初始，用户输入一个查询，系统返回这个查询上能建立的查询推荐。用户能够选择一个推荐，或者自己在现有查询的基础上创建查询，或者建立新的根查询节点。之后重复这个“推荐——选择”的过程。

**推荐**

推荐的过程是一个蒙特卡洛搜索的过程。

首先根据现有的搜索树初始化新的搜索树。（其实就是将每个节点的记录衰减一下。）

之后开始搜索，即完成K次模拟。

每次模拟要经过5个阶段：

* 在当前用户建立的节点中选出一个作为模拟的根节点
* 从这个模拟根节点开始向下选出一个子节点未被完全扩展的节点（选择策略是UCT，和传统MCTS的selection差不多）
* 在其未被扩展的节点中随机选一个扩展（和传统MCTS的expansion差不多）。所谓的扩展是指将创建这个节点并加入到搜索树中，之后开始记录这个节点上的模拟信息。创建节点可以用constructNewNodefromChild这个函数。
* 从这个新扩展的节点开始，不断向下模拟，每次在能建立的子节点中随机选一个，直至达到深度限制depthL。
* 之后反传，反传有两个阶段。第一个阶段，从模拟的尽头节点反传到新扩展的节点：调用一个评估函数，计算当前节点的收益prof，加上“γ\*下一个节点反传回的prof”后，反传给上一个节点。中间不记录信息。第二个阶段，从新扩展的节点反传到总的根节点：从新扩展的节点开始每个节点都要记录模拟信息（修改queryNode里面的属性）。按照公式来计算，prof(Q)调评估函数（这个值每次都一样，可以一次算好存起来，加快速度），后面的V(sub)要读取各个子节点的acc\_V/sim\_num，之后求平均。计算完成后得到V(Q)，让acc\_V += V(Q)，sim\_num += 1。一直反传到总的根节点，注意是总的根节点，不是新选出的根节点。

K次模拟完成后，开始对用户能建立的下一级查询节点排序。所谓下一级节点是指所有用户已经建立的查询节点的所有下一级子节点。计算所有这些下一级子节点的acc\_V/sim\_num，按照这个值排序，返回给用户，就完成了一次推荐。

**选择**

用户的选择有三种可能：

* 选一个推荐的节点：直接在搜索树中将这个节点的“被选中flag”置1；
* 从现有查询树上，自己选根节点、选条件、选数据源，自己建立新查询：按道理说，所有能创建的新查询都一定在我们的搜索树中，所以出现这种情况有两种可能，①我们只把评价高的查询推荐给了用户，用户想创建的节点并不在里面；②用户没看见他想要的节点。无论怎样，这个查询对应的节点一定在搜索树里面，用户选定它后，我们要根据其条件在搜索树中找到这个节点，并将其flag置1；
* 不依赖现有节点，直接输入查询：这种情况对应创建新的根节点，也是系统刚开始运行时的情况。除了创建新查询节点之外，还要将其加到根节点列表中。

以上三种情况可以分别用confirmNode，constructNewNodefromQuery，constructNewNodefromCondition这三个函数来实现。

* 评估函数可以先用查到数据的数量来代替。
* 需要给出一个接口，支持外部用户做出上述的三种选择。