哈爾濱工業大學 海量数据计算研究中心

Massive Data Computing Lab @ HIT

算法设计与分析—进阶篇

第三讲 图上的动态规划算法

哈尔滨工业大学
王宏志
wangzh@hit.edu.cn
http://homepage.hit.edu.cn/pages/wang/

本讲内容

3.1 最优二分搜索树

3.2 树的独立集合

3.3 任意两点最短路径问题

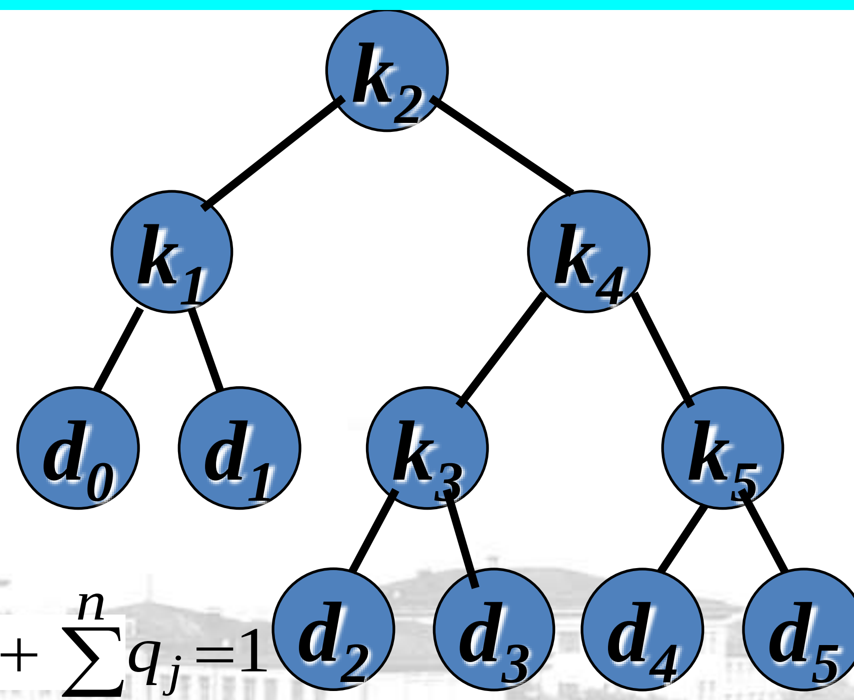
问题的定义

- 二叉搜索树T
  - 搜索树的期望代价

$$E(T)=\sum_{i=1}^{n}(DEP_T(k_i)+1)p_i+\sum_{j=0}^{n}(DEP_T(d_j)+1)q_j$$

d_n 对应区间 $(k_n,+\infty)$

  - 附加信息
    - 搜索k_i的概率为p_i
    - 搜索d_i的概率为q_i $\sum_{i=1}^{n}p_i+\sum_{j=0}^{n}q_j=1$



- 问题的定义

输入：$k=\{k_1, k_2, \ldots, k_n\}, k_1 < k_2 < \ldots < k_n$, $P=\{p_1, p_2, \ldots, p_n\}$, p_i为搜索k_i的概率 $Q=\{q_0, q_1, \ldots, q_n\}$, q_i为搜索值d_i的概率

输出：K的二叉搜索树T, $E(T)$最小

优化二叉搜索树结构的分析

- 优化子结构

  定理. 如果优化二叉搜索树T具有包含关键字集合$\{k_i, k_{i+1}, \ldots, k_j\}$子树$T'$, 则$T'$必是关于关键字集合$\{k_i, k_{i+1}, \ldots, k_j\}$子问题的优化解.

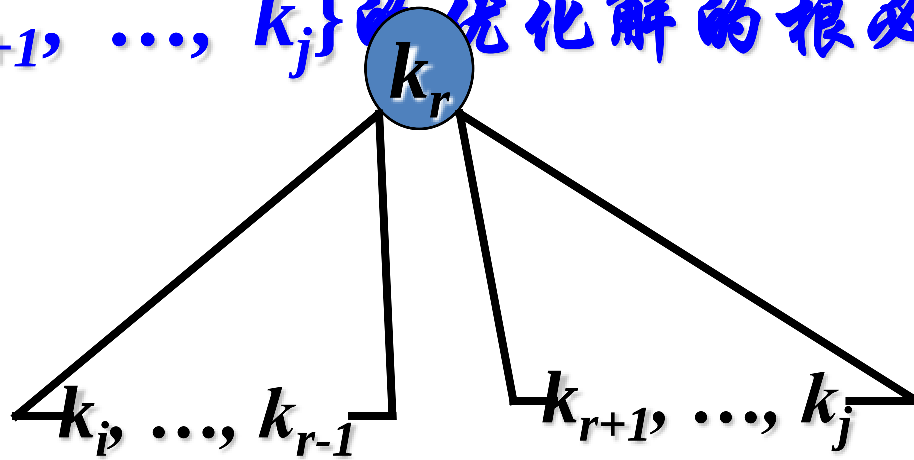
  证明: 若不然, 必有关键字集$\{k_i, k_{i+1}, \ldots, k_j\}$子树$T''$, T''的期望搜索代价低于T'.

  用T''替换T中的T', 可以得到一个期望搜索代价比T小的原始问题的二叉搜索树,

  与T是最优解矛盾.

- 用优化子结构从子问题优化解构造优化解

$K=\{k_i, k_{i+1}, \ldots, k_j\}$的优化解的根必为$K$中某个$k_r$

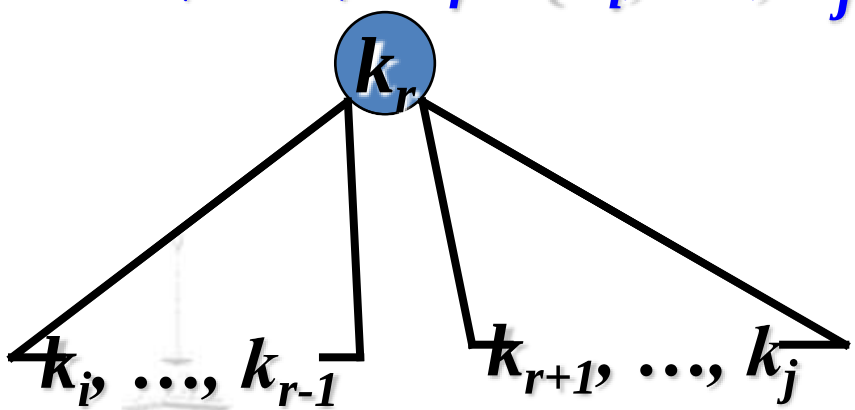


只要对于每个$k_r \in K$, 确定$\{k_i, \ldots, k_{r-1}\}$和$\{k_{r+1}, \ldots, k_j\}$的优化解, 我们就可以求出K的优化解.

如果$r=i$, 左子树$\{k_i, \ldots, k_{i-1}\}$仅包含d_{i-1}
如果$r=j$, 右子树$\{k_{r+1}, \ldots, k_j\}$仅包含d_j

建立优化解的搜索代价递归方程

- 令$E(i, j)$为$\{k_i, \ldots, k_j\}$的优化解T_{ij}的期望搜索代价
  - 当$j=i-1$时，T_{ij}中只有叶结点d_{i-1}，$E(i, i-1)=q_{i-1}$
  - 当$j \geq i$时，选择一个$k_r \in \{k_i, \ldots, k_j\}$:



当把左右优化子树放进T_{ij}时，每个结点的深度增加1

$$E(i, j)=P_r + E(左子树) + W(i, r-1) + E(右子树) + W(r+1, j)$$

- **计算W(i, r-1)和W(r+1, j)**

由 $$E(LT+1)=\sum_{l=i}^{r-1}(DEP_{左}(k_l)+2)p_l+\sum_{l=i-1}^{r-1}(DEP_{左}(d_l)+2)q_l$$

$$E(LT)=\sum_{l=i}^{r-1}(DEP_{左}(k_l)+1)p_l+\sum_{l=i-1}^{r-1}(DEP_{左}(d_l)+1)q_l$$

$E(i, j) = E(i, r\text{-}1) + E(r\text{+}1, j) + W(i, j)$

知 $$W(i,r-1)=E(LT+1)-E(LT)=\sum_{l=i}p_l+\sum_{l=i-1}q_l$$

同理, $$W(r+1,j)=\sum_{l=i}^{j}p_l+\sum_{l=i-1}^{j}q_l$$

$W(i, j)=W(i, r\text{-}1) + W(r\text{+}1, j) + P_r$

总之

$$E(i, j)=q_{i-1} \qquad \text{If } j=i-1$$

$$E(i, j)=\min_{i\le r\le j}\{E(i,r-1)+E(r+1,j)+W(i, j)\} \qquad \text{if } j\ge i$$

自下而上计算优化解的搜索代价

- $E(i, j)=\min_{i\le r\le j}\{E(i,r-1)+E(r+1,j)+W(i, j)\}$

$q_0 =$	$E(1,0)$	$E(1,1)$	$E(1,2)$	$E(1,3)$	$E(1,4)$
$q_1 =$		$E(2,1)$	$E(2,2)$	$E(2,3)$	$E(2,4)$
$q_2 =$			$E(3,2)$	$E(3,3)$	$E(3,4)$
$q_3 =$				$E(4,3)$	$E(4,4)$
$q_4 =$					$E(5,4)$

- $W(i, i\text{-}1) = q_{i-1},\quad W(i, j) = W(i, j\text{-}1) + p_j + q_j$

$q_0 =$	$W(1,0)$	$W(1,1)$	$W(1,2)$	$W(1,3)$	$W(1,4)$
	$q_1 =$	$W(2,1)$	$W(2,2)$	$W(2,3)$	$W(2,4)$
		$q_2 =$	$W(3,2)$	$W(3,3)$	$W(3,4)$
			$q_3 =$	$W(4,3)$	$W(4,4)$
				$q_4 =$	$W(5,4)$

- 算法
  - 数据结构
    - *M[1:n+1; 0:n]*: 存储优化解搜索代价
    - *W[1: n+1; 0:n]*: 存储代价增量
    - *Root[1:n; 1:n]*: *root(i, j)*记录子问题$\{k_i, \ldots, k_j\}$优化解的根

Optimal-BST(*p*, *q*, *n*)

```
For  i=1  To  n+1   Do
     E(i, i-1) = q_{i-1};
     W(i, i-1) = q_{i-1};
For  l=1  To  n   Do
     For   i=1   To   n-l+1   Do
          j=i+l-1;
          E(i, j)=∞;
          W(i, j)=W(i, j-1)+p_j+q_j;
          For  r=i  To  j   Do
               t=E(i, r-1)+E(r+1, j)+W(i, j);
               If   t<E(i, j)
               Then  E(i, j)=t;  Root(i, j)=r;
Return  E and  Root
```

思考：优化解的构造算法

本讲内容

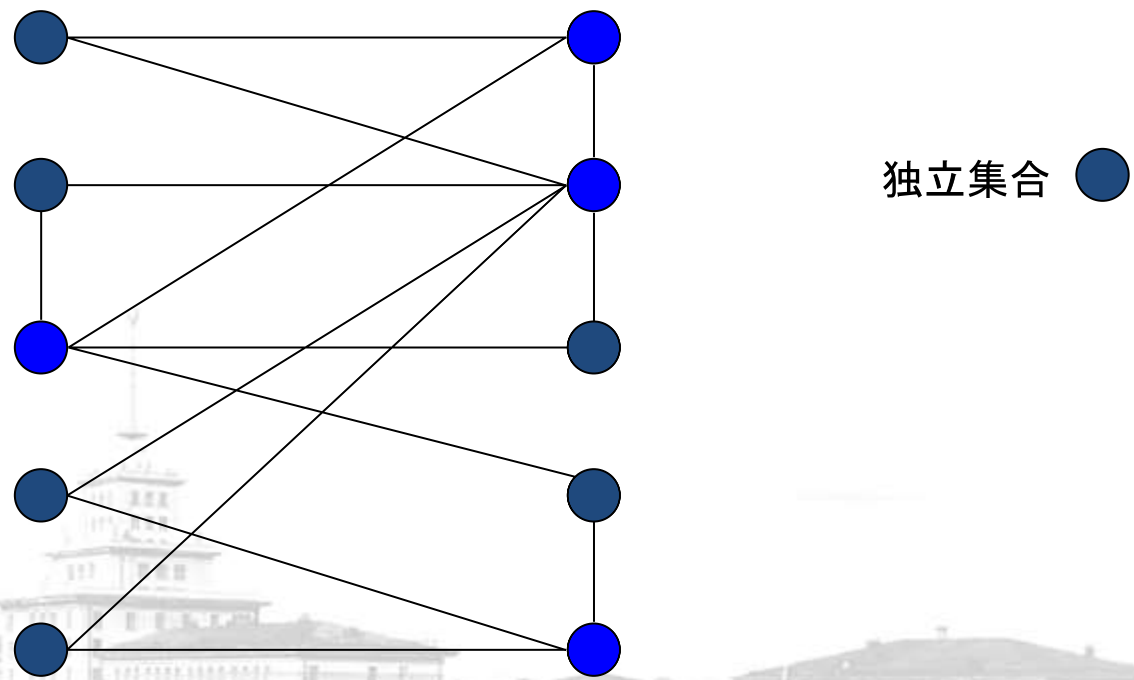
3.1 最优二分搜索树

3.2 树的独立集合

3.3 任意两点最短路径问题

最大独立集合

独立集合：输入图G = (V, E)，输出结点最大的子集合 S ⊆ V，满足E的每条边中两个顶点至多有一个在S之内。

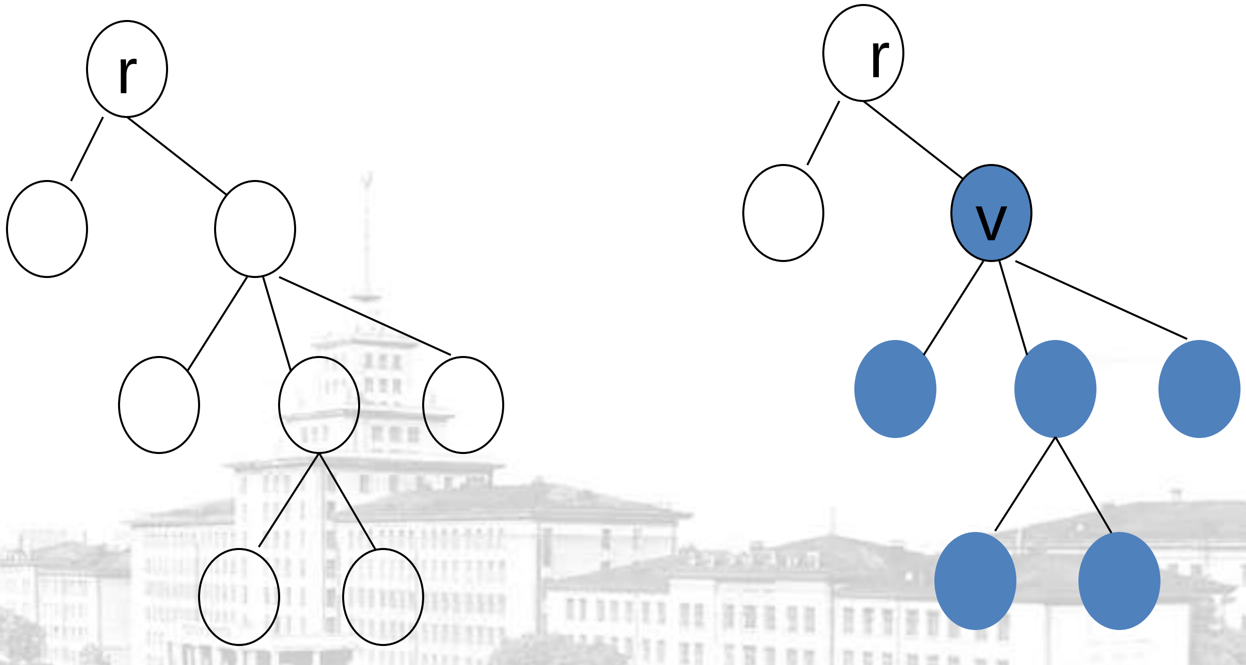


最大独立集合

- 对于一般的图是NP难问题
- 如果G是树T=(V,E)的时候呢?

最大独立集合

- 为T选择根r
- 对于T中的顶点v, 令T(v)为T中以v为根的子树



动态规划

- OPT(v): T(v)的最大独立子集合
- 我们要的结果是OPT(r)

$$OPT(v) = \begin{cases} 1, & \text{T(v) has one node} \\ \max\left(1 + \sum_{\text{w is a grandchild of v}} \text{OPT(w)}, \sum_{\text{w is a child of v}} \text{OPT(w)}\right) \end{cases}$$

算法

```
IS(v)
  If v 没有儿子
        M[v]=1
  ElseIf M[v] 非空
        SF←0;SN←1
        对于v的每一个儿子w
            SF ← SF+ IS(w)
        对于v的每一个孙子w
            SN ← SN+ IS(w)
        M[v]← max(SF,SN)
  End if
Return M[v]
```

本讲内容

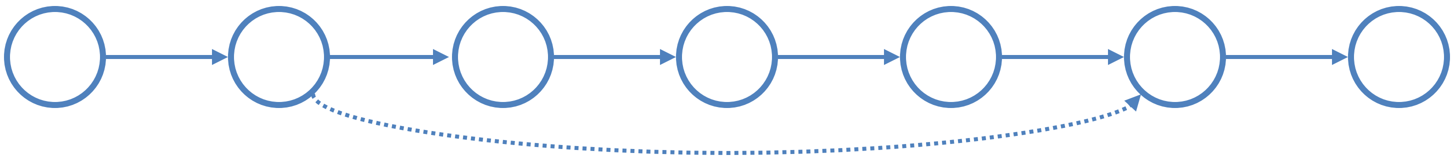
3.1 最优二分搜索树

3.2 树的独立集合

3.3 任意两点最短路径问题

最短路径的性质

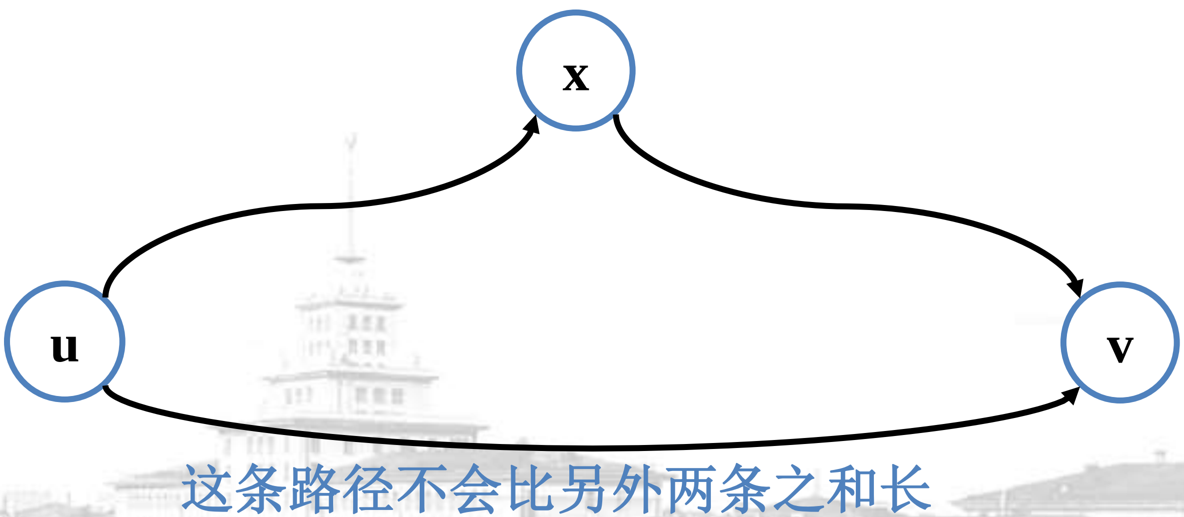
- 优化子结构: 最短路径包含最短子路径



  - 证明: 如果某条子路径不是最短子路径
    - 必然存在最短子路径
    - 用最短子路径替换当前子路径
    - 当前路径不是最短路径，矛盾!

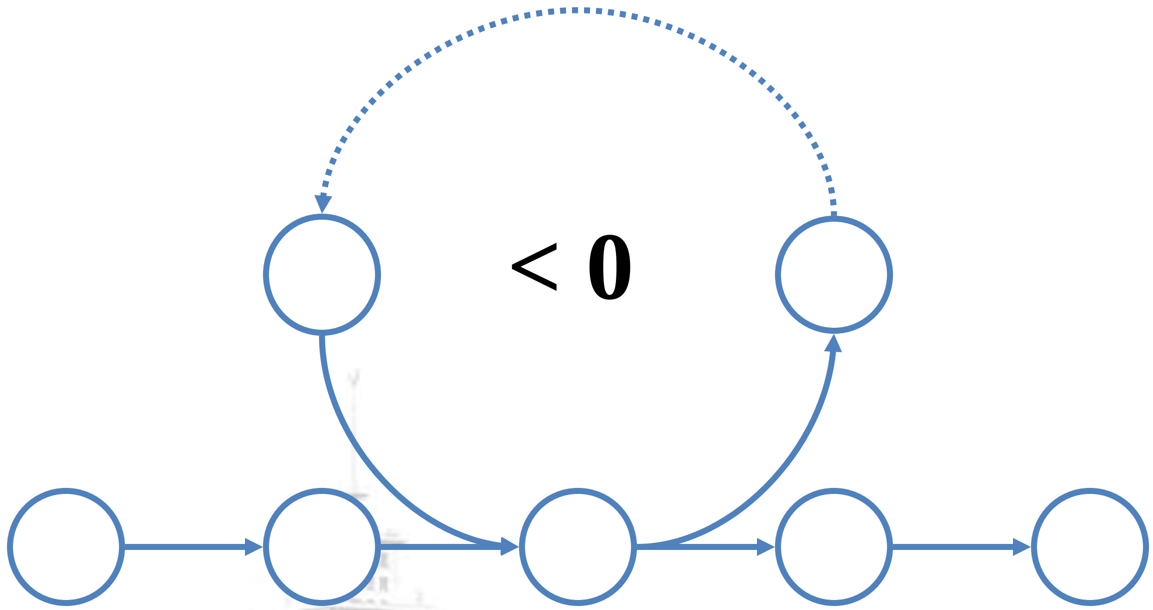
最短路径的性质

- $\delta(u,v)$ 是从u到v最短路径的权重
- 最短路径满足 *三角不等式*: $\delta(u,v) \le \delta(u,x) + \delta(x,v)$
- "证明"：



最短路径的性质

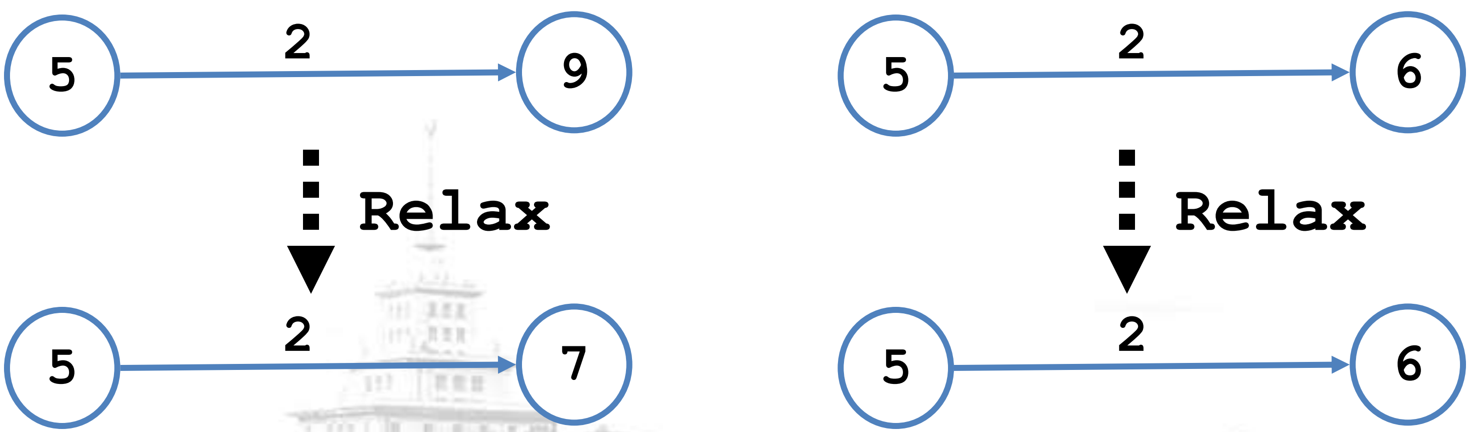
- 如果图中包含负圈，某些最短路径可能不存在 (*Why?*):



松弛

- 最短路径算法的核心技术是*松弛*

```
Relax(u,v,w) {
    if (d[v] > d[u]+w) then d[v]=d[u]+w;
}
```

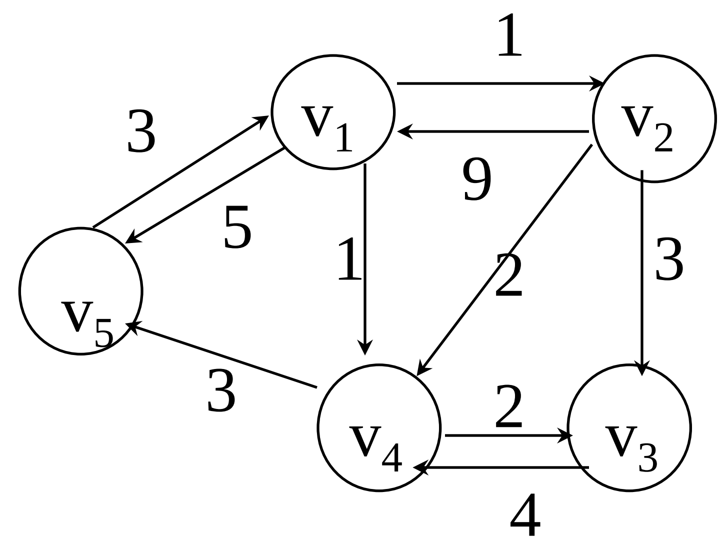


任意两点最短路径

- ***Problem:*** 找到图中每一对结点间最短路径
- 图可能包含负边但是不包含负圈
- 表示：权矩阵

图和权矩阵

	1	2	3	4	5
1	**0**	**1**	**∞**	**1**	**5**
2	**9**	**0**	**3**	**2**	**∞**
3	**∞**	**∞**	**0**	**4**	**∞**
4	**∞**	**∞**	**2**	**0**	**3**
5	**3**	**∞**	**∞**	**∞**	**0**



子问题

- 如何定义更小的问题?
- 一种方法是将路径限制在仅包含一个有限集合中的结点
- 开始这个集合是空的
- 这个集合可以一直增长到包含所有结点。

子问题

- 令 $D^{(k)}[i,j]$=从v_i 到 v_j仅包含 $\{v_1,v_2,...,v_k\}$ 的路径。
  - $D^{(0)}=W$
  - $D^{(n)}=D$ 目标矩阵
- 如何从$D^{(k-1)}$计算 $D^{(k)}$?

递归定义

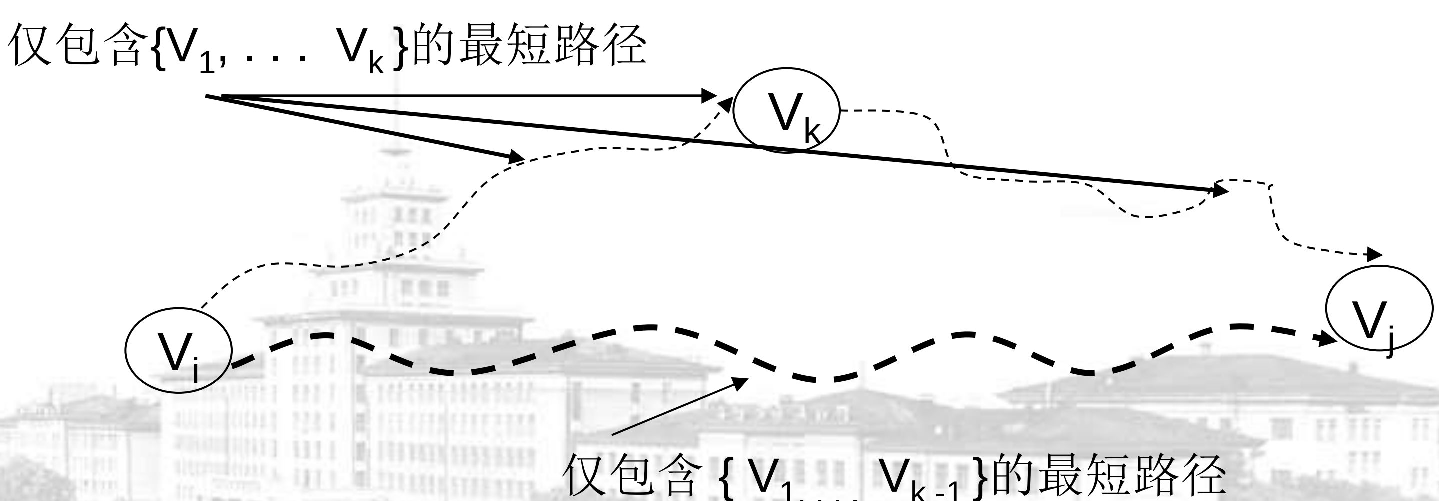
- 因为

  $D^{(k)}[i,j] = D^{(k-1)}[i,j]$ or

  $D^{(k)}[i,j] = D^{(k-1)}[i,k] + D^{(k-1)}[k,j]$.

  我们得到

- $D^{(k)}[i,j] = \min\{ D^{(k-1)}[i,j], D^{(k-1)}[i,k] + D^{(k-1)}[k,j] \}$.



Floyd算法

$D^0 \leftarrow W$ //初始化D

$P \leftarrow 0$ // 初始化 P

for $k \leftarrow 1$ **to** n

  do for $i \leftarrow 1$ **to** n

    do for $j \leftarrow 1$ **to** n

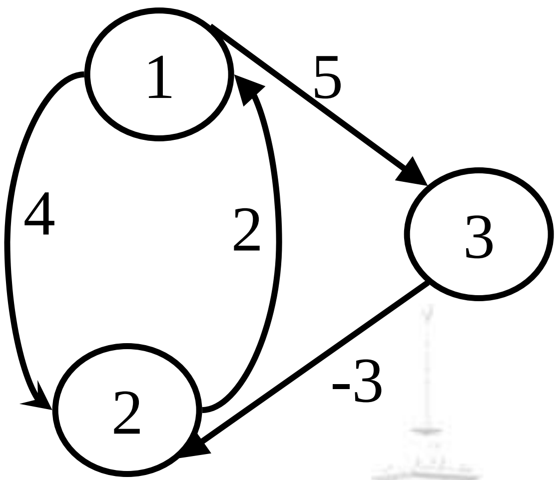
      if $(D^{k-1}[i, j] > D^{k-1}[i, k] + D^{k-1}[k, j])$

        then $D^k[i, j] \leftarrow D^{k-1}[i, k] + D^{k-1}[k, j]$

          $P[i, j] \leftarrow k;$

        else $D^k[i, j] \leftarrow D^{k-1}[i, j]$

例子

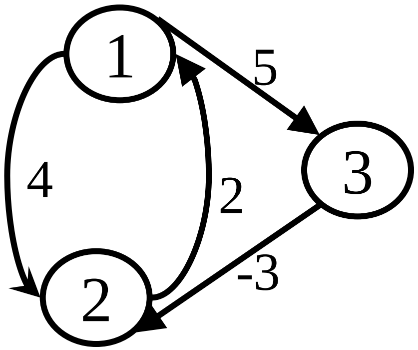


$W = D^0 =$

	1	2	3
1	0	4	5
2	2	0	∞
3	∞	-3	0

P =

	1	2	3
1	0	0	0
2	0	0	0
3	0	0	0



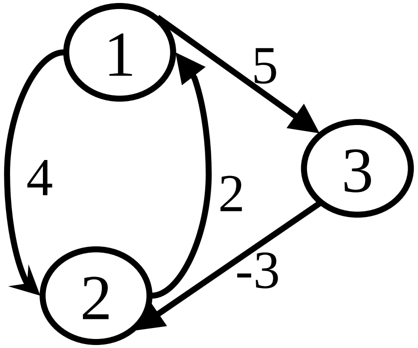
$$D^0 = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 4 & 5 \\ 2 & 2 & 0 & \infty \\ 3 & \infty & -3 & 0 \end{array}$$

$$D^1 = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 4 & 5 \\ 2 & 2 & 0 & 7 \\ 3 & \infty & -3 & 0 \end{array}$$

$$D^1[2,3] = \min(D^0[2,3], D^0[2,1]+D^0[1,3])$$
$$= \min (\propto, 7)$$
$$= 7$$

$$P = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 \end{array}$$

$$D^1[3,2] = \min(D^0[3,2], D^0[3,1]+D^0[1,2])$$
$$= \min (-3,\propto)$$
$$= -3$$



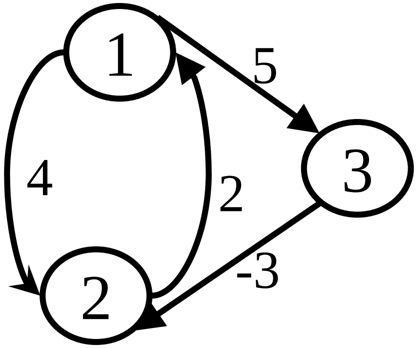
$$D^1 = \begin{array}{c|c|c|c|} & 1 & 2 & 3 \\ \hline 1 & 0 & 4 & 5 \\ \hline 2 & 2 & 0 & 7 \\ \hline 3 & \infty & -3 & 0 \\ \hline \end{array}$$

$$D^2 = \begin{array}{c|c|c|c|} & 1 & 2 & 3 \\ \hline 1 & 0 & 4 & 5 \\ \hline 2 & 2 & 0 & 7 \\ \hline 3 & -1 & -3 & 0 \\ \hline \end{array}$$

$$P = \begin{array}{c|c|c|c|} & 1 & 2 & 3 \\ \hline 1 & 0 & 0 & 0 \\ \hline 2 & 0 & 0 & 1 \\ \hline 3 & 2 & 0 & 0 \\ \hline \end{array}$$

$$\begin{aligned} D^2[1,3] &= \min(D^1[1,3], \\ &\quad D^1[1,2]+D^1[2,3]) \\ &= \min (5, 4+7) \\ &= 5 \end{aligned}$$

$$\begin{aligned} D^2[3,1] &= \min(D^1[3,1], \\ &\quad D^1[3,2]+D^1[2,1]) \\ &= \min (\propto, -3+2) \\ &= -1 \end{aligned}$$



$D^2 =$

	1	2	3
1	0	4	5
2	2	0	7
3	-1	-3	0

$D^3 =$

	1	2	3
1	0	2	5
2	2	0	7
3	-1	-3	0

$P =$

	1	2	3
1	0	3	0
2	0	0	1
3	2	0	0

$D^3[1,2] = \min(D^2[1,2],$
$D^2[1,3]+D^2[3,2])$
$= \min(4, 5+(-3))$
$= 2$

$^3[2,1] = \min(D^2[2,1],$
$D^2[2,3]+D^2[3,1])$
$= \min(2, 7+(-1))$
$= 2$

Floyd算法: 使用两个 D 矩阵

Floyd

1. $D \leftarrow W$
2. $P \leftarrow 0$
3. for $k \leftarrow 1$ to n

   // Computing D' from D

4. do for $i \leftarrow 1$ to n
5. do for $j \leftarrow 1$ to n
6. if ($D[\,i, j\,] > D[\,i, k\,] + D[\,k, j\,]$)
7. then $D'[\,i, j\,] \leftarrow D[\,i, k\,] + D[\,k, j\,]$
8. $P[\,i, j\,] \leftarrow k$;
9. else $D'[\,i, j\,] \leftarrow D[\,i, j\,]$
10. *Move D' to D.*

Floyd算法：使用1个D

Floyd

1. $D \leftarrow W$
2. $P \leftarrow 0$
3. **for** $k \leftarrow 1$ **to** n
4. **do for** $i \leftarrow 1$ **to** n
5. **do for** $j \leftarrow 1$ **to** n
6. **if** $(D[\,i, j\,] > D[\,i, k\,] + D[\,k, j\,])$
7. **then** $D[\,i, j\,] \leftarrow D[\,i, k\,] + D[\,k, j\,]$
8. $P[\,i, j\,] \leftarrow k$;

打印出从q到r的路径

```
path(index q, r)
    if (P[ q, r ]!=0)
        path(q, P[q, r])
        println( "v"+ P[q, r])
        path(P[q, r], r)
        return;
    //no intermediate nodes
    else return
```

$$P = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 3 & 0 \\ 2 & 0 & 0 & 1 \\ 3 & 2 & 0 & 0 \end{array}$$