定义都很自然且高中知识居多, 因此整理不会像树那么细, 简介算法为主

7 图

定义细节

- DAG (directed acyclic graph) : 有向无环图
- 无向图中两节点间有平行边不算环, (路径长度 ≥ 3,环=路径,不等式秒了)
  - 有向图两条边可以构成环
- 有根图 : 路径能到其他所有点
- 强连通 : 双方互有有向路径联通
  - 强连通分量 : 非强连通图的极大强连通子图
- 网络 : 带权的连通图
- adjacency matrix : 邻接矩阵
- 稀疏因子, $\delta < 5$ 认为是稀疏矩阵

$$\delta = \frac{非0元素个数(t)}{矩阵大小(m \times n)}$$

储存

- 邻接表节省时间空间
  - 无向图的邻接表表示, 同一边出现两次即可
  - 有向图也可以有逆邻接表

十字链表

- 每一弧 : 头, 尾, 下一条共尾弧, 下一条共头弧, 弧权值(info域)
- 每一顶点 : data域, 第一条以该顶点为终点的弧, 第一条以该顶点为始点的弧

> 稀疏矩阵的十字链表 : 行列指针序列 + 每个非0结点(值, 行后继, 列后继)
> 稀疏矩阵相乘 : 遍历A的行B的列即可.

遍历

实质上是节点的遍历

- 解决回路和非连通图问题 : 标志位

dfs, bfs

- 邻接表表示 : 有向图 $\Theta(n+e)$, 无向图 $\Theta(n+2e)$
- 相邻矩阵表示 : $\Theta(n^2)$

拓扑排序

- 对 DAG 不断删除 0 度边入队
- 邻接表表示 : $\Theta(n+e)$
- 相邻矩阵表示 : $\Theta(n^2)$

> 图算法需要考虑的问题: 有向无向, 回路, 连通性, 权值正负

算法

Dijkstra单源最短路径

分成已知最短路径和未知最短路径两组, 按长度递增将第二组节点逐个加入第一组

- 最坏 $\Theta((|V|+|E|)\log|E|)$, 但要具体分析, 维护优先队列只有找最小值和松弛两个操作.
- 不支持负权值, 支持负权值需要 Bellman-Ford 算法或者 SPFA 算法

Floyd算法求所有最短路径

实际上就是动态规划, 三层 for 循环即可

$$d[i][j]=\min\{d[i][j],d[i][k]+d[k][j]\}$$

- 复杂度 $\Theta(n^3)$

Prim算法求最小生成树

- MST (minimum-cost spanning tree)

框架与 Dijkstra 算法相同, 但距离值直接用最小边, 总时间 $O(n^2)$

- 适合稠密图, 对稀疏图可以像 Dijkstra 算法那样用堆来保存距离值.

Kruskal算法求最小生成树

边排序然后逐个入队, 遇到破坏连通性的就跳过
最坏情况是 $\Theta(\mathrm{e}\log\mathrm{e})$, 通常代价是 $\Theta(n\log\mathrm{e})$