- 图算法
  - 存储
    - 邻接链表
    - 邻接矩阵
  - 搜索
    - 广度优先搜索
    - 深度优先搜索
      - 拓扑排序
      - 分解强联通分量
    - 最小生成树
      - Prim算法
      - 克鲁斯卡尔算法
      - 最小生成树和最短路径树
  - 单源最短路径
    - Bellman-Ford算法
      - 有向无环图中的DAG算法
    - Dijkstra算法
  - 所有点对的最短路径
    - 回溯最短路径
    - 使用Dijkstra/Bellman
    - 重复平方法
    - Floyd-Warshell算法
    - Johnson算法
  - 最大流
    - 基本概念
    - 性质
    - 剩余网络
      - 增广流
      - 增广路径
    - 切割
      - 概念
      - 性质
      - 最大流最小割定理
    - Ford-Fulkson算法
      - Edmonds-Karp算法

图算法

存储

邻接链表

- 存储容量$O(V+E)$
  - 有向图$O(V+2E)$
  - 无向图$O(V+E)$
- 查询某一条边是否存在
  - 遍历整个邻接链表$O(E)$
- 查询某一个顶的入/出度
  - 遍历该顶点整个链表$O(d_v)$

邻接矩阵

- 存储容量$O(V^2)$

搜索

广度优先搜索

- **给定源节点**
  - 使用邻接链表存储, 有
    - 时间复杂度$O(V+E)$
      - 每次取某个节点的所有邻居时, 只需要遍历整个链表
    - 空间复杂度$O(V+E)$
  - 使用邻接矩阵存储, 有
    - 时间复杂度$O(V^2)$
      - 每次去某个节点的所有邻居时, 需要遍历那个节点对应的行, 则有$O(V)$的复杂度, 同时一共重复$O(V)$次(**每个节点最多入队一次**)
- 性质
  - 运行完`BFS(G,s)`后, 有$\forall v \in V$且v从s可达, $v.d=\delta(s,v)$, 其中$\delta(s,v)$为s到v的**最短距离(路径上最少的边的数目)**

深度优先搜索

- 白色路径定理
  - 如果u,v之间存在一条路径p, 满足$\forall a \in p, a.color=white$, 则$v$是$u$的后代
- **给定源节点**

- 使用邻接链表存储, 有
    - 时间复杂度$O(V+E)$
    - 空间复杂度$O(V+E)$, 递归调用显然小于$min\{V,E\}$
- 使用邻接矩阵存储, 有
    - 时间复杂度$O(V^2)$?
- **不给定源节点, 对整个图进行`DFS`**
    - 使用邻接链表存储, 有
        - 时间复杂度$O(V+E)$
- 边的分类
    - $$\text{树边} \subset \text{后向边}$$

拓扑排序

- 将深度优先搜索得到的点, 按照**变黑的先后顺序**入链表(队列), 链表中的节点$v.f$从大到小
- 只能产生一个有向无环图的拓扑排序
- 时间复杂度$O(DFS)$
- 空间复杂度$O(V)$

分解强联通分量

- 时间复杂度$O(DFS+V\lg V)$
    - 因为要对$v.f$排序, 然后在G^T上进行`DFS`
- 空间复杂度$O(\text{取决于存储方式})$
- 证明
    - **归纳**: 假设算法生成的前k棵树都是SCC, 记为$\mathcal{C}=\{C_1,\cdots,C_k\}$, 考虑算法生成的第$k+1$棵树
    - 首先, 有**事实**$C\subseteq T_{k+1}\in G^{SCC}, C\notin \mathcal{C}$, 要证明
    $$T_{k+1}\subseteq C$$
    - 对于C, 有$\forall u\in C, \nexists v\in G^{SCC}-\mathcal{C}, e(v,u)\in E(G)$
        - 否则会有$C.f<C'.f$, C'为第二次`DFS`还未涉及的强连通分量, 与选取`DFS`的根的策略相矛盾
    - 因此在转置图$G(V,E^T)$中, $\forall u\in C$, 如果$\exists v, e(u,v)\in E^T$, 则一定有$v\in\mathcal{C}$, 即**所有从C指出C的边一定指向$\mathcal{C}$中的某一个强连通分量, 而此强连通分量已经被遍历过, 因此v不会被包含在T_{k+1}中**
    - 因此T_{k+1}中不包含任何C之外的点, 即$T_{k+1}\subseteq C$
    - 因此$C\subseteq T_{k+1}\wedge T_{k+1}\subseteq C$, 得证$T_{k+1}$也是$SCC$

最小生成树

- **定义在无向图**

Prim算法

- 步骤
  - 和 `Dijkstra` 类似, 将 `RELAX` 操作改为修改每个还处在优先级队列中的顶的权重和前序节点(**前序节点记录了最小生成树的结构**)
- 分析
  - 时间复杂度

$$O(E * T_{DECREASE-KEY} + V * T_{EXTRACT-MIN}) = \begin{cases} O(E \lg V + V \lg V) \\ O(E + V \lg V) \end{cases}$$

克鲁斯卡尔算法

- 步骤
  - 将所有边按照权重排序
  - 取权重最小的边$e(u, v)$
  - 如果边$e(u, v)$中u和v在A中不属于同一个连通分量, $e(u, v)$纳入集合A
  - 否则跳过
- 分析
  - 时间复杂度
    - 将所有节点变为单独的集合, 有$O(V)$
    - 按照权重排序, 有$O(E \lg E)$
    - 判断边的两端是否在同一个连通分量, 有$O(\lg V) = O(\lg E)$
    - 将合格的边合并到A中, 有$O(\lg V) = O(\lg E)$
    - 总时间复杂度$O(E \lg E)$

最小生成树和最短路径树

- 两者都包含原图的每个顶点
- 两者都是连通的

单源最短路径

- 最短路径上没有环路, 但图上可以有环路

Bellman-Ford算法

- 功能

    - 如果图中有负权重的环, 输出false
    - 如果没有, 输出true, 同时每个节点的$v.d$都已经是s(源)到v的最短路径长度
- 步骤
    - 进行$V-1$次迭代, 每次迭代中遍历所有$e(u,v) \in E$
    - `RELAX(u,v,w)`
    - 所有迭代完成后, 对每个边再次调用`RELAX(u,v,w)`, 如果仍能进行更新, 则有负圈
- 时间复杂度$O(VE)$
    - 外层循环迭代$V-1$次
    - 内层循环E次
    - 最后检查是否有负圈需要遍历每个边, $O(E)$

有向无环图中的DAG算法

- 步骤
    - 对节点进行拓扑排序
    - 按此顺序对各个节点的邻边进行`RELAX`操作
- $O(V+E)$

Dijkstra算法

- 功能
    - 每个节点的$v.d$都已经是s(源)到v的最短路径长度
- 要求
    - **不存在权重为负的边**
- 步骤
    - 每次从$V-S$中抽取出$v.d$最小的点加入S, 并更细$V-S$中每个点的d
- 时间复杂度

$$O(E * T_{DECREASE-KEY} + V * T_{EXTRACT-MIN})$$

    - 依赖于优先级队列的实现
    - 二项堆:$O((E+V)lgV)$
    - 斐波那契堆:$O(E+VlgV)$
        - 因为`DECREASE-KEY`操作有时间复杂度$O(1)$
- 空间复杂度
    - $O(V+E)$

所有点对的最短路径

回溯最短路径

- 只需要最终最短路径矩阵$w_{i,j}$的前驱结点矩阵$\pi_{i,j}$, 因为$i \to j$的最短路径中若有$i \to r \to j$, 则$i \to r$一定也是最短路径, 即有$\pi_{i,j} = r$, 因此可以递归地进行回溯

使用Dijkstra/Bellman

- 时间复杂度
  - $O(V * \phi)$
  - ϕ为某一种单源最短路径算法的时间复杂度

重复平方法

- 时间复杂度$\Theta(V^3 \lg V)$
- 空间复杂度$\Theta(V^2)$

Floyd-Warshell算法

- 要求
  - **不能存在权重为负的环路**
  - 可以存在权重为负的边
- 细节
  - 按照点来扩张路径, 而非按照矩阵
  - 不管是求距离长度还是回溯路径, 都只需要最后一个矩阵$D_{ij}^{(n)}$
- 时间复杂度
  - 最外层循环每次扩张一个点$k+1$；有$O(V)$
  - 内层两个循环遍历当前图上每个顶点之间最短路径, 更新；有$O(V^2)$
  - 总$\Theta(V^3)$
- 空间复杂度
  - 如果保存每个矩阵, 则需要$O(V^3)$
  - 如果只保存一个, 则$O(V^2)$

Johnson算法

- 步骤
  - 给边重新赋值, 保证没有负边:
    - $w'(i,j) = w(i,j) + h(u) - h(v)$
    - $h(u)$是使用`Bellman-Ford`算法求得的从新加入的虚拟节点s到u的单源最短路径
  - 用新权重对每个节点调用`Dijkstra`, 得到的伪最短路径$\delta'(i,j)$, 有

$$\delta(i,j) = \delta'(i,j) + h(j) - h(i)$$

- 时间复杂度
  - 斐波那契堆: $O(VE+V^2lgV)$
  - 二项堆: $O(V^2lgV+VElgV)$

最大流

基本概念

- **流**
  - 一个**函数, 将**$V\times V$**映射到**R**, 即**$f(u,v):V^2\rightarrow R$
- **流的值**

$$|f|=\sum_{v\in V}f(s,v)-\sum_{v\in V}f(v,s)$$

  - 后一项一般是0
  - **最大流**⇔**最大流的值**
- **流是流, 即**$f(u,v)\in R$**; 容量是容量, 即**$c(u,v)\in R$**; 流的值是流的值, 即**$|f(u,v)|\in R$

性质

- **流量限制**

$$\forall u,v\in V\quad 0\leq f(u,v)\leq c(u,v)$$

- **流量守恒**

$$\forall u\in V-\{s,t\}\quad \sum_{v\in V}f(u,v)=\sum_{v\in V}f(v,u)$$

剩余网络

- 对于一个流网络$G(V,E)$, 其源点s, 汇点t, f为G上的一个流, 定义

$$c_f(u,v)=\begin{cases}c(u,v)-f(u,v) & 若e(u,v)\in E\\ f(u,v) & 若e(v,u)\in E\\ 0 & 其他\end{cases}$$

- 则**由**f**诱导产生的剩余网络**G_f**为**$G(V,E_f)$, 其中

$$E_f=\{e(u,v)\mid u,v\in V,c_f(u,v)>0\}$$

增广流

- 在由流f诱导产生的剩余网络G_f上定义一个流$f' : V^2 \to R$, 记f的增广流为$f \uparrow f'$, **定义**

$$f \uparrow f'(u,v) = f(u,v) + f'(u,v) - f'(v,u)$$

- 则有$f \uparrow f'$是G中的一个流, 且

$$|f \uparrow f'| = |f| + |f'|$$

- 因此, **沿着增广流来增加流量可以使流的值变大**

增广路径

- **定义:** 残余网络G_f中的一个$s \to t$的最短路径
- **残余容量**
  - 一条增广路径p中能为每条边增加的流量的最大值, 即

$$c_f(p) = min\{c_f(u,v) \mid e(u,v) \in p\}$$

- **进一步, 在G_f上定义流**

$$f_p(u,v) = \begin{cases} c_f(p) & e(u,v) \in p \\ 0 & \text{其他} \end{cases}$$

  - 将f增加f_p的量, 得到$f \uparrow f_p$是G的一个流, 且$|f \uparrow f_p| = |f| + |f_p| = |f| + |c_p| > |f|$(依据定义$c_p > 0$)

切割

概念

- **横跨切割的净流量**

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

- **切割的容量**

$$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$$

性质

- 给定G上的一个流f, 则**横跨任何切割的净流量$f(S,T)$相同, 等于$|f|$**

- **流网络G上的任意一个流f的值$|f|$不能超过G的任意切割的容量**

最大流最小割定理

对于流网络G中的一个流f, 有以下三者等价

- **f是G的一个最大流**
- **残余网络G_f中不包含任何增广路径**
- **$|f| = c(S, T)$, 其中$c(S, T)$是流网络G的最小切割 (容量最小)**

Ford-Fulkson算法

- 步骤
  - 一直在残余网络中寻找增广路径, 如果找不到了, 则得到最大流
  - 当容量为无理数时, 可能不收敛
- 分析
  - 在残余网络中使用`DFS`/`BFS`搜索一条$s \rightarrow t$的路径, 有时间复杂度$O(V+E)$
  - 最终时间复杂度$O(Ef^*)$
    - f^*是将有理数容量$c = \frac{p}{q}$转化为p后的网络的最大流的值

Edmonds-Karp算法

- 步骤
  - 使用广度优先搜索寻找$s \rightarrow t$的**最短路径(基于路径上的边数判定而非基于流量)** 作为增广路径
- 时间复杂度$O(VE^2)$