《计算机算法设计与分析》

第七章 分枝—限界法

马丙鹏

2020年11月10日

中国科学院大学
University of Chinese Academy of Sciences

第七章 分枝－限界法

- 7.1 一般方法
- 7.2 LC-检索
- 7.3 15-谜问题
- 7.4 LC-检索(续)
- 7.5 分枝-限界算法
- 7.6 0/1背包问题
- 7.7 货郎担问题

7.1 一般方法

- 基本概念
  - 问题解的n元组表示
  - 问题状态
  - 解状态
  - 答案状态
  - 活结点
  - E-结点
  - 死结点

7.1 一般方法

■ 基本概念

□ 回溯法：

- ➢ 使用限界函数的深度优先结点生成方法称为回溯法(backtracking)

□ 分枝-限界方法：

- ➢ E结点一直保持到死为止的状态生成方法称为分枝-限界方法(branch-and-bound)

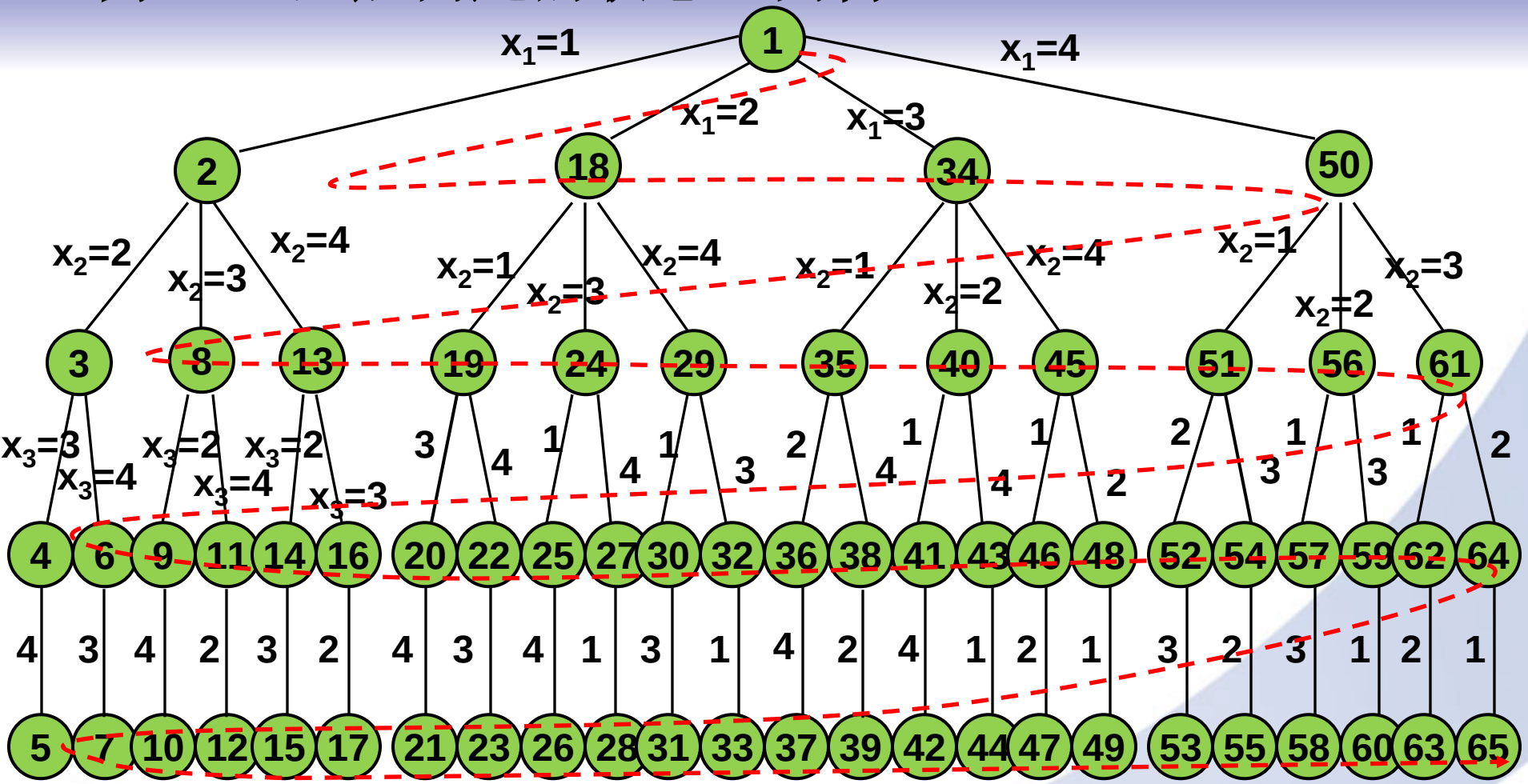
□ 限界函数：

- ➢ 在结点的生成过程中，需要用限界函数杀死还没有全部生成儿子结点的一些活结点——这些活结点无法满足限界函数的条件，不可能导致问题的答案。

7.1 一般方法

- 分枝-限界法：
  - 在生成当前E-结点全部儿子之后再生成其它活结点的儿子，且用限界函数帮助避免生成不包含答案结点子树的状态空间的检索方法。
  - 活结点表：
    - 活结点：已经被生成，但还没有被检测的结点。
    - 存储结构：队列(First In First Out, BFS)、栈(Last In First Out, D-Search)
  - 分枝-限界法的两种设计策略：
    - **FIFO检索**：活结点表采用队列
    - **LIFO检索**：活结点表采用栈

例7.1 4-皇后问题的状态空间树

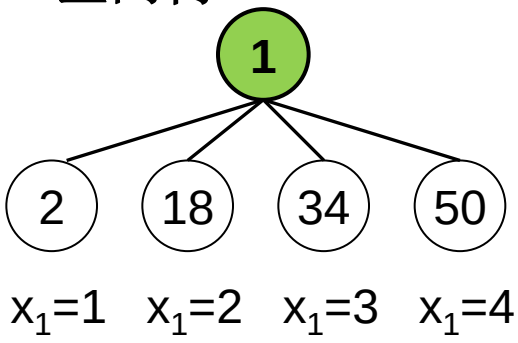


限界函数：如果$(x_1, x_2, \ldots, x_i)$是到当前E结点的路径，那么具有父一子标记x_{i+1}的所有儿子结点是一些这样的结点，它们使得$(x_1, x_2, \ldots, x_i, x_{i+1})$表示没有两个皇后正在相互攻击的一种棋盘格局。

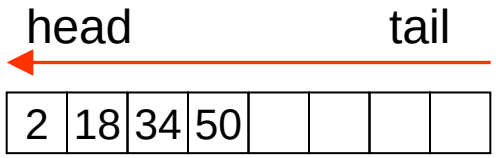
FIFO分枝－限界法检索4-皇后问题

活结点

扩展活结点得到的状态空间树



活结点表(队列)

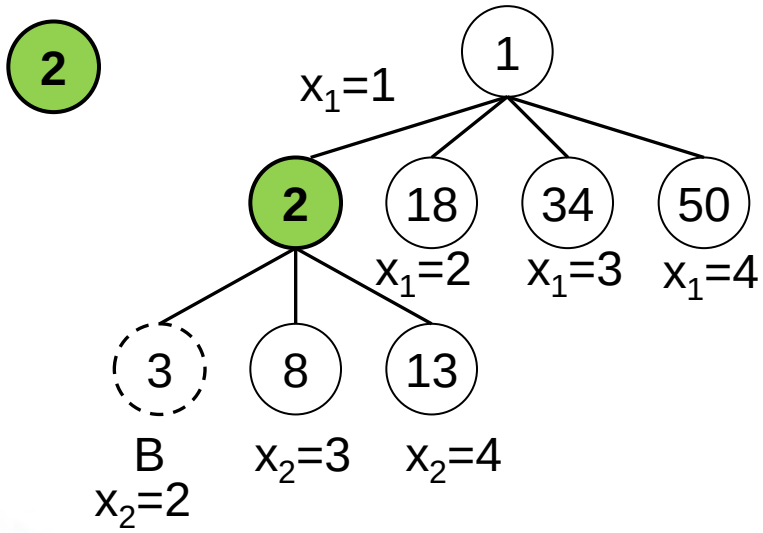


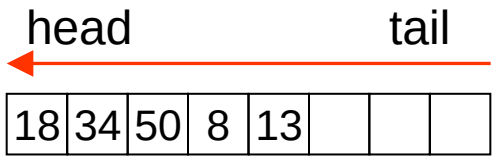
扩展结点1，得新结点2，18，34，50

活结点2、18、34、50入队列

FIFO分枝－限界法检索4-皇后问题

活结点　　扩展活结点得到的状态空间树　　活结点表(队列)





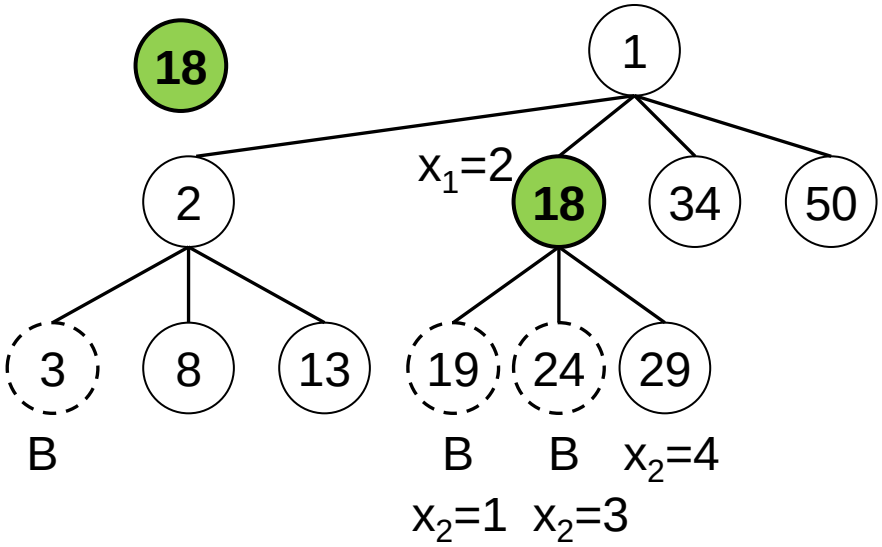
扩展结点2，得新结点3，8，13

利用限界函数杀死结点3

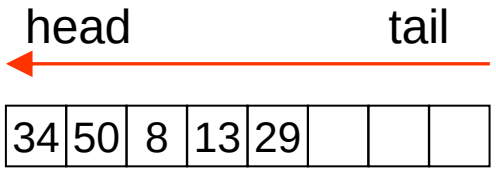
活结点8、13入队列

FIFO分枝－限界法检索4-皇后问题

活结点　　扩展活结点得到的状态空间树



活结点表(队列)



扩展结点18，得新结点19，24，29

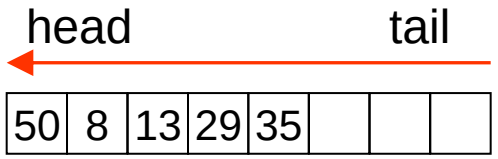
利用限界函数杀死结点19、24

活结点29入队列

FIFO分枝－限界法检索4-皇后问题

活结点　　扩展活结点得到的状态空间树

活结点表(队列)

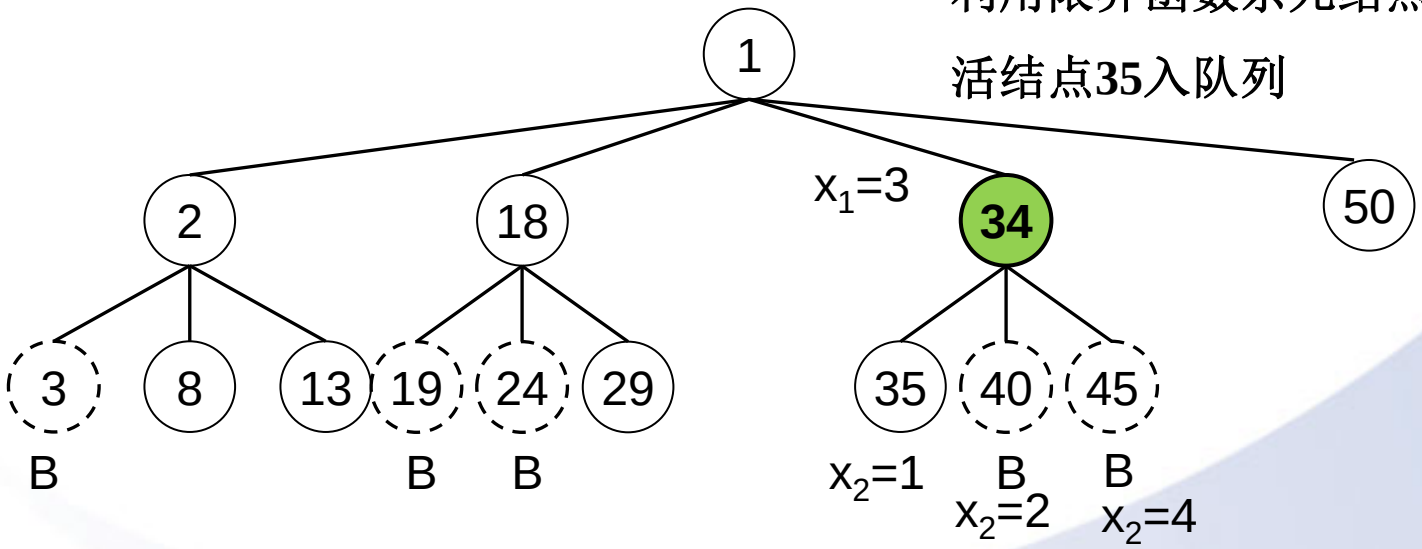


34

扩展结点34，得新结点35，40，45

利用限界函数杀死结点40、45

活结点35入队列

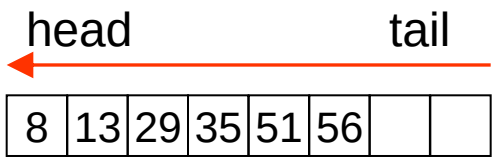


FIFO分枝－限界法检索4-皇后问题

活结点

扩展活结点得到的状态空间树

活结点表(队列)

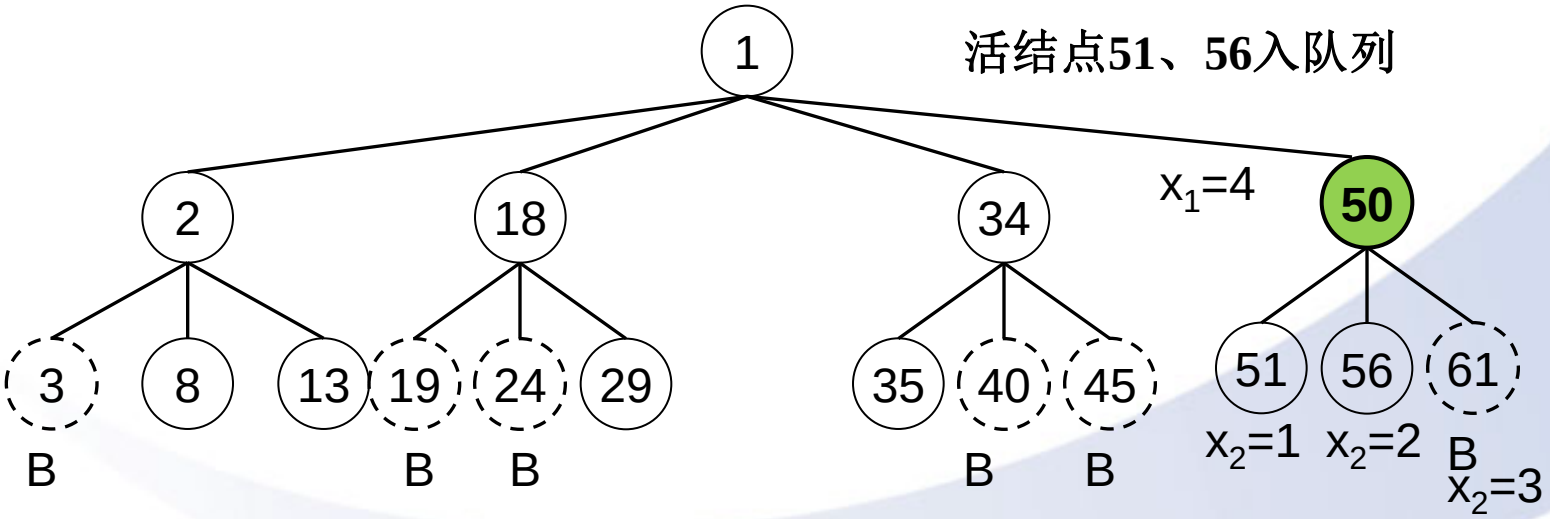


50

扩展结点50，得新结点51，56，61

利用限界函数杀死结点61

活结点51、56入队列

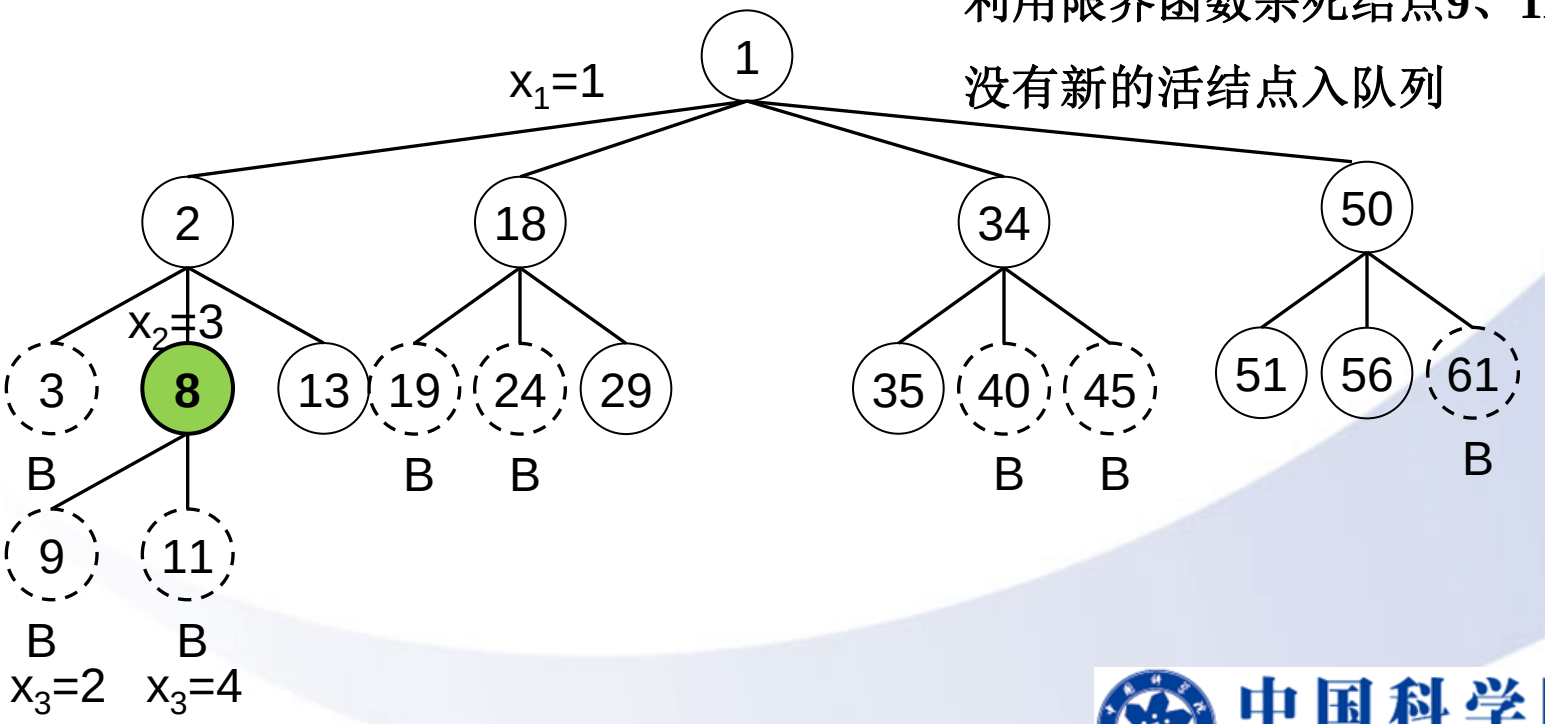


FIFO分枝－限界法检索4-皇后问题

活结点

8

扩展活结点得到的状态空间树



活结点表(队列)

head tail

13	29	35	51	56			

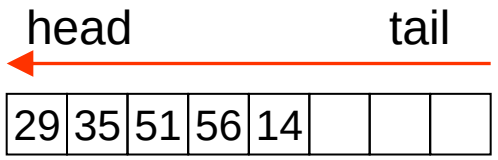
扩展结点8，得新结点9，11

利用限界函数杀死结点9、11

没有新的活结点入队列

FIFO分枝－限界法检索4-皇后问题

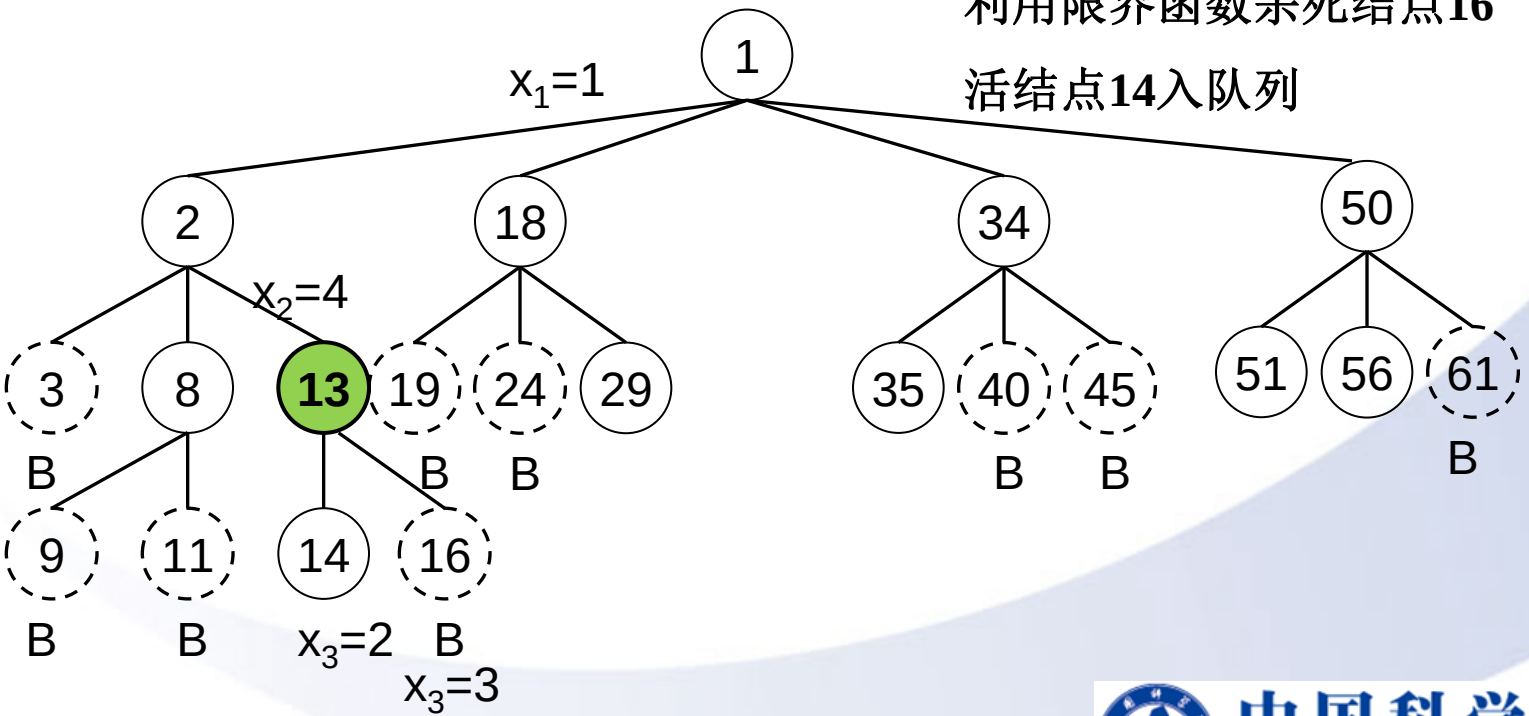
活结点　　扩展活结点得到的状态空间树　　活结点表(队列)



扩展结点13，得新结点14，16

利用限界函数杀死结点16

活结点14入队列



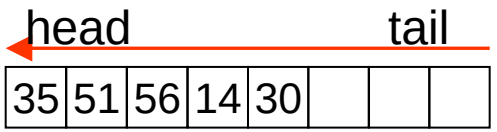
FIFO分枝－限界法检索4-皇后问题

活结点

29

扩展活结点得到的状态空间树

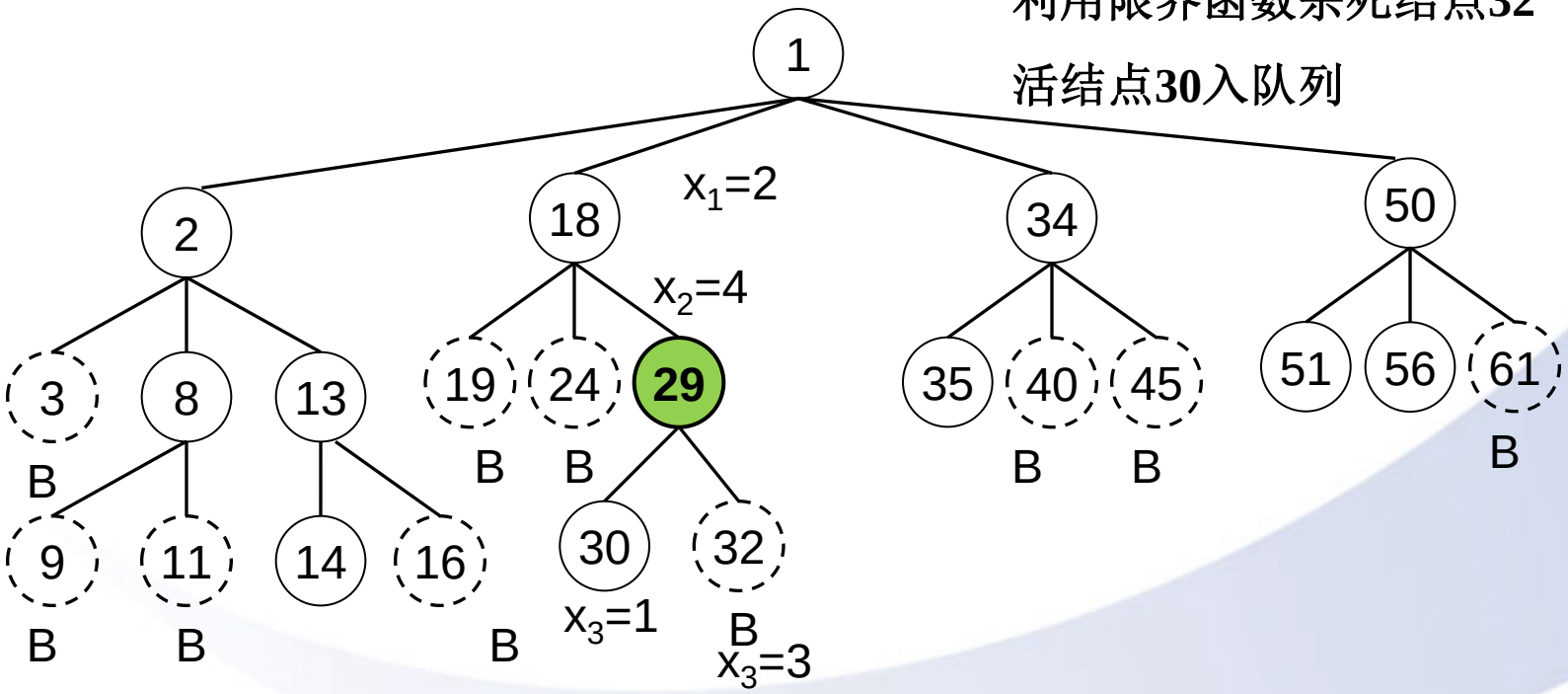
活结点表(队列)



扩展结点29，得新结点30，32

利用限界函数杀死结点32

活结点30入队列



FIFO分枝－限界法检索4-皇后问题

活结点

35

扩展活结点得到的状态空间树

活结点表(队列)

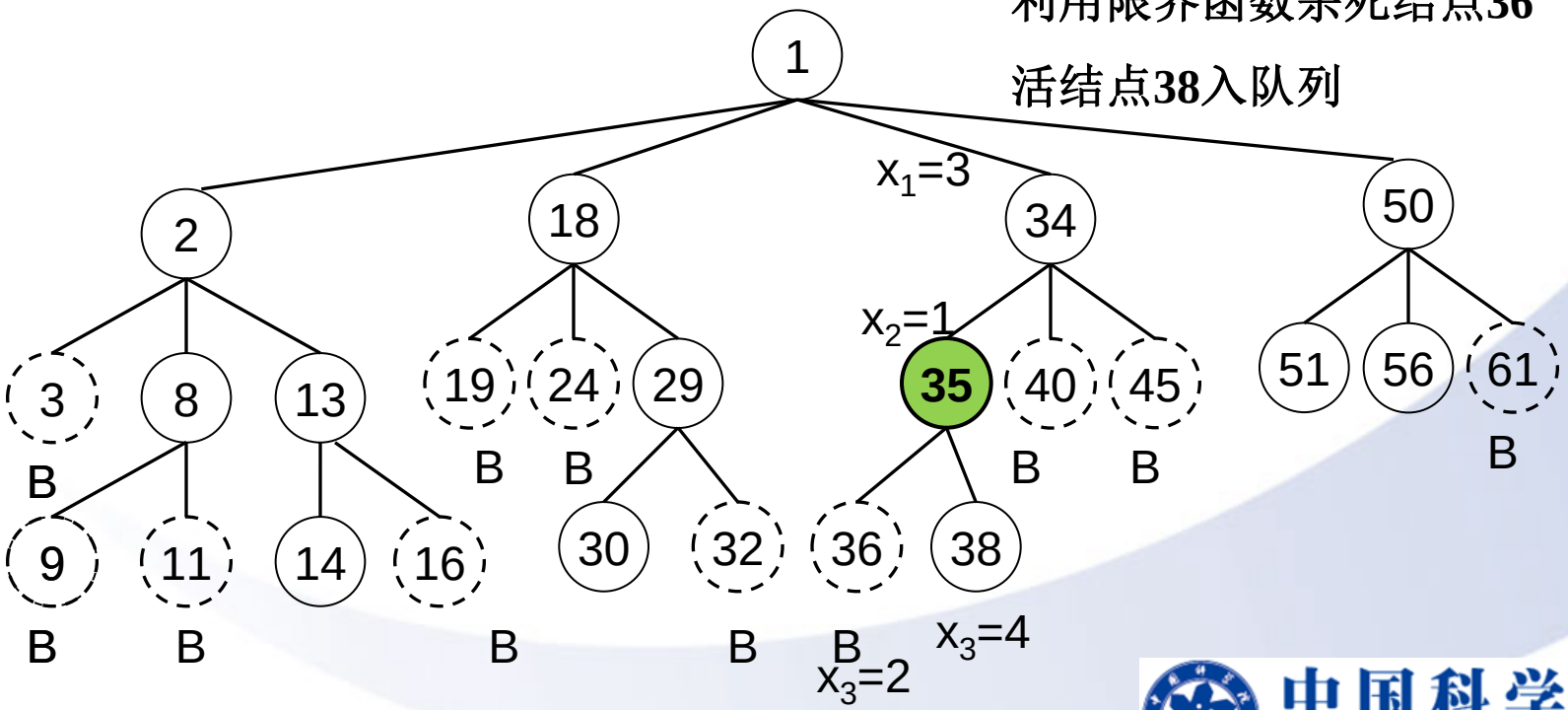
head tail

51	56	14	30	38			

扩展结点35，得新结点36，38

利用限界函数杀死结点36

活结点38入队列



FIFO分枝－限界法检索4-皇后问题

活结点　　扩展活结点得到的状态空间树　　活结点表(队列)

51

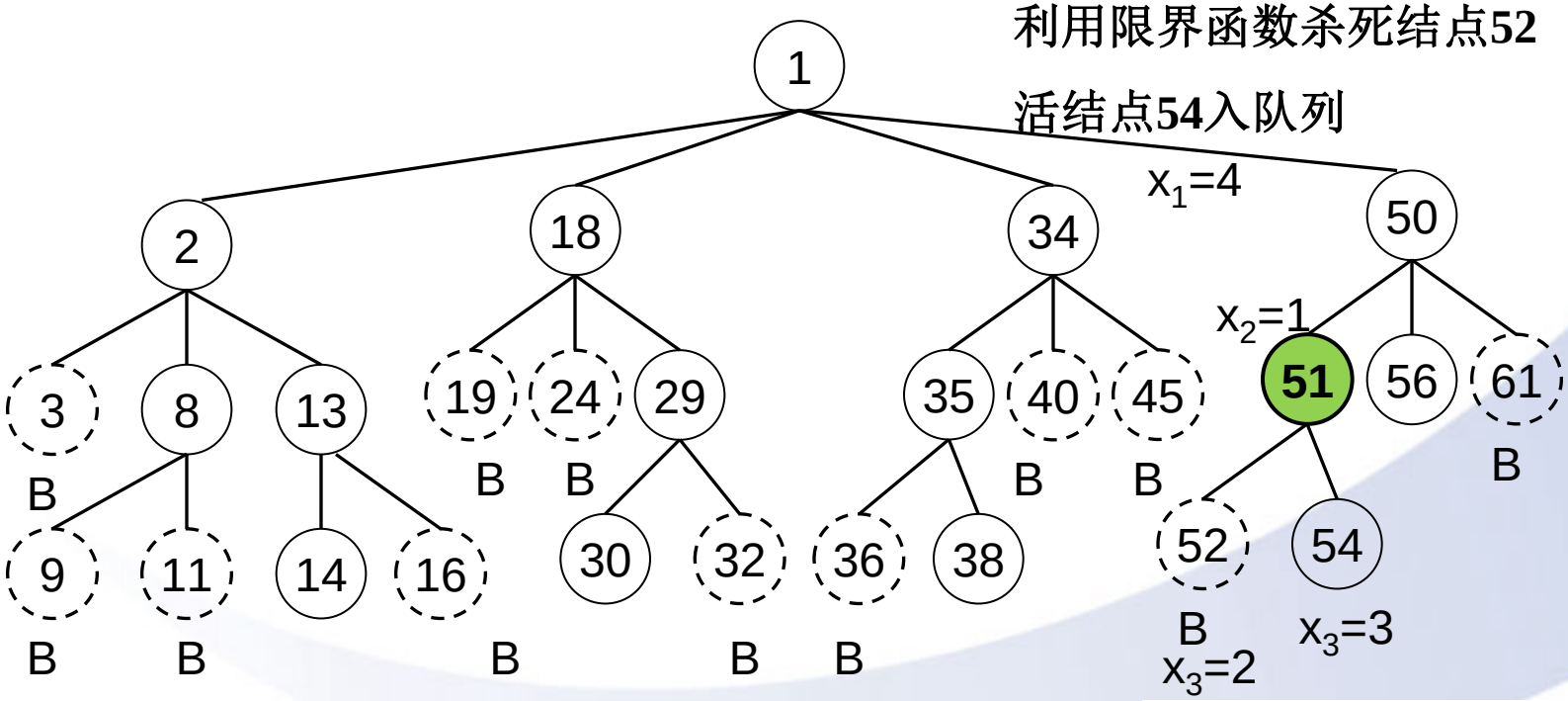
head　　tail

56	14	30	38	54			

扩展结点51，得新结点52，54

利用限界函数杀死结点52

活结点54入队列

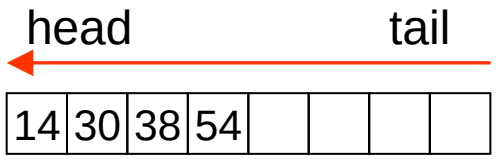


FIFO分枝－限界法检索4-皇后问题

活结点

扩展活结点得到的状态空间树

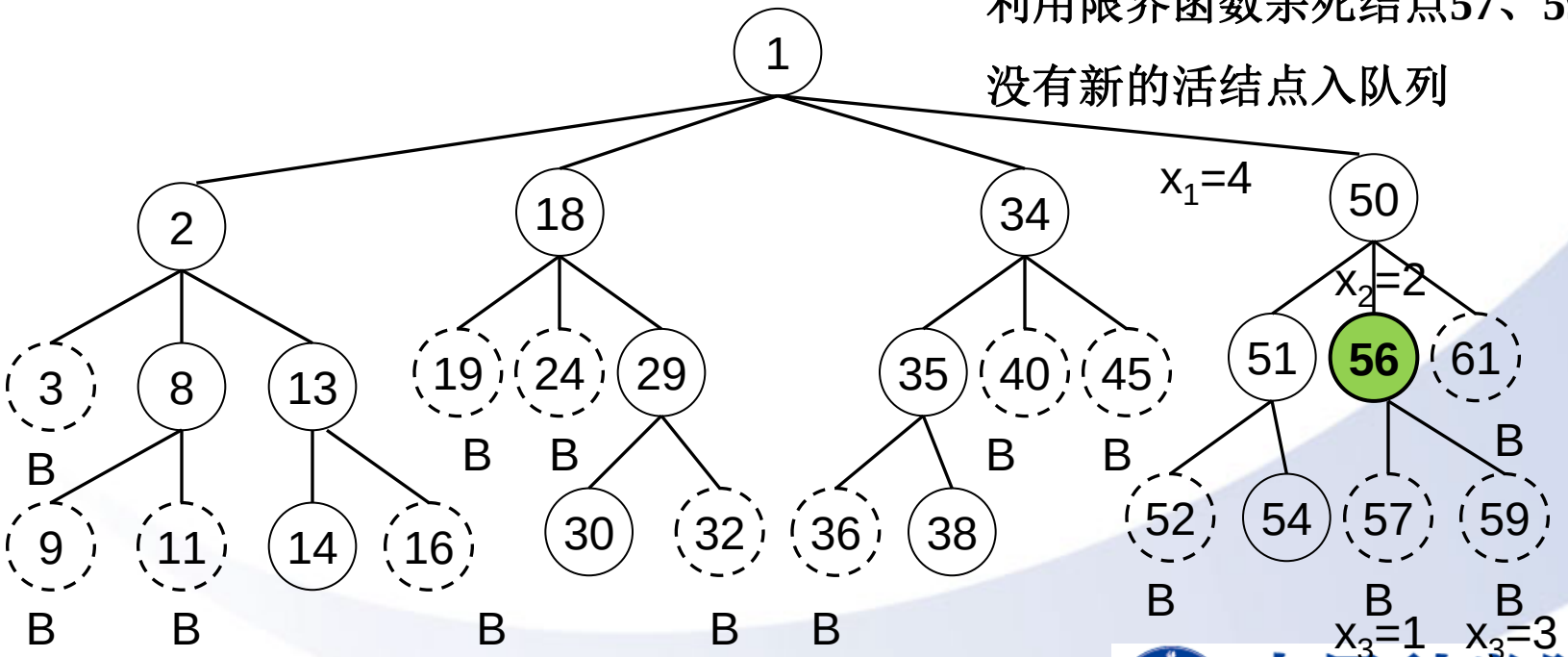
活结点表(队列)



扩展结点56，得新结点57，59

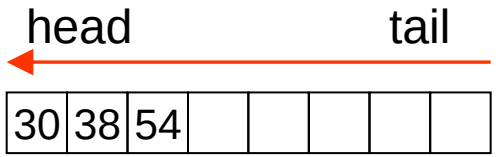
利用限界函数杀死结点57、59

没有新的活结点入队列

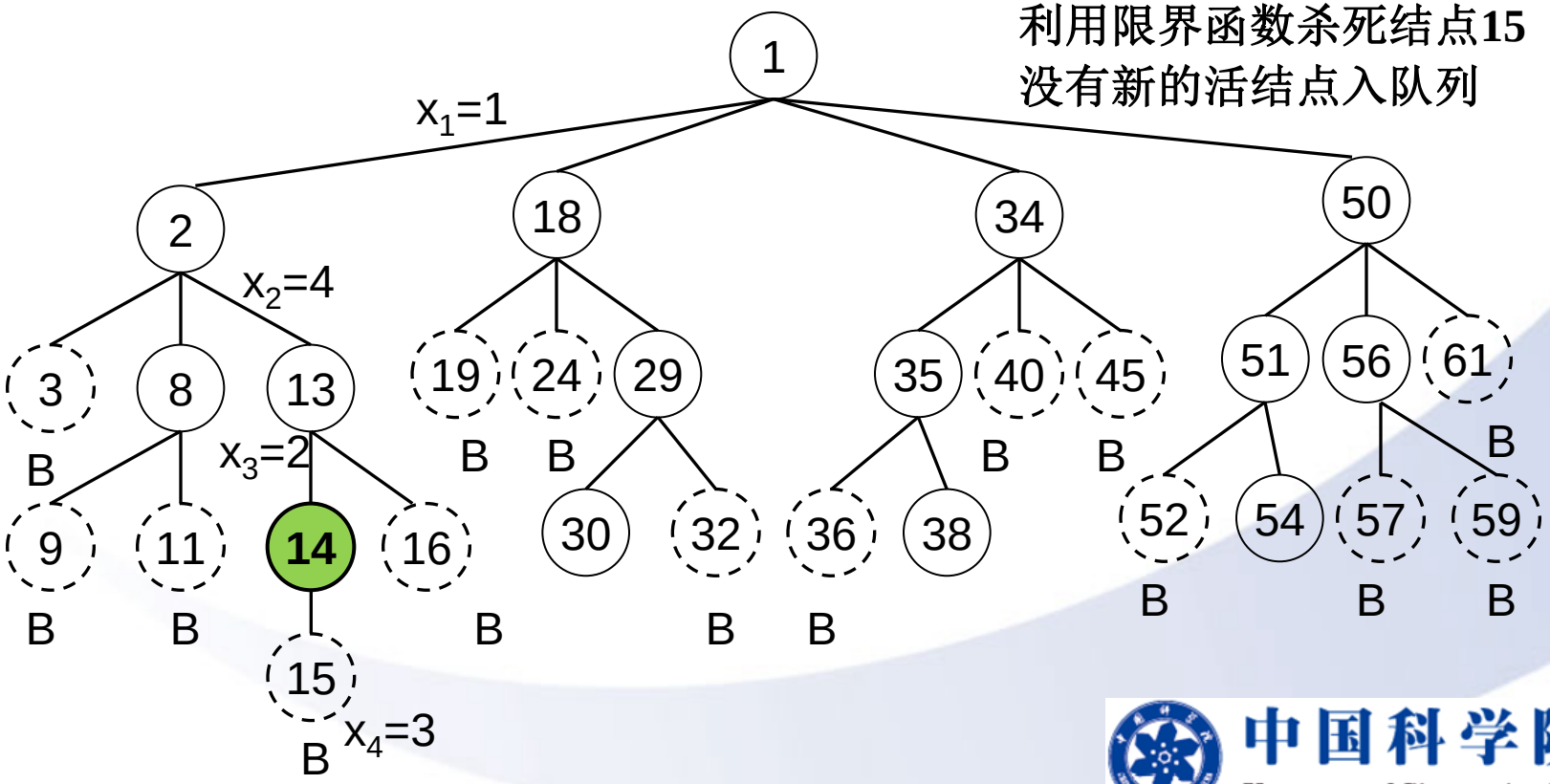


FIFO分枝－限界法检索4-皇后问题

活结点　　扩展活结点得到的状态空间树　　活结点表(队列)



扩展结点14，得新结点15
利用限界函数杀死结点15
没有新的活结点入队列

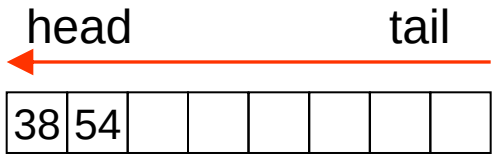


FIFO分枝－限界法检索4-皇后问题

活结点

扩展活结点得到的状态空间树

活结点表(队列)



扩展结点30，得新结点31

结点31为答案结点

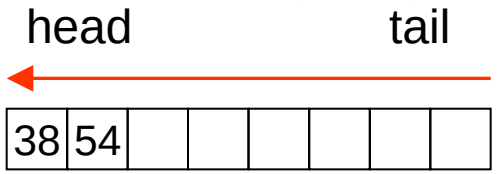
算法终止(找到一个答案结点)

FIFO分枝－限界法检索4-皇后问题

活结点

扩展活结点得到的状态空间树

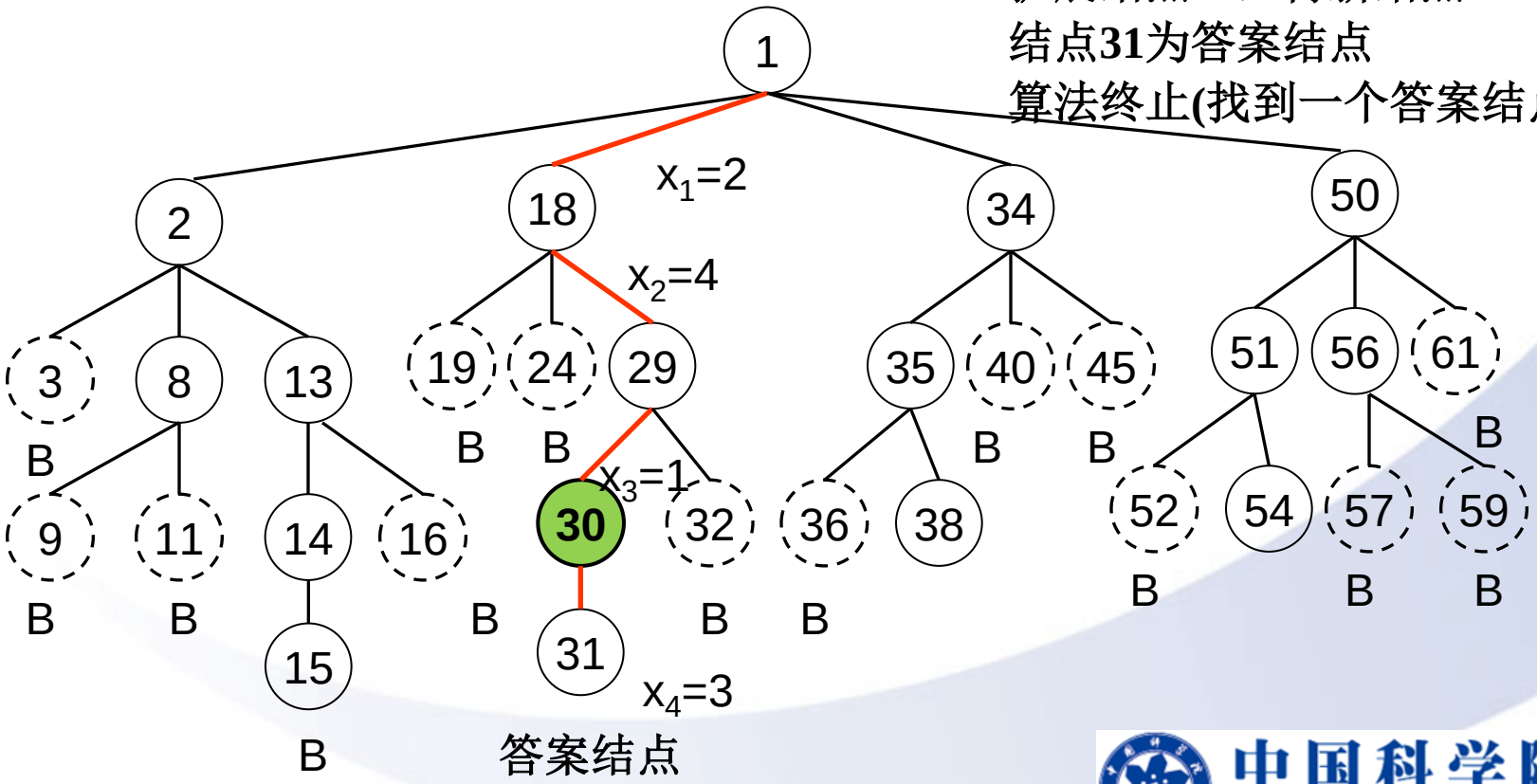
活结点表(队列)



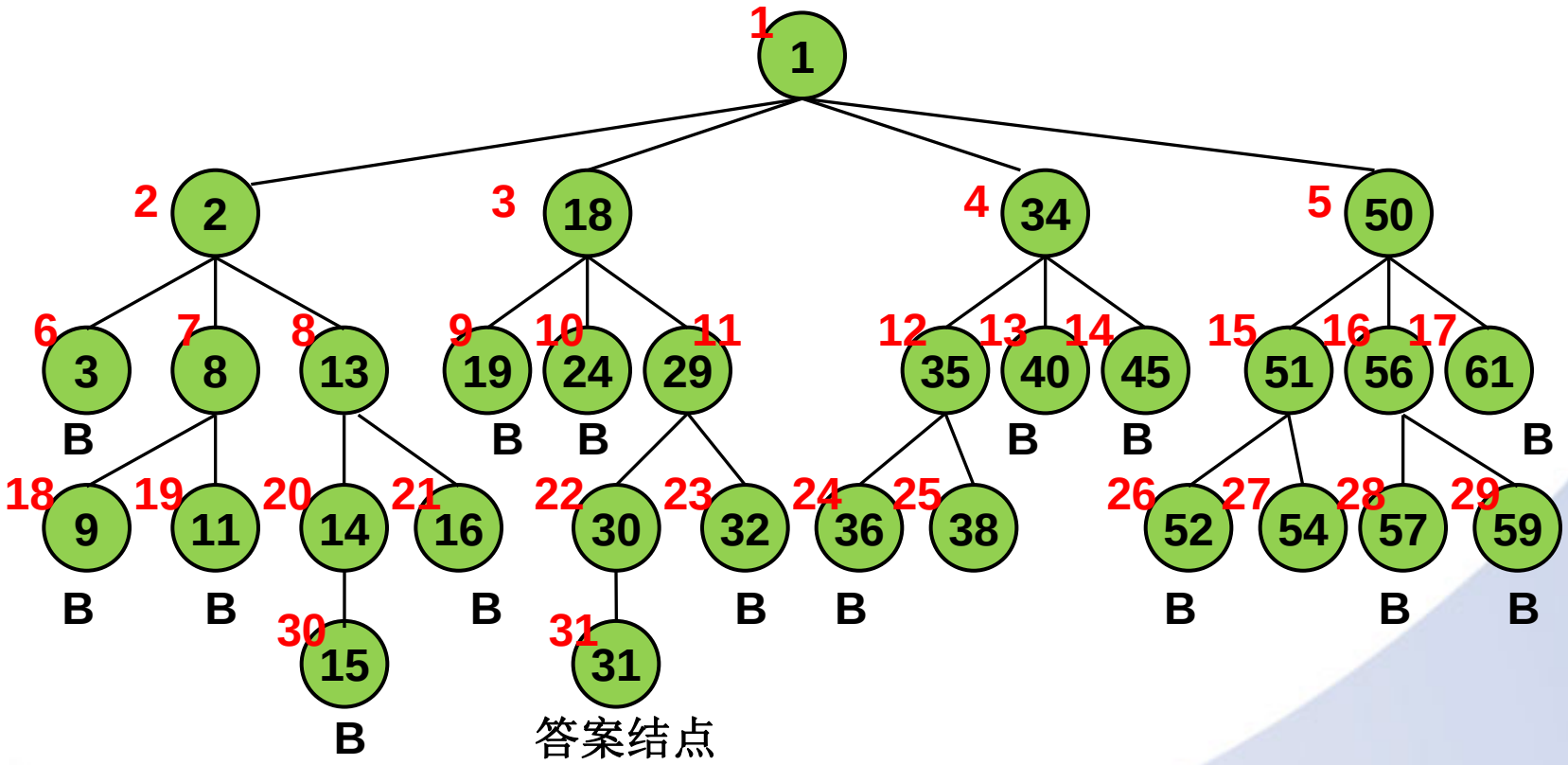
扩展结点30，得新结点31

结点31为答案结点

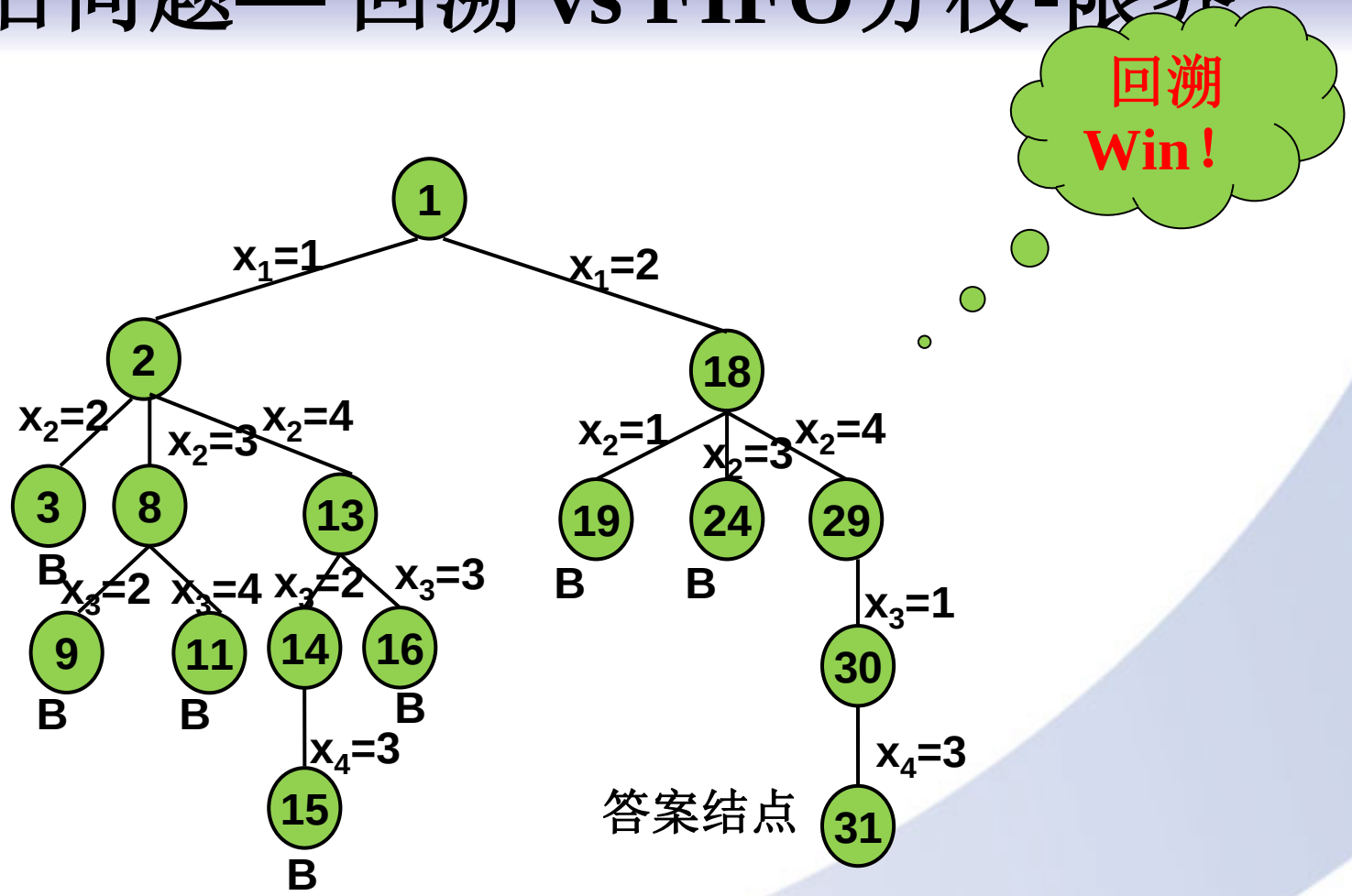
算法终止(找到一个答案结点)



FIFO分枝一限界法检索4-皇后问题



4-皇后问题— 回溯 vs FIFO分枝-限界

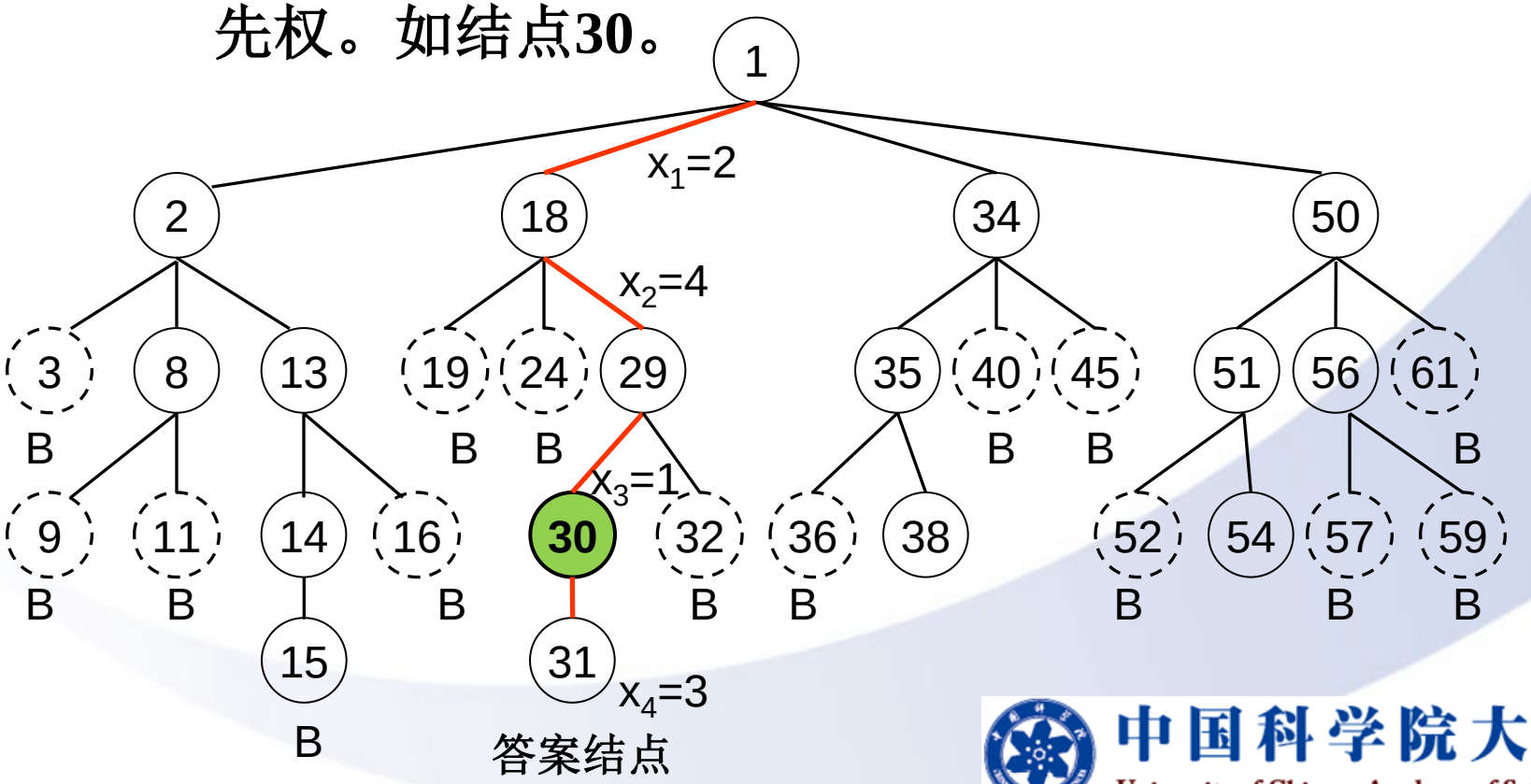


第七章 分枝－限界法

- 7.1 一般方法
- 7.2 LC-检索
- 7.3 15-谜问题
- 7.4 LC-检索(续)
- 7.5 分枝-限界算法
- 7.6 0/1背包问题
- 7.7 货郎担问题

7.2 LC-检索(Least Cost)

- **LIFO和FIFO分枝-限界法存在的问题**
  - 对下一个E-结点的选择规则过于死板、盲目。对于有可能快速检索到一个答案结点的结点没有给出任何优先权。如结点30。



7.2 LC-检索(Least Cost)

- LIFO和FIFO分枝-限界法存在的问题
  - 如何解决？
    - 做某种排序，让可以导致答案结点的活结点排在前面！
  - 新问题：怎么排序？
    - 寻找一种“有智力”的排序函数C( )，用C( )来选取下一个E结点，加快到达一答案结点的检索速度。
    - 如结点30，29→30→31

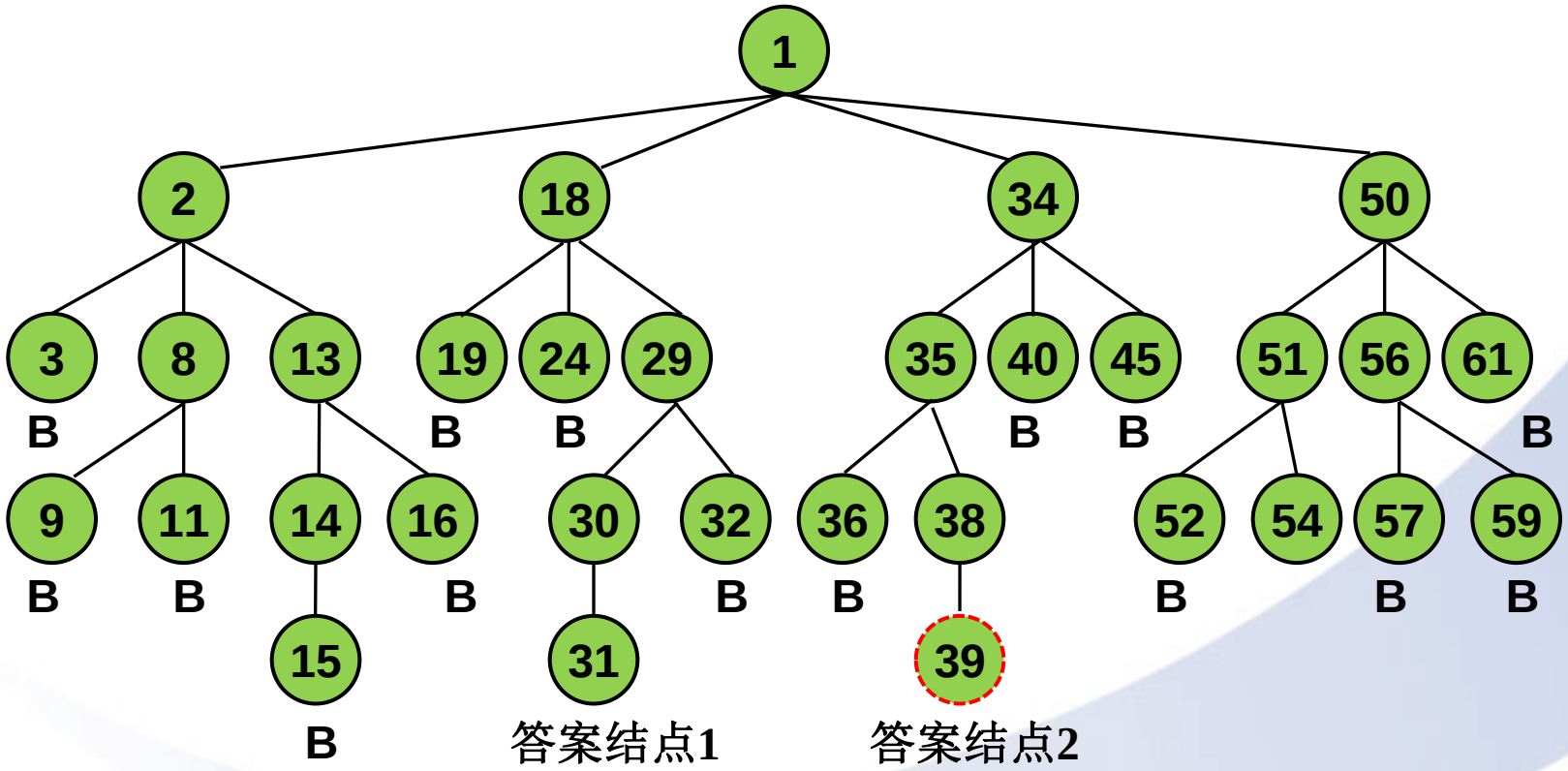
能否赋予一个比其它活结点高的优先级使其尽快成为E结点？

7.2 LC-检索(Least Cost)

- 如何衡量结点的优先等级？
  - 对于任一结点，用该结点导致答案结点的成本(代价)来衡量该结点的优先级——成本越小越优先。
  - 对任一结点X，可以用两种标准来衡量结点的代价：
    ① 在生成一个答案结点之前，子树X需要生成的结点数。
    偏向于选择生成儿子结点数目最小的结点作为E结点。
    ② 在子树X中离X最近的那个答案结点到X的路径长度。
    偏向于选择由根到最近的那个答案结点路径上的结点作为E结点进行扩展。

7.2 LC-检索(Least Cost)

■ 如何衡量结点的优先等级？



7.2 LC-检索(Least Cost)

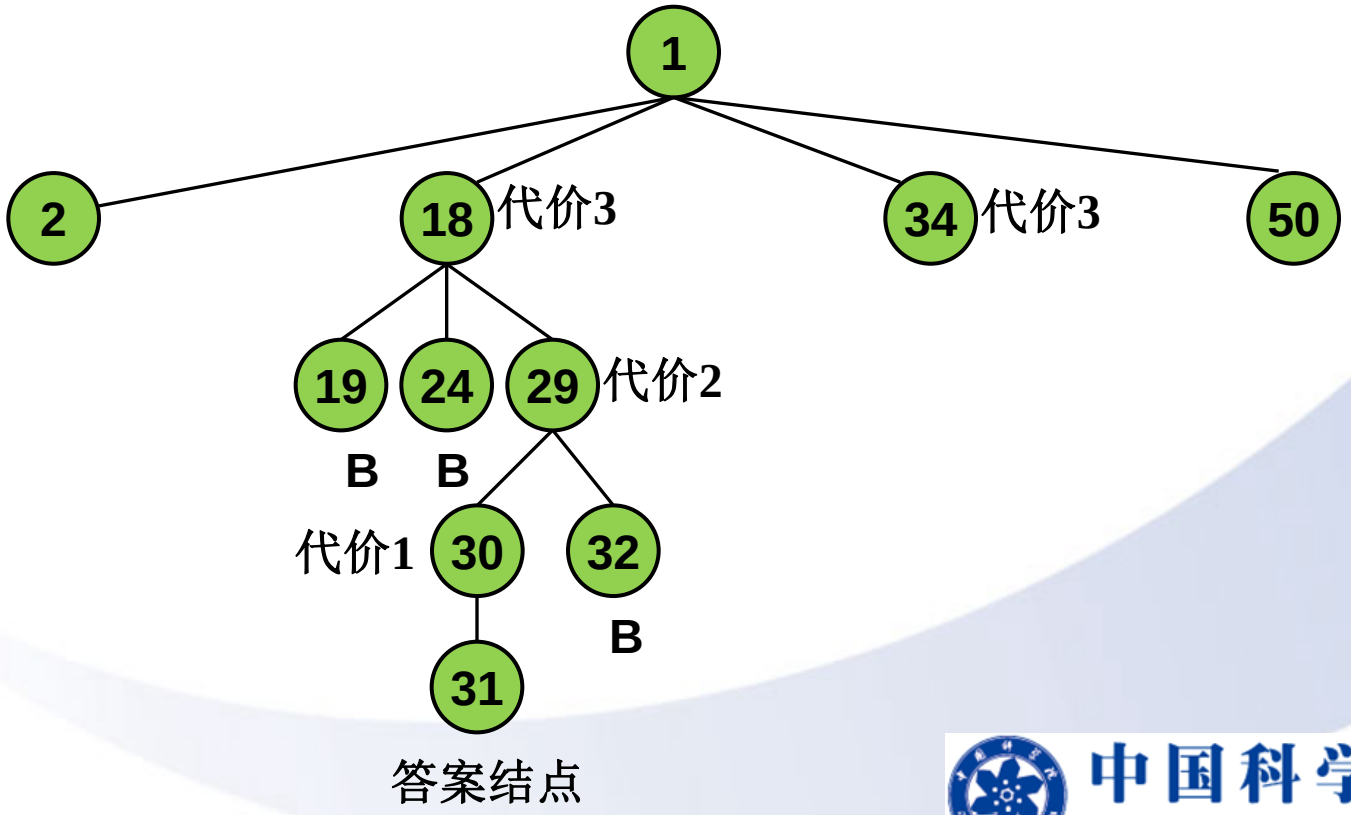
- 如何衡量结点的优先等级？
  - 例：在量度(2)下各结点的代价：

结点	代价
1	4
18, 34	3
29, 35	2
30, 38	1

  - 其余所有在2, 3, 4级上的结点的代价应该分别大于(除31、39) 3, 2, 1

7.2 LC-检索(Least Cost)

- 如何衡量结点的优先等级？
  - 如果以度量(2)的这些代价作为选择下一个E-结点的依据，则结点生成过程如下：

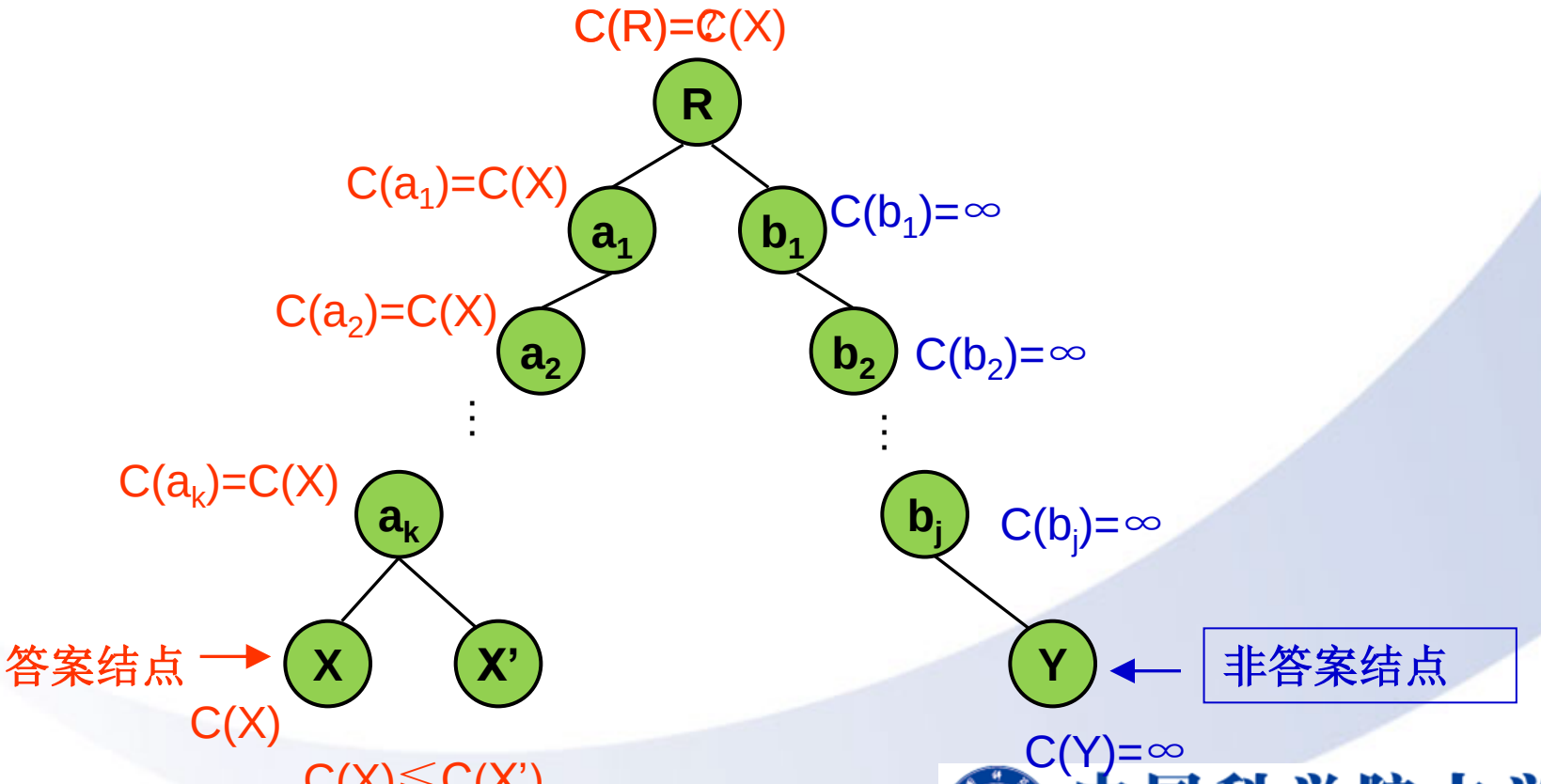


7.2 LC-检索(Least Cost)

- 结点成本函数
  - C( )：“有智力”的排序函数，依据成本排序，优先选择成本最小的活结点作为下一个扩展的E结点。C( )又称为“结点成本函数”。
  - 结点成本函数C(X)的定义：
    ① 如果X是答案结点，则C(X)是由状态空间树的根结点到X的成本(即所用的代价，可以是级数、计算复杂度等)。
    ② 如果X不是答案结点且子树X不包含任何答案结点，则C(X)=∞。
    ③ 如果X不是答案结点但子树X包含答案结点，则C(X)等于子树X中具有最小成本的答案结点的成本。

7.2 LC-检索(Least Cost)

- 结点成本函数
  - 结点成本函数的计算

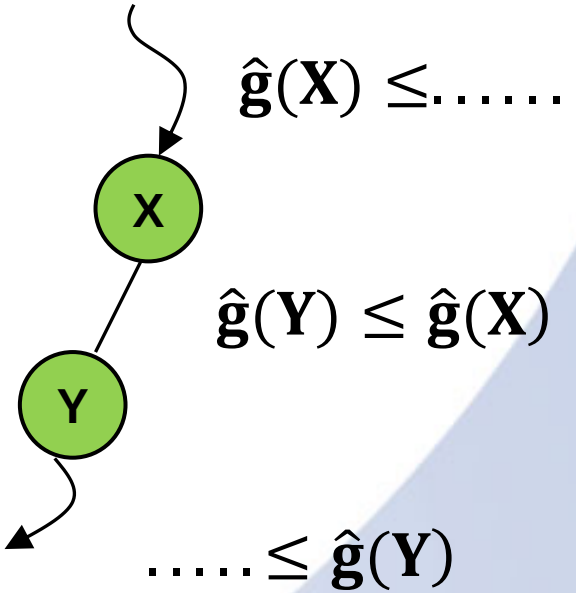


7.2 LC-检索(Least Cost)

- 结点成本函数
  - 计算结点成本函数的困难
    - 计算一个结点的代价通常要检索包含一个答案结点的子树X才能确定，因此计算C(X)的工作量和复杂度与解原始问题是相同的。
    - 计算结点成本的精确值是不现实的——相当于求解原始问题。
    - 结点成本的估计函数，包括两部分： $\hat{g}(X)$ 和 $h(X)$。
      $\hat{g}(X)$：设 $\hat{g}(X)$ 是由X到达一个答案结点所需成本的估计函数。
      性质：单纯使用$\hat{g}(X)$选择E结点会导致算法偏向纵深检查。

7.2 LC-检索(Least Cost)

- 结点成本函数
  - 计算结点成本函数的困难
    - 假设结点X是当前的E-结点，它的儿子为Y，由于通常要求 $\hat{g}(Y) \le \hat{g}(X)$，因此，活结点表中其它结点的成本估计值均大于 $\hat{g}(Y)$，于是Y将在X之后变成E-结点；
    - 同理，然后Y的儿子中有一个变成E-结点；接着Y的一个孙子变成E-结点等等，直到子树X全部检索完毕才可能生成那些除X子树以外的子树结点。



$\hat{g}(\bullet)$是X到答案结点的最小成本

7.2 LC-检索(Least Cost)

- 结点成本函数
  - 计算结点成本函数的困难
    - 纵深检索：直到子树X全部检索完才可能生成那些除了X子树以外的子树结点。
      ① 如果 $\hat{g}(X) = C(X)$，最理想！
      ② 否则，可能导致不能很快地找到更靠近根的答案结点。
      特例：对于结点W和Z完全可能有这样一种情况，Z比W更接近答案结点但是$\hat{g}(Z) > \hat{g}(W)$。此时若使用$\hat{g}()$给结点排序，必然导致对W子树作纵深检查，结果显然是不理想的。

7.2 LC-检索(Least Cost)

- 结点成本函数
  - 计算结点成本函数的困难
    - 如何避免单纯考虑$\hat{g}(X)$造成的纵深检查？
      - ✓引进h(X)改进成本估计函数。
      - ✓h(X)：根结点到结点X的成本。
    - 改进的结点成本估计函数

$$\hat{c}(X) = f(h(X)) + \hat{g}(X)$$

      - ✓f( )是一个非降函数。
      - ✓非零的f( )可以减少算法作偏向于纵深检查的可能性，它强使算法优先检索更靠近答案结点但又离根较近的结点。

7.2 LC-检索(Least Cost)

- 结点成本函数
  - LC-检索：
    - 选择$\hat{c}(\bullet)$值最小的活结点作为下一个E-结点。
  - BFS和D-Search都是LC-检索的特例：
    - BFS：依据级数来生成结点，令
      $\hat{g}(X) = 0$； f (h(X)) = X的级数
    - D-Search：令f (h(X)) = 0；而当Y是X的一个儿子时，总有$\hat{g}(X) \geq \hat{g}(Y)$。
  - LC分枝-限界检索
    - 伴之有限界函数的 LC-检索。

End