~~浅谈~~信息学竞赛中的独立集问题

二、定义及约定

定义 2.1. 对于无向图 $G=(V,E)$ 和点 $u,v\in V$，若 $(u,v)\in E$，则称 u,v **相邻**（*adjacent*）；定义点 $v\in V$ 的**邻域**（*neighborhood*）为 V 中与 v 相邻的结点集合，记为 $N(v)$；另外，$N_G(v)$ 表示 v 在图 G 中的邻域。

定义 2.2. 点 v 的**度**（*degree*）$\deg(v)$ 定义为 $N(v)$ 的大小，即 $\deg(v)=|N(v)|$；另外，$\deg_G(v)$ 表示 v 在图 G 中的度。

定义 2.3. 无向图 $G=(V,E)$ 的一个**独立集**（*independent set*）定义为 V 的一个子集，满足子集中的结点两两不相邻。形式化地，I 是 G 的一个独立集，当且仅当 $I\subseteq V$ 且 $\forall u,v\in I$，$(u,v)\notin E$。

定义 2.4. 无向图 $G = (V, E)$ 的一个**最大独立集**（*maximum independent set*）是指 G 中所含结点数 $|I|$ 最多的独立集 I。

定义 2.5. 无向图 $G = (V, E)$ 的**独立数**（*independence number*）定义为 G 的最大独立集 I 所含的结点数 $|I|$，记为 $\alpha(G)$。

定义 2.6. 无向图 $G = (V, E)$ 在 $S \subseteq V$ 上的**导出子图**（*induced subgraph*）[8]定义为以 S 为点集，两端点都在 S 内的边为边集构成的图，记为 $G[S]$。

三、一般图的独立集问题

1. 基于极大独立集搜索的独立集算法

1.1 朴素的搜索算法

考虑剪枝：

1. 若 $\deg(v) = 0$，则不存在与 v 关联的边，故总可以令 $v \in I$。

2. 若 $\deg(v) = 1$，考虑唯一的与 v 关联的结点 u，若 $u \notin I$，则总可以令 $v \in I$；否则，从 I 中删去 u 并加入 v，I 的大小不变。因此总可以令 $v \in I$。

3. 搜索时记录当前搜到的独立集的大小的最大值 a，记 P 为 $V - I$ 中不与 I 中结点相邻的点集，当 $|I| + |P| \leqslant a$ 时可进行最优性剪枝。

~~然而并没有什么用~~

三、一般图的独立集问题

1. 基于极大独立集搜索的独立集算法

1.2 极大独立集与Bron-Kerbosch算法

定义 3.1. 无向图 $G=(V,E)$ 的一个**极大独立集**（*maximal independent set*）是指 G 的一个独立集 I，满足对于任意的结点 $v \in V-I$，点集 $I+\{v\}$ 不是独立集。

一个定理：每个最大独立集都是极大独立集。（不然类

通常情况下，一个图的极大独立集个数比独立集个数少得多。

因此，可以找出 G 的所有极大独立集，来找 G 的最大独立集。

Bron-Kerbosch算法可以对任意的无向图 G 求出所有的极大独立集

调用 BronKerbosch(R, P, X)，将输出包含 R 中的所有结点、P 中的任意多个结点且不包含 X 中的结点的所有极大独立集。

调用 BronKerbosch(Φ, V, Φ) 即可得到所有极大独立集。

算法 1 BronKerbosch(R, P, X)

if $P = X = \varnothing$ **then**

  print R

end if

选择结点 $u \in P \cup X$，使得 $|P \cap (\{u\} \cup N(u))|$ 最小

for all $v \in P \cap (\{u\} \cup N(u))$ **do**

  BronKerbosch($R \cup \{v\}, P - (\{v\} \cup N(v)), X - (\{v\} \cup N(v))$)

  $P \leftarrow P - \{v\}$

  $X \leftarrow X \cup \{v\}$

end for=0

算法 1 BronKerbosch(R, P, X)

if $P = X = \varnothing$ **then**
  print R
end if
选择结点 $u \in P \cup X$，使得 $|P \cap (\{u\} \cup N(u))|$ 最小
for all $v \in P \cap (\{u\} \cup N(u))$ **do**
  BronKerbosch$(R \cup \{v\}, P - (\{v\} \cup N(v)), X - (\{v\} \cup N(v)))$
  $P \leftarrow P - \{v\}$
  $X \leftarrow X \cup \{v\}$
end for=0

R：可以看作记录答案的变量

P∪X：递归中的图G（每选择一个结点u，将{u} ∪ N(u)删去，其他结点不受影响）

X：防止重复计算

“选择结点 u ∈ P ∪ X，使得 | P ∩ ({u} ∪ N(u)) | 最小”：这个是Pivoting过程：

一个定理：对于无向图 G = (V, E) 和 u ∈ V，G 的任意极大独立集 I 满足 I ∩ ({u} ∪ N(u)) ≠ Φ

对于P∪X中的某一结点u，{u} ∪ N(u)至少有一个结点属于 I 。

三、一般图的独立集问题

1. 基于极大独立集搜索的独立集算法

1.3 极大独立集的个数

一个定理： **定理 3.3.** *Bron-Kerbosch*算法的递归调用次数为 $O(3^{\frac{n}{3}})$。

该定理的证明较为复杂，本文略去[12]。

所以

一个定理：n 阶无向图的极大独立集个数为 O(3 ^ (n / 3))

这个上界是很容易达到的，构造 n/3 个相互独立的三元环即可。

但在图随机生成的情况下，这个上界是很不满的。

~~为了说明这一点，作者花了不小篇幅对随机图的极大独立集个数进行了研究。~~

从边数为 m 的 n 阶简单无向图中随机生成一个图 G = (V, E)

那么， S 是 G 的极大独立集的条件为：

1、 S 是独立集，即对于任意的 u, v ∈ S，(u, v) ∉ E；

2、对于任意的 v ∈ V − S，V + {v} 不是独立集，即 V − S 中的每个点至少与 S 中的一个点相邻。

用容斥原理，枚举 k 个 V − S 中的点不与 S 中的点相邻。

记 i = |S|，则剩下 n − i − k 个点可以和 S 中的点连边，以及 V − S 中任意两点（一共 (n − i)(n − i − 1)/2 对点）可以连边。

则满足 S 是极大独立集的图 G 个数为

$$\sum_{k=0}^{n-i}(-1)^k\binom{n-i}{k}\binom{(n-i-k)i+\frac{(n-i)(n-i-1)}{2}}{m}$$

记 G 的极大独立集个数为 x， x 的期望值为 E(x)

由于边数为 m 的 n 阶简单图共有 C(n(n-1)/2, m) 个，故有

$$E(x) = \sum_{S \subseteq V} P([S \text{ is a maximal independent set}])$$

$$= \sum_{i=0}^{n} \sum_{S \subseteq V, |S|=i} \binom{\frac{n(n-1)}{2}}{m}^{-1} \sum_{k=0}^{n-i} (-1)^k \binom{n-i}{k} \binom{(n-i-k)i + \frac{(n-i)(n-i-1)}{2}}{m}$$

$$= \binom{\frac{n(n-1)}{2}}{m}^{-1} \sum_{i=0}^{n} \binom{n}{i} \sum_{k=0}^{n-i} (-1)^k \binom{n-i}{k} \binom{(n-i-k)i + \frac{(n-i)(n-i-1)}{2}}{m}$$

$E(x)$	$m = n$	$m = \lfloor \sqrt{3}n \rfloor$	$m = 2n$	$m = 3n$	$m = \frac{n^2}{4}$
$n = 20$	84	101	99	81	49
$n = 30$	706	933	909	691	157
$n = 40$	5.95×10^3	8.67×10^3	8.40×10^3	5.88×10^3	403
$n = 50$	5.02×10^4	8.07×10^4	7.76×10^4	5.01×10^4	891
$n = 60$	4.23×10^5	7.51×10^5	7.18×10^5	4.27×10^5	1779
$n = 70$	3.57×10^6	6.99×10^6	6.64×10^6	3.63×10^6	3291
$n = 80$	3.01×10^7	6.51×10^7	6.14×10^7	3.10×10^7	5730
$n = 90$	2.54×10^8	6.06×10^8	5.68×10^8	2.64×10^8	9506
$n = 100$	2.15×10^9	5.64×10^9	5.25×10^9	2.25×10^9	15154

可见随机情况下，极大独立集的个数远少于 3 ^ (n / 3)

三、一般图的独立集问题

1. 基于极大独立集搜索的独立集算法

1.4 应用

例 1. (图的3-染色问题)

给定 n 阶简单无向图 G = (V, E)，用三种颜色对 V 中的结点进行染色，使得每条边 (u, v) ∈ E 的两端点颜色不同。n ≤ 40。

一个定理：无向图 G = (V, E) 能够3-染色的充要条件是 G 存在一个极大独立集 I，使得图 G−I 是二分图。

solution:

那我们可以找出所有的极大独立集 I，然后判断 G-I 是不是二分图就好了

复杂度：O(3^(n/3) * n^2)

例 2. (WC2013 小Q运动季 测试点10)
给定一个 n 元一次同余方程组，求一组解 (x1, x2, …, xn) 满足尽量多的方程。提交答案题。

$$\begin{cases} a_{0,0}x_0 + a_{0,1}x_1 + \ldots + a_{0,n-1}x_{n-1} \equiv c_0 \pmod{b_0} \\ a_{1,0}x_0 + a_{1,1}x_1 + \ldots + a_{1,n-1}x_{n-1} \equiv c_1 \pmod{b_1} \\ \ldots \\ a_{m-1,0}x_0 + a_{m-1,1}x_1 + \ldots + a_{m-1,n-1}x_{n-1} \equiv c_{m-1} \pmod{b_{m-1}} \end{cases}$$

n=50，m=90。

solution:
该测试点中，通过建立图论模型，将每个方程看成一个点，相互冲突的方程间连一条边，可以转化为点数 n = 90，边数 m = 223 的无向图的最大独立集问题。

由于具体转化过程超出了本文的范围，故略去。

三、一般图的独立集问题

2. 基于动态规划的独立集算法

2.1 算法

两个定理：

对于无向图 $G = (V, E)$ 和 $V' \subseteq V$，则对于任意 $I \subseteq V'$，I 是 G 的独立集当且仅当 I 是 $G[V']$ 的独立集。

对于无向图 $G = (V, E)$ 和 $v \in V$，若 $I \subseteq V$ 且 $v \in I$，则 I 是 G 的独立集当且仅当 $I - \{v\}$ 是 $G[V - \{v\} - N(v)]$ 的独立集。

根据以上两个定理，我们可以用状态压缩的DP对于任意的无向图 $G = (V, E)$ 求出 G 的独立数 $\alpha(G)$。

对点集 $S \subseteq V$，定义 $f(S)$ 为 S 在 G 上的导出子图的独立数，即 $f(S) = \alpha(G[S])$。

显然 $f(\Phi) = 0$。

考虑 $S \neq \Phi$ 的情况：任取 $v \in S$，考虑一个点集 $I \subseteq S$。
若 $v \notin I$，则 I 是 $G[S]$ 的独立集当且仅当 I 是 $G[S - \{v\}]$ 的独立集；
若 $v \in I$，则 I 是 $G[S]$ 的独立集当且仅当 $I - \{v\}$ 是$G[S - \{v\} - N(v)]$ 的独立集。

所以：

$$f(S) = \begin{cases} 0, & S = \varnothing \\ \max\{f(S - \{v\}), f(S - \{v\} - N(v)) + 1\}, \forall v \in S, & S \neq \varnothing \end{cases}$$

实现时，结点 v 可以选取 S 中编号最大的点。
同样可以使用压位技巧来存集合 S。
另外，计算 f 可以使用记忆化搜索。

复杂度为 O(2^(n/2))。（构造 n/2 个相互独立的二元连通分量）

该算法不仅能求出独立数，还能求出一个最大独立集

算法 2 Subset-Dynamic-Programming

$S = V$
$I = \varnothing$
while $S \neq \varnothing$ **do**
  令 v 为 S 中编号最大的点
  if $f(S - \{v\}) > f(S - \{v\} - N(v)) + 1$ **then**
    $S \leftarrow S - \{v\}$
  else
    $S \leftarrow S - \{v\} - N(v)$
    $I \leftarrow I + \{v\}$
  end if
end while
return I =0

例 3. (团的计数)

给定无向简单图 G = (V, E)，求 G 有多少个团。
一个团定义为一个点集S ⊆ V，满足 S 中任意两点都有边相连。n ≤ 50。

记 G' 为 G 的补图，不难发现，S 是 G 的团当且仅当 S 是 G' 的独立集。

显然这类计数问题无法用搜索优化的策略，不过，使用上述的Subset-Dynamic-Programming算法即可在 O(2^(n/2)) 时间内解决问题。

$$f(S)=\begin{cases}1, & S=\Phi\\ f(S-\{v\})+f(S-\{v\}-N(v)), & S\neq\Phi\end{cases}$$

三、一般图的独立集问题

2. 基于动态规划的独立集算法

2.2 效率优化

类比优化搜索算法，我们用一些“剪枝”来优化上述DP。
以下优化可以大大提高DP的效率：

1、在状态转移方程中，结点 v 不取 S 中编号最大的点，而取 G[S] 中度数最大的点；

2、当图 G[S] 不连通时，记每个连通块的点集分别为 S1, S2, …, Sk，由于每个连通块是独立的，可以转化为规模更小的子问题解决：

$$f(S) = \sum_{i=1}^{k} f(S_i)$$

3、当图 G[S] 不含环时，可以改用树形DP求解。

通过对随机图的测试，优化后的DP比之前的DP快得多，其中优化1、2效果明显，尤其对于较稀疏的图。这是因为较稀疏的图在不断删去度数大的点时，导出子图 G[S] 很容易不连通。

三、一般图的独立集问题

2. 基于动态规划的独立集算法

2.3 与搜索算法的联系

然而该DP有一定的缺陷：空间复杂度比较大。

而搜索算法的空间是多项式级别的，支持运行较长时间。

因此可以只记忆化较小的 S 的 f (S) 值，剩余部分采用搜索的方法，这样就能在较低的空间需求下解决问题了。

三、一般图的独立集问题

2. 基于动态规划的独立集算法

2.4 测试与对比

算法	40, 60	50, 85	60, 120	90, 223
Simple-Search	2s	–	–	–
Maximal-Search	< 0.01s	< 0.1s	1s	–
Subset-Dynamic-Programming	< 0.01s	< 0.1s	1s	–
Optimized-Subset-Dynamic-Programming-1	< 0.01s	< 0.01s	< 0.01s	1s
Optimized-Subset-Dynamic-Programming-2	< 0.01s	< 0.01s	< 0.01s	1s

我们用随机图来测试一下各个算法的运行效率：

1：朴素搜索
2：基于极大独立集搜索的Bron-Kerbosch算法——剪枝
3：动态规划——记忆化
4：带有优化1（度数）&优化2（连通块）的动态规划——剪枝+记忆化
5：带有优化1、2、3（树形DP）的动态规划——优化3效果一般

四、特殊图的独立集问题

1. 基于图匹配思想的最大独立集算法

1.1 二分图的最大独立集

一个定理：

对于 n 阶二分图 G，$\alpha(G) = n - \nu(G)$，其中 $\alpha(G), \nu(G)$ 分别为图 G 的独立数和匹配数。

四、特殊图的独立集问题

1. 基于图匹配思想的最大独立集算法

1.2 无爪图的最大独立集

定义集合 A, B 的对称差：$A \Delta B = \{x \mid [x \in A] \neq [x \in B]\}$

求二分图的最大匹配/最大独立集时，可以通过找增广路不断增加匹配大小。

那么求一般图的最大独立集能否用增广的方式？
然而，在任意图上，两个独立集的对称差的导出子图不一定是若干条路径或环，
所以并不能用找增广路的方法求最大独立集。

不过，无爪图可以。

无爪图：为所有导出子图都不是爪的无向图。
爪：即两部分别含有 1 个点和 3 个点的完全二分图。

一个定理：设 I1, I2 为无爪图 G 的两个独立集，则 G[I1ΔI2] 的每一个连通块都是一条简单路径或简单环。

注意到如果 |I1| < |I2|，那么 G[I1ΔI2] 必然存在一个连通块C，满足连通块中属于 I2 的结点比属于 I1 的结点多。由于C中属于 I1, I2 的结点交替出现，而C为简单环时，C中属于 I1, I2 的结点一样多，所以C为简单路径，C中属于 I1, I2 的结点个数相差1。

故必然存在一条路径满足属于 I2 的点数比属于 I1 的点数多1。我们把C称为 I1 的增广路。

这样类比一般图最大匹配的算法，用增广路算法求无爪图 G = (V, E) 的最大独立集：
初始时令 I = Φ，每次从一个点出发找一条增广路，然后将增广路上的点状态取反，
即：原来不属于独立集的点加入独立集，原来属于独立集的点从独立集中删去。

四、特殊图的独立集问题

2. 基于图上阶段划分思想的最大独立集算法

2.1. 分层图上的动态规划

对于图 G = (V, E)，将点集 V 划分为 k 个不相交的集合 V1, V2, …, Vk，
使得对任意u∈Vi, v∈Vj，若 |i − j| > 1，则 (u, v)∉ E，
则称集合序列 < V1, V2, …, Vk > 是 G 的一个分层。

如果每个Vi中的结点都不多，那么可以按V1, V2, …, Vk顺序进行决策，
在每个阶段只需状压一个层的选取情况即可，效率远高于一般图中的对整个图状压DP。

记 f(i, S) 为图G[V1∪V2∪…∪Vi]中包含S为子集的最大的独立集，状态转移方程如下：

$$f(i,S)=\begin{cases}-\infty, & S \text{ is not independent},\\ 0, & i=0,\\ \max\{f(i-1,S')\mid S'\subseteq V_{i-1}, S\cup S' \text{ is independent}\}+|S'|, & \text{otherwise}\end{cases}$$

复杂度：$O(\sum_{i=1}^{k-1} 2^{|V_i|+|V_{i+1}|})$

四、特殊图的独立集问题

2. 基于图上阶段划分思想的最大独立集算法

2.2. “k-仙人图”上的动态规划

例 4. (LYDSY 4316)
给定简单无向图 G = (V, E)，保证每条边最多属于一个简单环，求 G 的独立数。
$|V| \le 50{,}000$，$|E| \le 60{,}000$。

如果 G 不连通，那么求出 G 的每个连通块的独立数并求和即可。

假设 G 是连通图。
每条边最多属于一个简单环，那么G是一个仙人掌。

我们任选一个 $r \in V$，以 r 为根对图G进行深度优先搜索，得到一个DFS树 $T = (V, ET)$。
显然 T 是 G 的一个生成树。

定义树边为属于 ET 的边，非树边为不属于 ET 的边。

一个定理：
对任意非树边 $e = (u,v)$，在 T 中 u 是 v 的祖先，或者 v 是 u 的祖先。

对于非树边 $e = (u,v)$，若树边 e' 在树 T 中从 u 到 v 的简单路径上，则称树边 e' 被非树边 e 覆盖。

对于仙人掌，我们有：
每条树边最多被一条非树边覆盖。

假设G是一个树，可以记 f(i, 0) 为 i 为根的子树内最大的独立集大小，
f (i, 1) 为 i 的父结点属于独立集的情况下，i 子树内最大的独立集大小。

然而当G是仙人掌时，这样不能保证非树边的两端点不同时属于独立集。

我们可以加一维状态：
记 f(i, 0/1, 1)：i与父亲间的树边被一条非树边ei'=(ui, vi)覆盖，非树边的低深度结点ui属于独立集
记 f(i, 0/1, 0)：非树边对 i 是否属于独立集不产生影响

当 $j = 1$ 或 $k = 1 \land i = v_i$ 时：

$$f(i, j, k) = \sum_{c \in \mathrm{Ch}_i} f(c, 0, [e'_c = e'_i]k)$$

否则：

$$f(i, j, k) = \max\{\sum_{c \in \mathrm{Ch}_i} f(c, 0, [e'_c = e'_i]k), 1 + \sum_{c \in \mathrm{Ch}_i} f(c, 1, [e'_c = e'_i]k + [u_c = i])\}$$

我们提前O(m)预处理好所有低深度结点ui和高深度结点vi，就可以O(n)动态规划求出最大独立集了。

例 5.
给定简单无向图 $G = (V, E)$，保证每条边最多属于k个简单环，求 G 的独立数。

每条边最多属于 k 个简单环的图称为“k-仙人图”。

求解k-仙人图 $G = (V, E)$ 的独立集问题时，同样先取一个点为根对图进行DFS，得到DFS树 T。
接下来对每个点 i，记 C_i 为覆盖 i 与其父结点的连边 e_i 的非树边集合，则有一个性质：

在k-仙人图中，对于任意的 $i \in V$，$|C_i| \le k$。

记 f(i, j, S)：i 的父结点属于/不属于独立集，且 Ci 中的边的低深度结点构成的集合 Ui 中属于独立集的结点集合为 S 的情况下，T中以 i 为根的子树内最大的独立集大小。
则状态转移方程如下：

若 j = 1 或 S∪{ i } 不是独立集：

$$f(i,j,S)=\sum_{c\in Ch_i} f(c,0,S\bigcap U_c)$$

否则：

$$f(i,j,S)=\max\{\sum_{c\in Ch_i} f(c,0,S\bigcap U_c),1+\sum_{c\in Ch_i} f(c,1,(S\bigcup\{i\})\bigcap U_c)\}$$

当 k 视为常数时，该算法的复杂度为 O(n)。

事实上，只要 $\sum_{v\in V} 2^{|C_v|}$ 不大，这个算法的效率都是很高的。

例 6. (UOJ 259 测试点7,8)
给定无向图 $G = (V, E)$，$|V| = n$，$|E| = m$。求大小为k的独立集个数。由于答案可能很大，只需输出答案对p取模的结果。提交答案题。

这两个测试点满足 G 是连通图。

测试点编号	$n =$	$m =$	$k =$	$p =$
7	4998	5002	666	1000000009
8	11986	12011	1098	1000000007

由于 G 是连通图，且 $m - n$ 很小，可以发现 G 可以通过往一个树中加入 $m - n + 1$ 条边得到。

记：

- C_i 为覆盖 i 与其父结点的连边的非树边 e_i' 的低深度端点集合
- f(i, j, S, s)：C_i 中属于独立集的结点集合为S，i 的前 j 个子树中，大小为 s 的独立集个数
  边界：f(i, 0, S, 0) = 1
- g(i, j, S, s)：C_i 中属于独立集的结点集合为S，i∈S，i 以及 i 的前 j 个子树中，大小为 s 的独立集个数
  边界：g(i, 0, S, 0) = 0，g(i, 0, S, 1) = 1
- a(i, S, s)：C_i 中属于独立集的结点集合为S，以 i 为根的子树中，大小为 s 的独立集个数
- F(i, S, s) = f (i, t_i, S, s)，G(i, S, s) = g(i, t_i, S, s)，这里 t_i 为 i 的子节点数

状态转移：

对于 f，设 i 的前 j−1 个子树内共有 s1 个点，第 j 个子树内有 s2 个点，第 j 个子节点为 cj，则

$$f(i,j,S,s)=\sum_{x=\max\{s-s_1,0\}}^{\min\{s,s_2\}} f(i,j-1,S,s-x)a(c_j,S\bigcap C_{C_j},x)$$

对于 g，如果存在非树边e，使得 ue∈S 且 ve = i，那么 i 不能属于独立集，有

$$g(i,j,S,s)=0$$

否则

$$g(i,j,S,s)=\sum_{x=\max\{s-s_1,0\}}^{\min\{s-1,s_2\}} g(i,j-1,S,s-x)F(c_j,(S\bigcup\{i\})\bigcap C_{c_j},x)$$

对于 a，有

$$a(i,S,s)=G(i,S,s)+F(i,S,s)$$

最后的答案就是 a(root, Φ, k)。

可以只计算满足 s ≤ k 的状态，复杂度$O(k \sum_{v \in V} 2^{|C_v|})$。

内存可以动态分配。

值得注意的是，选取不同的点 r∈V 当根，以及用不同的顺序进行DFS，运行效率是不同的。
可以选择一个根 r 进行DFS，使得复杂度最小，然后再执行上述算法。

五、总结

NP-Hard问题的算法优化方法数不胜数，本文仅仅提到了若干种独立集问题的优化算法，这些方法解决的问题相类似，但思想各有区别——针对普通的最优化问题（如最大独立集），可以用带最优性剪枝的搜索算法减少枚举量；针对计数或有额外约束的问题（如独立集计数），可以用状态压缩动态规划，通过优化状态数来提高运行效率；针对可“增广”的图以及具有明显阶段性的图，又可以用多项式复杂度的算法来高效完成。

同时，本文对几种算法在随机情况下的运行时间进行了分析和比较，让大家对独立集问题求解的效率有更进一步的认识。