Algorithm and Data structures Summary

ftiasch

September 9, 2010

1 Graph Theory

1.1 Minimum spanning tree

1.1.1 Prim's Algorithm

时间复杂度$O(V^2 + E)$或$O((V + E)logV)$。

1.1.2 Kruskal's Algorithm

时间复杂度是$O(ElogE)$。根据Kruskal算法的流程可知，最小生成树不仅是边权的和最小的生成树，而且是边权最大的边最小的生成树。

1.1.3 Cut property

树$T = (V, E)$是最小生成树$\Leftrightarrow \forall e \in E, e' \notin E, w(e) \leq w(e')$。利用环切性质，可以通过不断添加新边同时删去形成的环上的最大边得到生成树。另一方面，可以通过环切性质检验最小生成树。

1.1.4 Second minimum spanning tree

可以证明次小生成树和最小生成树互为邻树，所以只需根据环切性质进行最小差额添删操作即可。算法的瓶颈在于查询树上两点间的最大边，朴素的实现是$O(N^2)$的，事实上利用Tarjan算法可以$O(V + E)$出解。不计求解最小生成树，时间复杂度$O(V + E)$。

1.1.5 Zhu-liu's Algorithm

每个点选取最小的入边，如果出现环则收缩环，将边权改为替换环上最大边的差额，否则结束。注意去掉自环。时间复杂度是$O(VE)$。

1.1.6 Degree constrained spanning tree

除单点度限制生成树外，其他度限制生成树问题是NP-完全问题。首先删去有度限制的节点V_0，求得其他节点的最小生成树。类似于次小生成树，不断地添加边(V_0, V_i)并利用环切性质。不计求解最小生成树，时间复杂度$O(dV)$。

1.2 Shortest path

1.2.1 Triangle inequality

$\forall(u,v) \in E, d[u]+w[u][v] \geq d[v]$。显然利用最短路算法可解形如$x_i + w_{i,j} \geq x_j$的不等式组问题。

1.2.2 Dijkstra's Algorithm

适用于正权图，由于每次更新距离标号单调不减，故最小标号不再更新。时间复杂度$O(V^2+E)$或$O((V+E)logV)$。

1.2.3 Floyd-warshall Algorithm

设$f^k[i][j]$表示V_i经过$(V_1, V_2, \ldots, V_k)$到达V_j的最短路径，初始时$f^0[i][j] = w[i][j]$。递推时枚举k作为中间点，方程$f^k[i][j] = min\{f^{k-1}[i][k]+f^{k-1}[k][j]\}$。时间复杂度$O(V^3)$。

1.2.4 Minimum cycle

作为Floyd算法的一个扩展，最小环问题可以通过枚举环中标号最大的点V_k，再枚举V_i和V_j，并用$w[k][i] + f^{k-1}[i][j] + w[j][k]$更新答案。时间复杂度$O(V^3)$。

1.2.5 Bellman ford Algorithm

适用于任意图，可以判定图中是否存在负环。检查所有边，如果不满足三角形不等式则松弛。当循环超过V次则说明存在负环，因为显然最短路经过不超过V个节点。时间复杂度是$O(VE)$。

1.2.6 Shortest path faster Algorithm

对于Bellman ford算法的一个浅显的改进，利用队列保存待更新的节点。时间复杂度是$O(VE)$，但是实际效果良好。作为一个迭代算法，同时可以用于求解方程的近似解，或者阶段性不明显的最优化问题。

1.3 Lowest common ancestor

1.3.1 Tarjan's Algorithm

离线算法，借助并查集完成。不仅可以解决LCA问题，而且可以解决没有修改的路径问题。时间复杂度$O(V+Q)$。

1.3.2 Doubling Algorithm

在线算法。预处理$f^k[i]$表示V_i的2^k级祖先，则可以$O(logV)$地得到节点的任意级别祖先。先将节点深度统一，再同时向上直到恰好重合。实现时注意两个点恰好重合的情况。时间复杂度$O(VlogV) - O(logV)$。

1.3.3 Transformation to ± 1 Range minimum qurey

求解LCA问题可以转化为求解以深度为关键字的欧拉序列上的± 1 RMQ问题。但是大部分情况下没有必要。时间复杂度是$O(V) - O(1)$。

1.4 Bipartite graph

1.4.1 Coloring property

图G是二分图等价于图G可以2染色。如果块中存在一个奇环，则块中的节点都在至少一个奇环上。

1.4.2 Hungarian Algorithm

二分图最大匹配。时间复杂度$O(V(V+E))$。

1.4.3 Kuhn-Munkras Algorithm

设立顶标$\{L_i\}$满足$\forall(X_i,Y_j)\in E, L_{X_i}+L_{Y_j}\geq w(X_i,Y_j)$。取$G'=(V,E'), E'\subset E, \forall(X_i,Y_j)\in E', L_{X_i}+L_{Y_j}=w(X_i,Y_j)$。容易看出不等式的等号成立当且仅当$G'$存在完备匹配，而该匹配显然是$G$的最大匹配。初始时可以设$L_{X_i}=max\{w(X_i,Y_j)\}, L_{X_j}=0$，当寻找增广链$P$失败时，令$\delta=min\{L_{X_i}+L_{Y_j}-w(X_i,Y_j)|X_i\in P, Y_j\notin P\}$，同时若$X_i\in P$则令$L'_{X_i}=L_{X_i}-\delta$，若$Y_j\in P$则令$L'_{Y_j}=L_{Y_j}+\delta$。容易验证能够保留扩大相等子图。时间复杂度是$O(V(V+E))$。实际算法效率比理论复杂度的$O(V^3)$优秀得多，$1s$可以支持$N=10^3$的数据范围。相比费用流算法只能支持$N=200$显得优秀很多。

值得注意的是$0\leq L_i$且$\sum L_i$最小，因此可以用于求解一类满足$L_{X_i}+L_{Yj}\geq w(X_i,Y_j)$同时最小化$\sum L_i$的线性规划。

1.4.4 Minimum-vertex-covering

新建源点S和汇点T，新建$S\rightarrow X_i$，容量为W_{X_i}，新建$X_i\rightarrow Y_j$，容量为∞，新建$Y_j\rightarrow T$，容量是W_{Y_j}。求解最小割C，对于一条边(X_i,Y_j)，必然有$(S,X_i)\in C$或$(Y_j,T)\in C$成立，因此割集和点覆盖集是一一对应的。因为点覆盖集和点独立集互为补图，所以同时可以求出最大点独立集。

考虑一个特殊情况，当$\forall W_i=1$时，最大流等价于最大匹配，最小割等价于最小点覆盖，根据最大流值等于最小割值，得到最大匹配数等于最小点覆盖数，也即Konig's定理。

1.4.5 Minimum-edge-covering

最小边覆盖数等于顶点数减去最大匹配数。

1.5 Directed acyclic graph

1.5.1 Topology sorting

广度优先遍历。时间复杂度$O(V+E)$。如果用优先队列替换队列，则可以求出字典序最小的。

1.5.2 Minimum-path-covering

拆点转化为最大匹配求解。如果不要求路径不相交，则传递闭包后求解即可。如果允许多次经过，则可以利用网络流求解。

1.6 Biconnected component and Strongly connected component

1.6.1 Depth first search tree

无向图深度优先搜索树的重要性质是不会有横叉边，所导致的最好的结果是后向边只会影响儿子到它的祖先的一段路径。从这个意义上考虑，节点标号不一定要采用节点发现的序号，只需要可以分辨祖先关系即可，例如取深度也是可行的。

1.6.2 Bridge and cut point

边(u,v)为桥$\Leftrightarrow dfn[u] < low[v]$，点$u$为割点$\Leftrightarrow \exists v, dfn[u] \leq low[v]$。如果需要计算一条边属于多少个圈，则可以顺便作树形动态规划。以上算法复杂度都是$O(V+E)$。

1.6.3 Biconnected component

在深度优先遍历地过程中，如果(u,v)不是桥且满足$dfn[u] = low[v]$则可以导出一个块。将非双连通图变为双连通图需要添加的边数为(叶子数+1)/2。

1.6.4 Strongly connected component

类似于无向图，当$dfn[u] = low[u]$时可以导出一个强连通分量。将非强连通图变为强连通图需要添加的边数为$max\{$最低强连通分量数, 最高强连通分量数$\}$。

1.7 Euler tour

1.7.1 Euler tour on undigraph

圈套圈算法。时间复杂度$O(V+E)$。

1.7.2 Euler tour on mixed graph

将无向边任意定向，之所以不能直接拆成两条的原因是不能保证无向边必选。统计每个点的入度和出度利用流量平衡构造网络求解。

1.8 Network flow

1.8.1 Dinic's Algorithm

利用广度优先搜索求出残量网络的最短路图，再在最短路图上利用深度优先搜索多路增广。时间复杂度$O(V^2E)$，实际上，E对于算法的效率影响较大。$1s$能支持数据范围在$N = 10^4, M = 10^5$左右。对于单位容量网络，时间复杂度$O(E\sqrt{V})$。

1.8.2 Maximum flow-Minimum cut Theorem

最大流-最小割定理作为强对偶定理的特殊形式，事实上等价于Dilworth's定理、Konig's定理和Hall's定理。一般来说，讲节点划分为两个集合的题目，都可以通过补集转化得到最小割模型。

1.8.3 Minimum cost maximum flow

每次增广费用最小的增广路，可以证明当前的流总是当前流值下的最小费用流。另外一个问题是可以证明如果原图没有负费用环，则增广过程中不会出现负费用环。该结论可以看作是前面结论的显然推论，因为沿着一个负费用环增广会使费用减小，这与前面的论断矛盾。此外这个结论是消圈算法的理论支撑。

1.8.4 Network flow with lower bound constrained

通过新建弧$T \rightarrow S$，容量为$(0, \infty)$，可以转化为无源汇网络。新建虚源S'和虚汇T'并分离下界可以判定是否有可行流。为了求得最大流或最小流，那么可以分别二分查找$T \rightarrow S$的下界和上界，再判定是否有可行流即可。时间复杂度和无下界的网络流的复杂度相同。

2 String Algorithm

2.1 Suffix Array

倍增算法的复杂度理论上是$O(NlogN)$，而且大家都说算法的常数很大。对于一个朴素版本的倍增算法，可以支持到$N = 200,000$的范围，已经十分可观。

2.2 Aho-Corasick Automation

在自动机中，如果没有记忆化地保存每个节点的转移，则在求解失败指针的时候可能退化为$O(N^2)$（当然实际上由于字符集较小退化不是特别明显），记忆化后时间和空间复杂度都是$O(NS)$。特别地，作为自动机单串版本的KMP没有这个问题。