

SGU 122 解题手记

要求在一个图中找哈密顿回路。给出的条件接近一个著名的哈密顿回路存在的充分条件—— $n \geq 3$ 且每个顶点的度大于等于 $\lfloor \frac{N+1}{2} \rfloor$ ，只有 $n=2$ 时是个例外（ $n=2$ 的情况不会出现）。

其构造算法如下：

1. 随便寻找 1 条链（最少 1 个点）；
2. 不断寻找不在链上，但能加在链头或者链尾的点，将其加在链头或者链尾；
3. 若找不到这样的点：
 1. 说明链上至少有 $\lfloor \frac{N+3}{2} \rfloor$ 个点，且与链头或链尾相连的点都在链上，设此链为 $a_1 \dots a_m$ ，那么，必然存在一个 $i (1 \leq i \leq m-1)$ ，使得 a_i 与 a_m 相连， a_1 与 a_{i+1} 相连（抽屉原理）；
 2. 把链变成环： $a_1 \dots a_i a_m \dots a_{i+1} a_1$ ；
 3. 若环中已经包含所有的点，则算法结束，否则任意一个不在环中的点就会与环中的某个点相连，在相连处将环断开，把这个不在环中的点加上，形成一条链；
4. 回到 2。

编码过程中查到了一个 list 的方法 splice，可以整体移动一段 list，原型如下：

void splice(iterator position, list<T, Alloc>& x);	position must be a valid iterator in *this, and x must be a list that is distinct from *this. (That is, it is required that &x != this.) All of the elements of x are inserted before position and removed from x. All iterators remain valid, including iterators that point to elements of x. This function is constant time.
void splice(iterator position, list<T, Alloc>& x, iterator i);	position must be a valid iterator in *this, and i must be a dereferenceable iterator in x. Splice moves the element pointed to by i from x to *this, inserting it before position. All iterators remain valid, including iterators that point to elements of x. If position == i or position == ++i, this function is a null operation. This function is constant time.
void splice(iterator position, list<T, Alloc>& x, iterator f, iterator l);	position must be a valid iterator in *this, and [first, last) must be a valid range in x. position may not be an iterator in the range [first, last). Splice moves the elements in [first, last) from x to *this, inserting them before position. All iterators remain valid, including iterators that point to elements of x. This function is constant time.

Submit 1: TLE on 1。What?难道是因为常数大? 减小一点。

Submit 2: TLE on 1。时间没减少，不像常数大。经过不断的实验，包括写了一个用数组的 pas 版 (AC)，然后又翻译成 cpp(TLE)，现在基本确定是读入数据超时，可能 cpp 在读入的时候无法正确处理换行。换一种 ws 的处理方式——gets 然后 sscanf。

Submit 2.1: AC。194ms, pas 版 AC。

Submit 2.2: TLE on 20。424ms, cpp 版光荣 TLE.....读入数据的问题解决了，改去调试最初的 list 版。

Submit 3: WA on 3。改了一下与算法第 2 步相关的语句，使其更简洁。没找到 WA 的原因。

Submit 4: WA on 14。在处理算法第 3.3 步的时候，把断开环和加入点的顺序弄反了，把点加在了错误的位置，这也是 Submit 3 错误的原因。

Submit 5: TLE on 20。489ms，又一个 cpp 死活过不去的题诞生了……