接的边，D[i][j]应该定义为

A. 正无穷

多源最短路算法

```
void Floyd()
{  for ( i = 0; i < N; i++ )
     for( j = 0; j < N; j++ ) {
       D[i][j] = G[i][j];
       path[i][j] = -1;
     }
   for( k = 0; k < N; k++ )
     for( i = 0; i < N; i++ )
       for( j = 0; j < N; j++ )
         if( D[i][k] + D[k][j] < D[i][j] ) {
           D[i][j] = D[i][k] + D[k][j];
           path[i][j] = k;
         }
}
```

$T = O(|V|^3)$