6-4：

> *考虑最大团问题的子集空间树中第 i 层结点 x，设 $minDegree(x)$ 是结点 x 所选择的最小团中结点度数的最小值。*
>
> *(1) 设 $x.u = min\{x.cn + n - i + 1, minDegree(x) + 1\}$，证明以结点 x 为根的子树中任一叶结点所相应的团的大小不超过 $x.u$，依此 $x.u$ 的定义重写算法 $bbMaxClique$。*
>
> *(2) 比较新旧算法所需的计算时间和产生的排列树结点数。*

题目分析

为方便阅读，将题设中的变量名进行解释或修改：

- $x.cn = cliqueSize$，即该结点当前团的顶点数，为方便，下记为 cnt，初始值为0
- n 为总结点数，$i = level$ 为该结点在自己空间树中所处的层数，下记为 $level$，初始值为1

(1)：

证明：

若当前已知 $cnt, level$，即团中已有 cnt 个结点，当前在第 $level$ 层，则剩下 $n - level$ 个结点，若包括第 $level$ 层之后的结点均包括在内，则团的大小为 $cnt + n - level + 1 \leq x.cn + n - i + 1$ 个；而由 $minDegree(x)$ 的定义知，当前已选取的团中最小的结点度数保证这个团最多只有 $minDegree(x) + 1$ 个结点（否则易反证，该结点集不是团），不超过 $minDegree(x) + 1$，因此：

$$upperSize = min\{x.cn + n - i + 1, minDegree(x) + 1\} \tag{1}$$

Q.E.D.

对新算法，我们可以取上式的 $upperSize$ 作为上界函数，取新的上界函数判断式为：

$$upperSize > bestn \tag{2}$$

若该式成立，则说明子树中可能存在最优解，需要继续搜索下去。

同时，为了减少新算法生成的活结点数量，我们适当修改优先队列的判断，令 $upperSize$ 在相同的时候，堆顶优先放置 cnt 较大的结点，尽可能先找出一组可能的优解。

代码实现

只是重构一下程序，应该不需要我贴代码吧？

输出示例

输出格式为：

- 输入一行两个整数 n, m，其中 n 为结点数，m 为边数
- 接下来 m 行，每行两个整数 $i, j (1 \leq i, j < \infty)$，表示 i, j 有边相连。

```
输入1:
5 8
1 2
1 4
1 5
2 3
2 5
3 5
4 5
1 3
输出1:
旧算法:
最大团中结点的个数为：4
结点编号为：1 2 3 5
形成的排列树结点数为：13
bbMaxClique函数耗时为：11ms
新算法:
最大团中结点的个数为：4
结点编号为：1 2 3 5
形成的排列树结点数为：13
bbMaxClique函数耗时为：11ms

输入2:
9 12
1 2
1 4
1 5
2 3
2 5
3 5
4 5
1 3
8 9
6 7
3 6
2 9
（注：此样例于上个样例相比，为了更好的测试新算法的上界函数，多增加了悬挂点部分）
输出2:
旧算法:
最大团中结点的个数为：4
结点编号为：1 2 3 5
形成的排列树结点数为：173
bbMaxClique函数耗时为：166ms
新算法:
```

```
最大团中结点的个数为：4
结点编号为：1 2 3 5
形成的排列树结点数为：128
bbMaxClique函数耗时为：106ms
```

算法分析

时间复杂度分析：

- 旧算法：while循环外层 $O(2^n)$，内侧由于要判断是否为合法结点，需要 $O(n)$，因此总时间复杂度为 $O(n2^n)$。
- 新算法：degree数组的维护需要 $O(N)$， $minDegree$ 的更新均为动态维护，时间复杂度为 $O(1)$，因此总时间复杂度仍然为 $O(n2^n)$

对比两个算法在样例二上的速度与结点数可以发现，更换了上界函数后，生成的结点数大大降低，效率更是提高了将近33%。