道路与回路

有向道路与有向回路

- 定义:有向图G中边序列$P=(e_{i_1}, e_{i_2},\ldots, e_{i_q})$, 其中$e_{ij}=(v_k,v_l)$满足: v_k是e_{ij-1}的终点, v_l是e_{ij+1}的始点,就称P是G的一条有向道路.如果e_{iq}的终点也是e_{i_1}的始点,则称P是G的一条有向回路.
- 简单有向道路/回路:P中边不重复出现.
- 初级有向道路/回路: P中结点不重复出现.

道路与回路

- 定义:无向图G中,若点边交替序列

$$P=(v_{i_1}, e_{i_1}, v_{i_2}, e_{i_2}, \cdots, e_{i_{q-1}}, v_{i_q})$$

满足: v_{ij}和$v_{i_{j+1}}$是e_{ij}的两个端点,则称P是G中的一条链或道路(*path*).如果$v_{i_q}=v_{i_1}$,则称P是G中的一个圈或回路(*circuit*).

- 简单道路/回路:P中没有重复出现的边.
- 初级道路/回路:P中没有重复出现的结点.

连通性

- 若无向图G的任意两个结点之间都存在道路,就称G是**连通的**(*connected*).
- 对有向图G,若不考虑边的方向时是连通的,则称G是(弱)连通的.
- 若G的连通子图H不是G的任何连通子图的真子图,则称H是G的极大连通子图,或称**连通支**(*connected component*).
  - 显然G的每个连通支都是它的导出子图.
  - 任何非连通图都是2个以上连通支的并.

道路与回路的判定

- 判断图中两个结点之间是否存在道路,或说是否连通.
- 方法:
  - 邻接矩阵法
  - Warshall算法
  - 搜索法
    - 广探法(广度优先搜索)
    - 深探法(深度优先搜索)

邻接矩阵法

- 利用G的邻接矩阵$A=(a_{ij})_{n\times n}$判断两点间是否可经一条边连通,因为

$$a_{ij}=1 \ iff \ (v_i, v_j)\in E(G)$$

- 利用$A^2=(a_{ij}^{(2)})_{n\times n}$判断两点间是否可经两条边连通,因为

$$a_{ij}^{(2)}=\Sigma^n_{k=1}\ a_{ik}a_{kj}\neq 0 \ iff \ \exists k\text{使}a_{ik}=a_{kj}=1$$

$$iff \ \exists v_k\text{使}(v_i, v_k),(v_k, v_j)\in E(G)$$

- 一般地,利用A^s判断两点可经s条边连通.

邻接矩阵法(续)

- 令$P = A + A^2 + \ldots + A^n = (p_{ij})_{n\times n}$
  - 若$p_{ij}=t$,则从v_i到v_j有t条道路;
  - 若$p_{ij}=0$,则n步之内从v_i不能到达v_j,从而v_i和v_j之间没有道路.

Warshall算法

- 若只关心两点间道路的存在性,不关心道路的长度和数量,则前面的方法中可改用逻辑运算:

  $a_{ij}^{(s)} = \vee^{n}_{k=1} (a_{ik}^{(s-1)} \wedge a_{kj}^{(s-1)})$

  $P = A \vee A^2 \vee \ldots \vee A^n$

  – P称为图G的道路矩阵: $p_{ij} = 1$ iff v_i 与v_j之间有道路.

- 计算道路矩阵P的Warshall算法

```
P ← A;
for i = 1 to n do
    for j = 1 to n do
        for k = 1 to n do
            p_jk ← p_jk ∨ (p_ji ∧ p_ik)
```

Warshall算法(续)

- 算法思想
  - 第i次循环:对当前(第$i-1$次循环的结果)不直接连通的顶点v_j和v_k(即没有边(v_j,v_k)),看它们是否可以通过v_i间接连通(即存在边(v_j,v_i)和(v_i,v_k)).如果是,则在原图中增加边(v_j,v_k).
  - 最后得到道路矩阵
- 定理:Warshall算法的结果确是图G的道路矩阵.

广探法

- 广度优先搜索(*breadth first search*,BFS)

  判断从v_0到v_i是否存在道路:

  (1)令$A_0 = \{v_0\}$.

  (2)对A_k中的每个结点v,求$\Gamma^+(v)$,令

  $$A_{k+1} = \cup_{v \in A_k} \Gamma^+(v)$$

  (3)若$v_i \in A_{k+1}$,则存在从v_0到v_i的道路;

  否则,若还有未搜索过的结点,令$A_k \leftarrow A_{k+1}$,返回(2)继续.

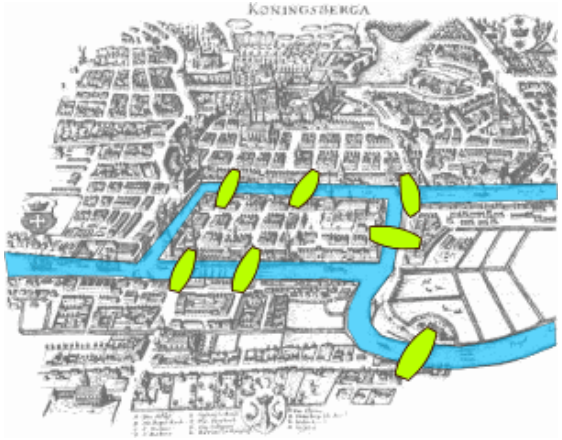
  – 可通过做记号,避免重复搜索同一个结点.

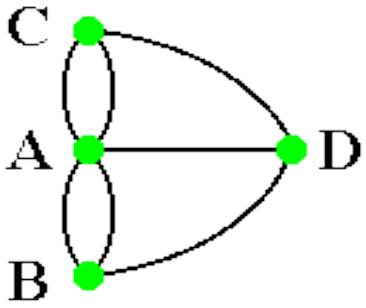
深探法

- 深度优先搜索(*depth first search*,DFS)
  判断从v_0到v_i是否存在道路:
  (1)令$v_k = v_0$;
  (2)求v_k的一个未搜索过的直接后继v;
  (2)若$v = v_i$则道路存在;
  (3)若$v \neq v_i$且v有直接后继,则令$v_k = v$并返回(2);
  若$v \neq v_i$且v没有直接后继,则返回(2)回溯.
  - 可通过把经过的结点压入堆栈,来记住回溯点.
  - 可通过做记号,避免重复搜索同一个结点.

哥尼斯堡七桥问题

- 1736年,Euler发表论文“哥尼斯堡的七座桥”,解决了下图中是否存在经过每条边一次且仅一次的回路的问题.





The Seven Bridges of Königsberg

1.

The branch of geometry that deals with magnitudes has been zealously studied throughout the past, but there is another branch that has been almost unknown up to now; Leibniz spoke of it first, calling it the "geometry of position" (*geometria situs*). This branch of geometry deals with relations dependent on position alone, and investigates the properties of position; it does not take magnitudes into consideration, nor does it involve calculation with quantities. But as yet no satisfactory definition has been given of the problems that belong to this geometry of position or of the method to be used in solving them. Recently there was announced a problem that, while it certainly seemed to belong to geometry, was nevertheless so designed that it did not call for the determination of a magnitude, nor could it be solved by quantitative calculation; consequently I did not hesitate to assign it to the geometry of position, especially since the solution required only the consideration of position, calculation being of no use. In this paper I shall give an account of the method that I discovered for solving this type of problem, which may serve as an example of the geometry of position.

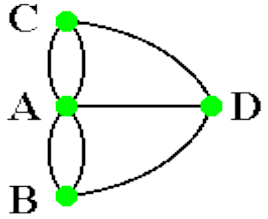
2. The problem, which I understand is quite well known, is stated as follows: In the town of Königsberg in Prussia there is an island A, called

欧拉道路与回路

- 定义:无向连通图$G=(V,E)$中包含所有边的简单回路(道路)称为G的欧拉回路(道路).
- **定理:无向连通图G中存在欧拉回路的充要条件是G中各结点的度都是偶数.**

  $\Rightarrow$:对任一v,回路沿e_i进入v必然有e_j从v出来.

  $\Leftarrow$:从任一v_0出发必能构造一简单回路C(否则终点不是偶数度).令$G'=G-C$,对G'的各连通支继续此过程.最后合并所有回路即所求.
- 例:七桥问题无欧拉回路.



欧拉道路与回路(续)

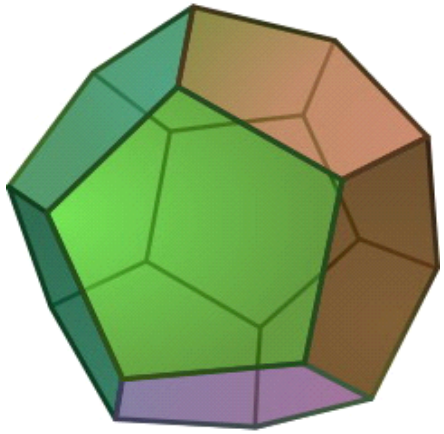
- 推论**1:**无向连通图***G***中存在欧拉道路当且仅当***G***中只有两个度为奇数的结点**.**

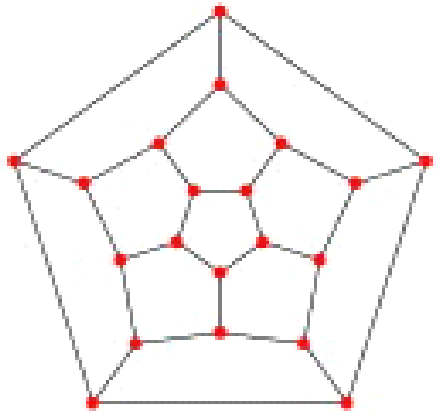
  证:连接这两顶点,则有回路.再删去这条边.

- 推论2:若有向连通图*G*中各结点的正负度相等,则*G*中存在有向欧拉回路.

周游世界游戏

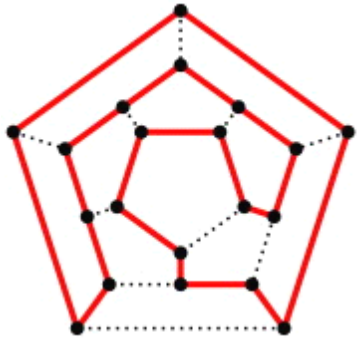
- 1857年,William Rowan Hamilton发明了*Icosian*游戏. 游戏目标是沿着一个十二面体(12个正五边形的面,20个顶点,30条棱)的棱边找到一条不重复地遍历各顶点的回路.





哈密顿回路与道路

- 定义:无向图G的一条经过全部结点的初级回路(道路)称为G的哈密顿回路(道路).
  - 简记为H回路(道路).
- 对H回路问题
  - 要求$V(G) = n \geq 3$
  - 只需考虑简单图,因为重边和自环不起作用
- H回路的判定很困难,没有发现充分必要的条件,只有若干充分条件.



*H*道路的判定

- 定理:若简单图G的任意两结点v_i与v_j之间恒有

$$d(v_i) + d(v_j) \geq n-1$$

则G中存在H道路.

证明思路:

(1)由定理条件:*G*是连通图.

(2)令*P*是*G*中最长初级道路,则*P*是*H*道路.若不是:

(i) 由定理条件:必有经过*P*中结点的初级回路*C*.

(ii) 由连通性:*C*必可与*C*外某相邻结点构成比*P*更长的初级道路.

H回路的判定

- 定理**(Ore,1960)**:若简单图$G(n\geq 3)$的任一对不相邻结点v_i与v_j都满足

$$d(v_i) + d(v_j) \geq n$$

则G有H回路.

  - 书上推论2.4.1条件更宽,且漏了$n\geq 3$的条件.

- 定理**(Dirac,1952)**:若简单图$G(n\geq 3)$的任一结点的度$\geq n/2$,则G有H回路.

  - 书上推论2.4.2漏了$n\geq 3$的条件.

H回路的判定(续)

- 引理:简单图G若有不相邻结点v_i, v_j满足$d(v_i) + d(v_j) \geq n$,则

  G有H回路 *iff* $G+(v_i,v_j)$有H回路.

  – 对G不断加入这样的边(v_i,v_j),直至不再有满足条件的结点对,最终得到的图称为G的**闭合图**,记作$C(G)$.
- 引理:简单图G的闭合图是唯一的.
- **定理(Bondy&Chvátal,1976)**

  简单图G有H回路 *iff* $C(G)$有H回路.
- 推论:若$C(G)=K_n$,则G有H回路.

  – Ore定理和Dirac定理显然是这个推论的直接推论.

旅行商问题

- 实际问题中往往涉及赋权图.
- TSP(*traveling salesman problem*):给定一正权完全图,求总权值最小的H回路.

右图:经过德国15个大城市的一个TSP行程. 这是43,589,145,600个可能行程中最短的一个.

精确解法:穷举搜索

- 最直接的精确解法就是穷举搜索,即检查所有可能的*H*回路,计算各自的总权值,选择最小者.
- 对K_n,存在$(n-1)!/2$个*H*回路.(Why?)
  - 例如: 对$n=25$,

    $$24!/2 \approx 3.1\times10^{23}$$

    如果每纳秒(10^{-9}秒,十亿分之一秒)检查一条*H*回路,大约需要一千万年才能得到最优解.

精确解法:分支与定界

- ***Branch and Bound*** (BB)是一种求解各类最优化问题(尤其是离散与组合最优化问题)的一般算法.
- 算法思想:系统地枚举所有的候选解,利用被优化量的上界和下界,从候选空间将大批不可能候选全部丢弃.
- 最早由A. H. Land和A. G. Doig于1960年在线性规划领域中提出.

求解TSP:分支与定界

结点: $v_1,\ldots, v_n$,边:$e_{12}, e_{13},\ldots, e_{1n}, e_{23}, e_{24},\ldots, e_{(n-1)\,n}$

初始上界$d_0\leftarrow\infty$;

(1)所有边按权值从小到大排序: $e^{(1)},e^{(2)},\ldots,e^{(m)}$

(2) $i\leftarrow 1$,对边序列按DFS选n条边构成边集s_i;经过的点入栈.

(3)判断s_i是否H回路.

- 方法: s_i中每个结点标号只出现2次,且所有边只构成一个回路.

(4)若否:删除s_i中最长边$e^{(k)}$,换成$e^{(k+1)}$,得s_{i+1}.$i\leftarrow i+1$,返回(3);

若是:且$i=1$则$d_0\leftarrow d(s_1)$,结束;否则若$d(s_i)<d_0$,则$d_0\leftarrow d(s_i)$.

(5)若栈空,则结束,d_0即最优解.

否则退栈,建立一个新分支:若新分支的$d(s_i)\geq d_0$,继续退栈;若$d(s_i)<d_0$,则返回(3).

近似解法:便宜算法

- 在K_n满足三角不等式(即:无捷径)时的一个算法:设顶点集为$S=\{1,2,\ldots,n\}$.

  (1)回路$T=(1,1)$; 剩余结点$S'=\{2,3,\ldots,n\}$;

  $w(1,1)=0$; $\forall i\in S', k\in T$: $w(i,k)=w(i,1)$;

  (2)在S'与T之间求最近的一对顶点$j\in S'$, $t\in T$,即

  $w(j,t)=\min_{i\in S',k\in T} w(i,k)$

  将j插入到$T=(\ldots, t_1, t, t_2,\ldots)$中:若

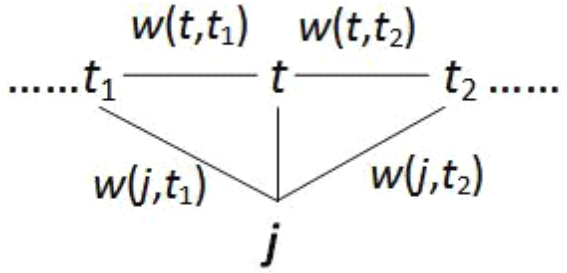
  $w(j,t_1)-w(t,t_1)\le w(j,t_2)-w(t,t_2)$

  则插到t与t_1之间;否则插到t与t_2之间.

  (3) $S'\leftarrow S'-j$;若$S'=\varnothing$则结束;

  否则$\forall i\in S'$, $w(i,k)\leftarrow \min\{w(i,k),w(i,j)\}$;

  转(2).



便宜算法的效果

- 定理:设正权完全图的边权满足三角不等式,其TSP的最佳解是O_n,便宜算法的解是T_n,则$T_n/O_n<2$.
  (教材中此定理的证明对记号的使用有问题)
- 便宜算法的计算复杂度为$O(n^2)$.

最短路径问题

- *Shortest Path Problem*:给定赋权图,求结点间的最短道路,即道路权值之和最小.
  - Single-pair:某对结点间;
  - Single-source:某结点到其他各结点间;
  - All-pairs:任意结点对之间.
- 权值的情况:
  - 都是正数;
  - 都等于1;
  - 任意实数.

约定和记号

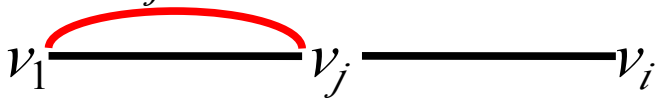
- 我们讨论*single-source*情形.即局限于求v_1到其他各点v_i的最短路径.
- 记v_1到v_i的一条路径$P(i)$的长度为

$$\pi(i) = \Sigma_{e \in P(i)} w(e) = \Sigma_{(v_j, v_k) \in P(i)} w_{jk}$$

- 若$P(i)$中含有回路C,令$P'(i)$是其中不含C的初级道路,显然$\pi(i) = \pi'(i) + \pi(C)$.
  - 若$\pi(C) < 0$,则不存在最短$P(i)$.
  - 若$\pi(C) \geq 0$,则$\pi'(i) \leq \pi(i)$.即最短路径一定是初级道路.

正权图最短路径的性质

- 引理1:正权图中,若$P(i)$是v_1到v_i的最短路径,且$v_j \in P(i)$,则$P(j)$是v_1到v_j的最短路径.



- 引理2:正权图中任意一条最短路径的长度大于其局部路径长度.
- 将v_1到各点的最短路径从小到大排列:

$$\pi(1)=\pi(i_1) \le \pi(i_2) \le \ldots \le \pi(i_n)$$

由引理2:对$k>l\ge 1$,$P(i_k)$不是$P(i_l)$之部分.(由短生长)

再由引理1:$\boldsymbol{\pi(i_l) = \min_{1\le j<l}(\pi(i_j) + w_{i_j i_l})}$.

– 此即Dijkstra算法核心思想.

Dijkstra最短路径算法

- Dijkstra算法

(1)置$S=\{2,3,\ldots,n\}$;$\pi(1)=0$;

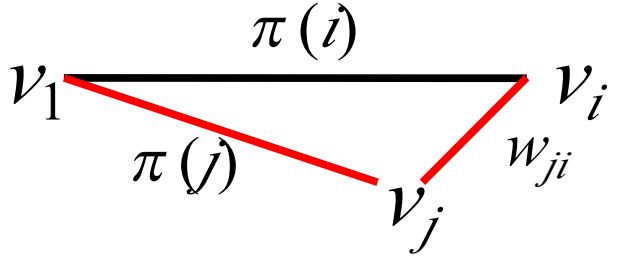
对$i\in S$,若$i\in\Gamma^+(1)$则$\pi(i)=w_{1i}$,否则$\pi(i)=\infty$.

(2)令$\pi(j)=\min_{i\in S}\pi(i)$.

置$S\leftarrow S-\{j\}$,若$S=\varnothing$,结束.

(3)对$\forall i\in S\cap\Gamma^+(j)$,置$\pi(i)\leftarrow\min(\pi(i),\pi(j)+w_{ji})$,

转(2).



关键路径方法

- 关键路径方法(*critical path method*,CPM)是用来调度一系列工程活动的基于数学的算法,是项目管理的重要工具 .
  - CPM由DuPont公司于1950s开发.
- CPM有广泛应用.任何由相互依赖的任务组成的项目都可应用此调度方法.
- CPM的基本技术是构造项目模型,其中包括一系列任务,各任务所需时间,任务间的依赖关系.

关键路径

- 关键路径是构成最长路径的任务序列.它决定了项目的最早结束时间.其中任何任务的延误都将直接影响项目完成时间.
- CPM计算关键路径,以及各任务的开始和结束的最早/最晚时间.
  - 此过程确定了哪些任务是关键的(不可延误),哪些是浮动的(可延迟).

求解方法

- 假设任务之间只存在时间次序的约束.
- 有两种图示法:
  - PT图
  - PERT图

PT图

- PT(*potential task*)图:用结点表示工序,若工序*i*完成之后工序*j*才能开始,则用有向边(*i*,*j*)表示,边权w_i表示工序*i*所需时间.
- 必不存在有向回路.
- 任一工序*i*的**最早开始时间**=从工序1到工序*i*的最长路径(时间).
- 任一工序*i*的**最晚开始时间**=(从工序1到完工的最长路径) – (从工序*i*到完工的最长路径).
- 任一工序*i*的**允许延误时间**=最晚开始时间–最早开始时间.

工序最早开始时间的分析

- 引理:若图G中不存在有向回路,则必存在负度或正度为零的结点.
- 定理:若图G中不存在有向回路,可以将G的结点重新编号为v_1', v_2',…, v_n',使得对任意边$(v_i', v_j') \in E(G)$,都有$i<j$.
  - 设v_1'到各点的最长路径为

$$0=\pi(v_1'),\pi(v_2'),\ldots,\pi(v_n')$$

  则 $\boldsymbol{\pi(v_j') = \max_{1\le i<j}(\pi(v_i') + w(v_i', v_j'))}$.
  - 此即最长路径算法的基础.

工序最早开始时间的算法

- 算法1(最长路径算法)

(1)对结点如前所述重新编号: v_1', v_2',…, v_n';

(2)$\pi(v_1') \leftarrow 0$;

(3)对j从2到n,计算

$$\pi(v_j') = \max_{v_i' \in \Gamma^-(v_j')} (\pi(v_i') + w(v_i', v_j')).$$

工序最晚开始时间的分析

- 设$\pi(v_n')$是最早完工时间,则工序i的最晚开始时间为$\tau(v_i')=\pi(v_n')-\pi(v_i',v_n')$.
- 问题变成:如何计算$\pi(v_i',v_n')$?
- 方法1:图G转置可得G'.在G'中以v_n'为起点,它到各点的最长路径可用前面算法实现.
  - 上式就成了$\tau(v_i')=\pi(v_1')-\pi(v_i')$,可以算出$\tau(v_i')$.
- 方法2:更好的算法仍利用原图G进行计算.根据

  $$\pi(v_i',v_n') = \max_{v_j' \in \Gamma^+(v_i')}(\pi(v_j',v_n') + w(v_i',v_j'))$$

  逆序求出$0=\pi(v_n',v_n')$, $\pi(v'_{n-1},v_n')$,…,$\pi(v_1',v_n')$.

工序最晚开始时间的算法

- 算法2

(1)对结点如前所述重新编号: v_1', v_2',…, v_n';

(2)$\tau(v_n') = \pi(v_n')$;

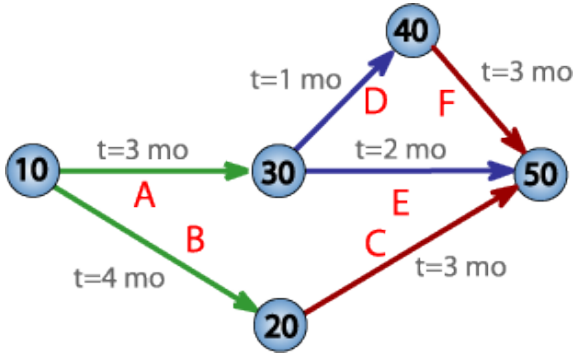
(3)对j从n–1到1,计算

$$\tau(v_j') = \pi(v_n') - \pi(v_j', v_n')$$
$$= \pi(v_n') - \max_{v_i' \in \Gamma^+(v_j')}(\pi(v_i', v_n') + w(v_j', v_i'))$$
$$= \min_{v_i' \in \Gamma^+(v_j')}(\pi(v_n') - \pi(v_i', v_n') - w(v_j', v_i'))$$
$$= \mathbf{\min_{v_i' \in \Gamma^+(v_j')}(\tau(v_i') - w(v_j', v_i'))}.$$

- 工序i的允许延误时间$t(v_i')$=$\tau(v_i') - \pi(v_i')$.

PERT图

- PERT(*Program Evaluation and Review Technique*):是一种项目管理模型,用来表示和分析项目涉及的任务.
  - 由通用动力公司(GD)和美国海军于1950s开发.
- 有向边表示工序,边权值表示该工序所需时间.
- 结点表示工序的开始和结束.
- 如果工序e_i完成后e_j才能开始,则令v_k是e_i的终点和e_j的始点.



PERT图的分析

- 关键路径:从v_1到v_n的最长路径.
  - 其长度就是最早完工时间.
- $e_k=(v_i,v_j)$的最早开始时间是$\pi(v_i)$.
  - 用算法1.
- e_k的最晚开始时间是

  $\tau(v_i,v_j)=\pi(v_n)-\pi(v_j,v_n)-w(v_i,v_j)=\tau(v_j)-w(v_i,v_j)$;
  - 用算法2计算$\tau(v_j)$.
- 工序e_k的允许延误时间是

  $t(v_i,v_j)=\tau(v_i,v_j)-\pi(v_i)=\tau(v_j)-\pi(v_i)-w(v_i,v_j)$;

中国邮递员问题

- *Chinese Postman Problem*(CPP):邮递员从邮局出发,走遍所管区域的每条街道(允许重复),并返回邮局.要求路径总长度最短.
  - 我国管梅谷教授于1960s首先研究.
- 问题特点:不一定存在欧拉回路,这时某些街道必须重复经过.关键要使重复路段最短.

无向图的CPP

- 图G所有结点的度都是偶数:
  - 则存在欧拉回路,且任一欧拉回路都是解.
- 图G只有两个度为奇数的结点v_i和v_j:
  - 则存在v_i到v_j的欧拉道路E_{ij}.再找一条v_j到v_i的最短道路P_{ji},欧拉回路$E_{ij}+P_{ji}$是解.
- 图G有两个以上($2k$个)度为奇数的结点:
  - 定理:图G有最佳邮路L *iff*

    (1) L中的任一边最多重复一次;

    (2) 对G的任一回路C, L中处在C上的重复边长度之和不超过C长度的一半.

最佳邮路的构造

- 奇偶点图上作业法

  (1)找出度为奇数的结点(2*k*个);

  (2)奇数度结点任意分成*k*对,每一对用添加的重复边(道路)相连;

  (3)删除重复添加的边(每次删除两条),使得每条边最多重复一次;

  (4)检查所有回路*C*:

  - 若在*C*上的重复边长度和超过*C*长度的一半,则将原重复边删去,为原来没有重复的边添加重复边.

  (5)对最终的图求欧拉回路.

End