高级搜索树

B-树：查找

10-B3

邓俊辉
deng@tsinghua.edu.cn

高至天低至深海

每寸搜索着这天下

寻觅着那个"它"

...按照模型的运算量，用现有的最高计算能力模拟百分之一秒的聚变过程，就需大约二十年时间。而研究过程中的模拟需要反复进行，这使得模型的实际应用成为不可能。

算法

- 将根节点作为当前节点 //常驻RAM

  只要当前节点非外部节点

  在当前节点中顺序查找 //RAM内部

  若找到目标关键码，则

  返回查找成功

  否则 //止于某一对下层引用

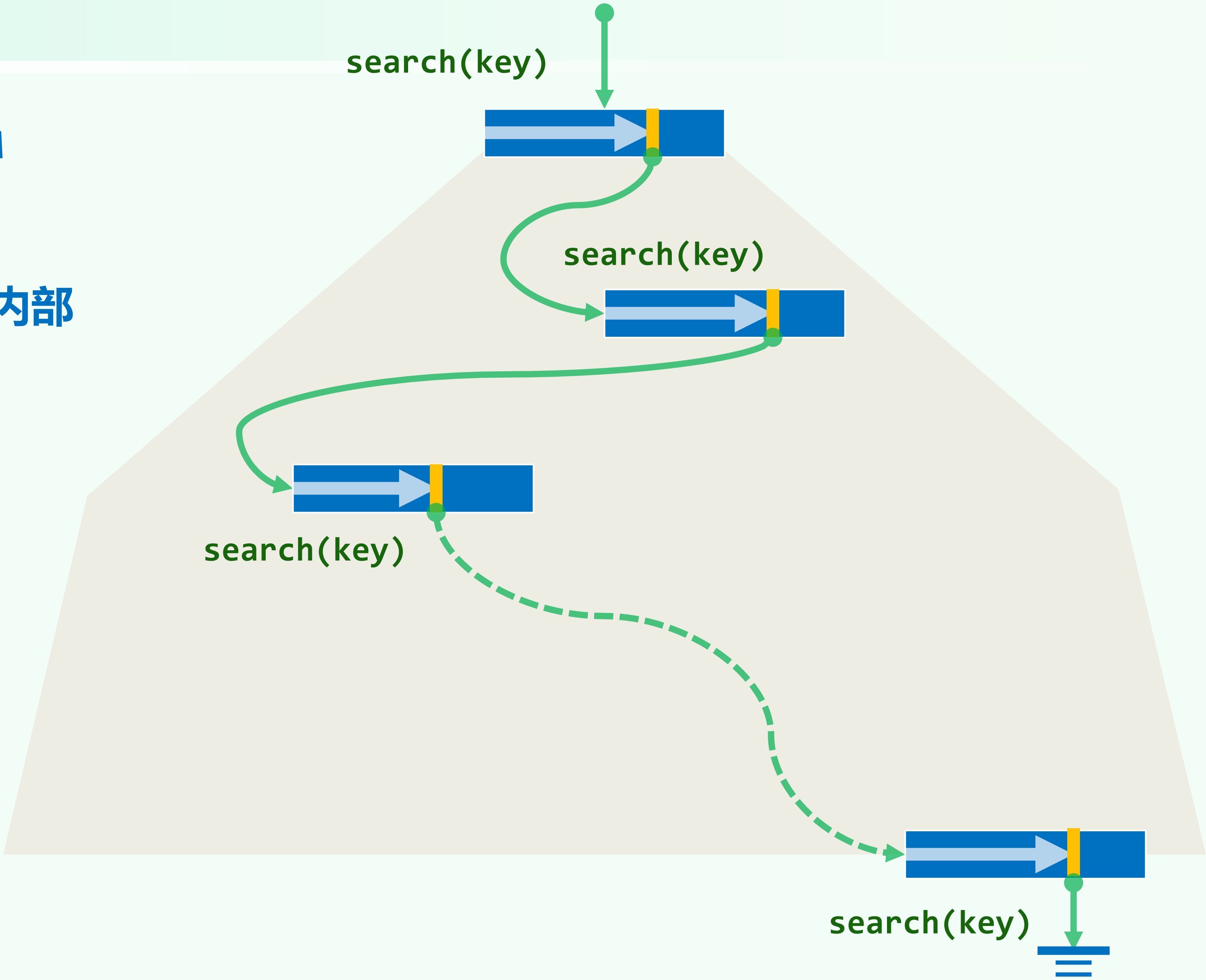
  沿引用，转至对应子树

  将其根节点读入内存

  //I/O，最为耗时

  更新当前节点

  返回查找失败



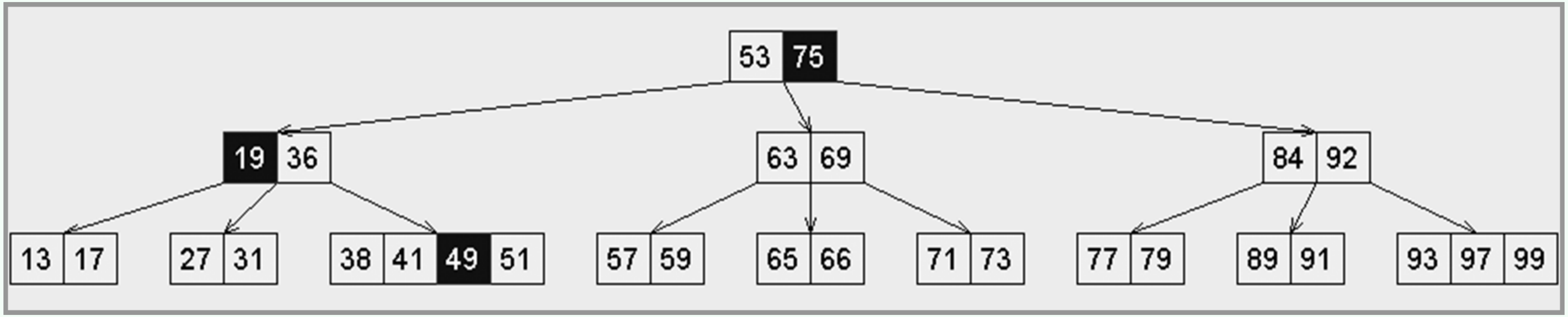
实例

❖ (3,5)-树：　53 97 36 89 41 75 19 84 77 79 51 57 99 91

　　92 93 17 73 13 66 59 49 63 65 71 69 27 31 38

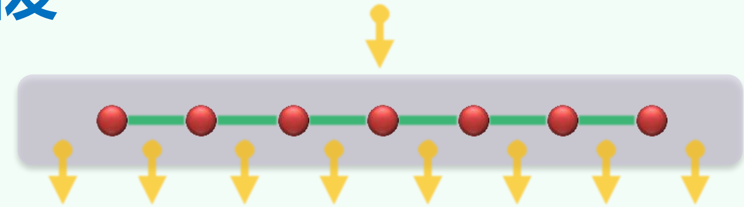
成功查找：　75, 19, 49

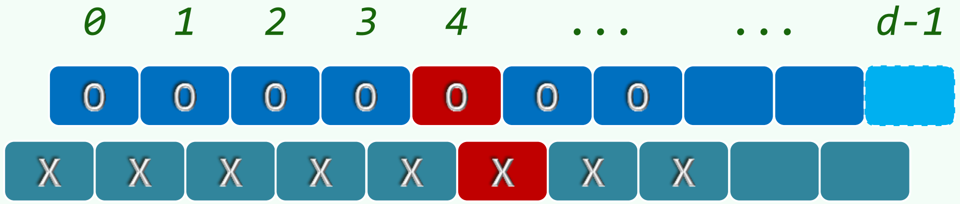
失败查找：　5, 45



实现

```
❖ template <typename T> BTNodePosi(T) BTree<T>::search( const T & e ) {
    BTNodePosi(T) v = _root; _hot = NULL; //从根节点出发
    while ( v ) { //逐层查找
      Rank r = v->key.search( e ); //在当前节点对应的向量中（顺序）查找
      if ( 0 <= r  && e == v->key[r] ) return v; //若成功，则返回；否则...
      _hot = v; v = v->child[ r+1 ]; //沿引用转至对应的下层子树，并载入其根（I/O）
    } //若因!v而退出，则意味着抵达外部节点
    return NULL; //失败
}
```





性能

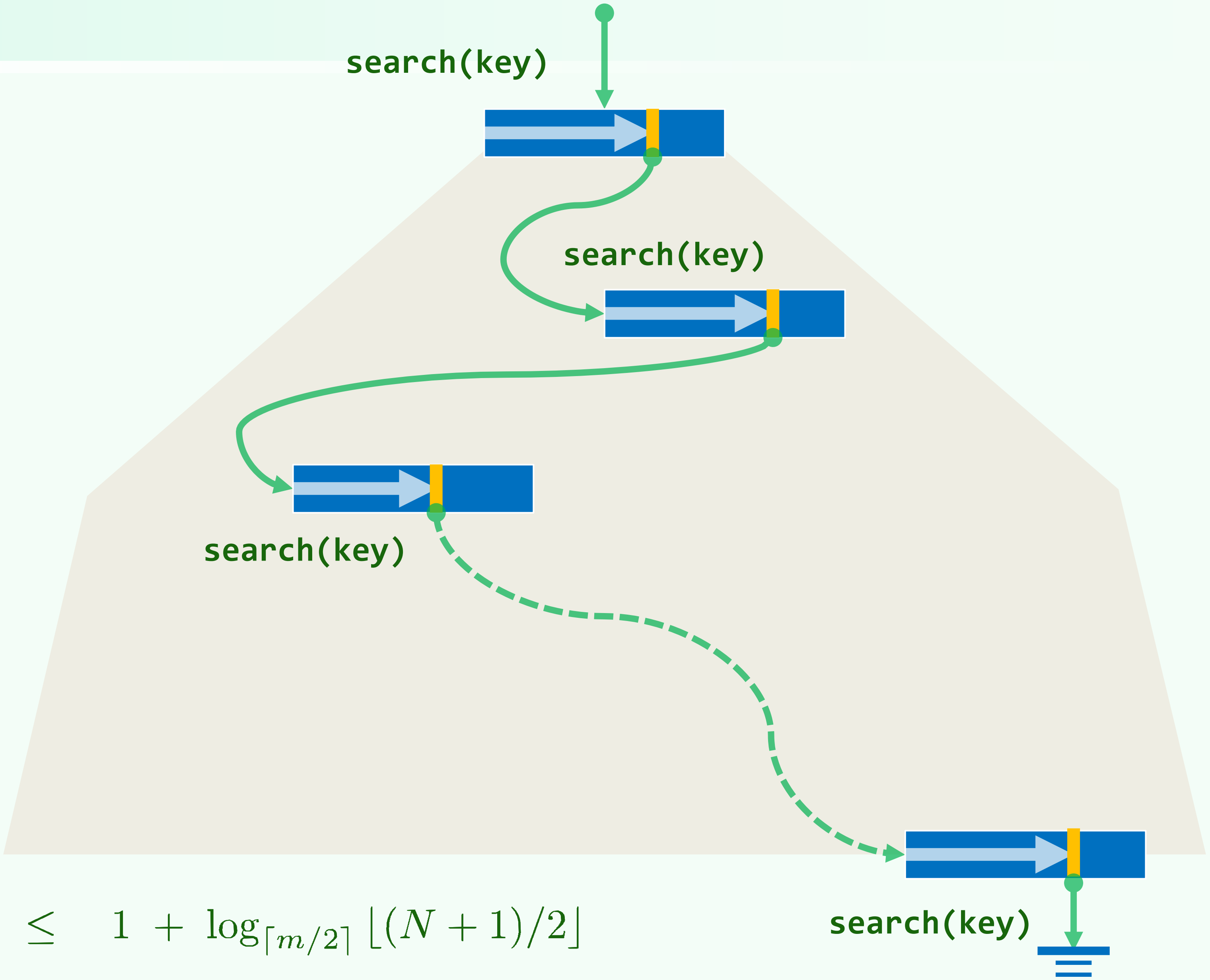
❖ 约定： 根节点常驻RAM

❖ 忽略内存中的查找

运行时间主要取决于I/O次数

❖ 在每一深度至多一次I/O

❖ 故运行时间 = $\mathcal{O}(\log n)$



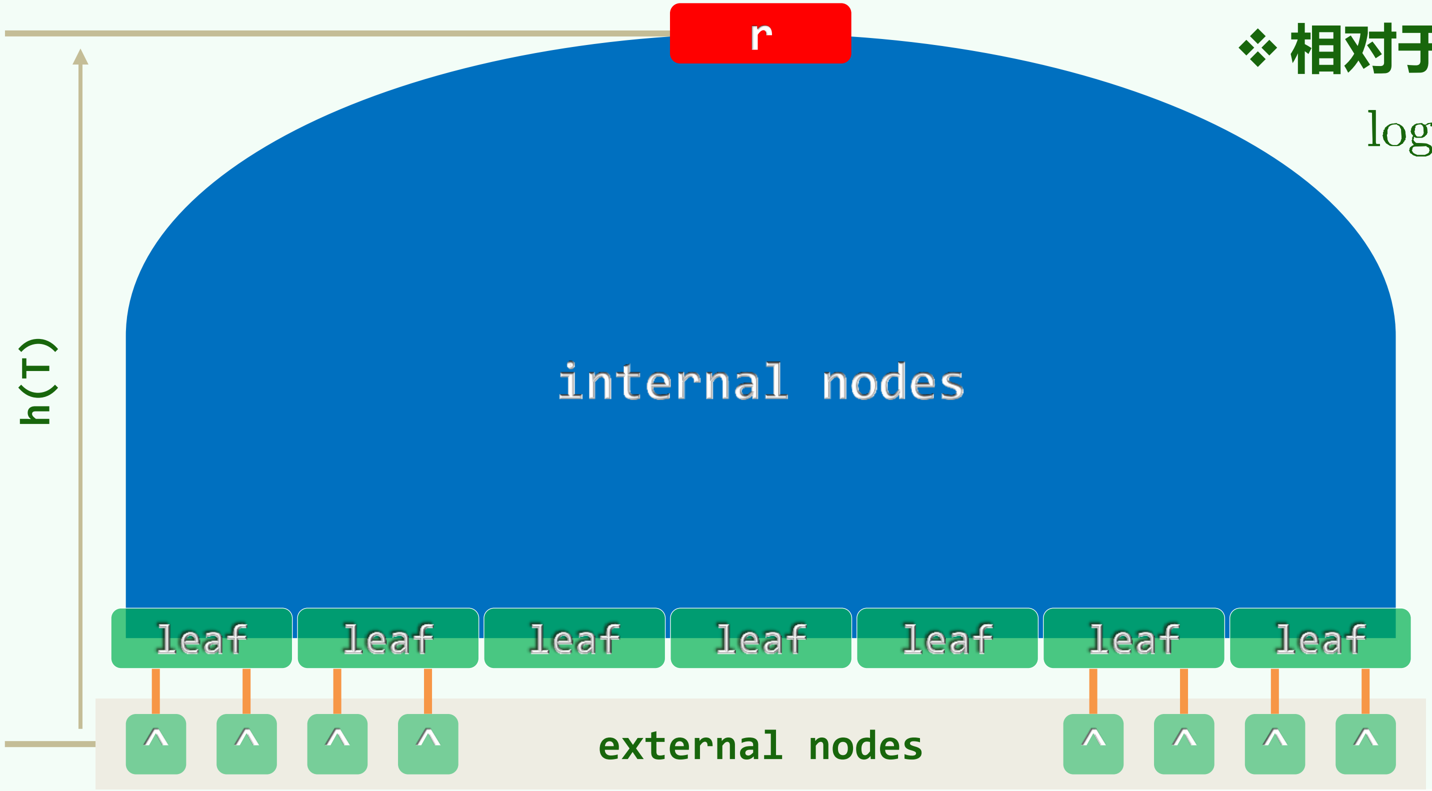
❖ 可以证明：$\log_m (N+1) \quad \leq \quad h \quad \leq \quad 1 + \log_{\lceil m/2 \rceil} \lfloor (N+1)/2 \rfloor$

最大树高

❖ 含有N个关键码的m阶B-树，可能有多高？

❖ 为此，内部节点应尽可能地“瘦”

$$n_k \ \geq \ 2 \times \lceil m/2 \rceil^{k-1}, \quad \forall\, k > 0$$



❖ 考查（所有）外部节点所在的那层：

$$N + 1 \ = \ n_h \ \geq \ 2 \times \lceil m/2 \rceil^{h-1}$$

$$h \ \leq \ 1 \ + \ \log_{\lceil m/2 \rceil} \lfloor (N+1)/2 \rfloor \ = \ \mathcal{O}(\log_m N)$$

❖ 相对于BBST：

$$\log_{\lceil m/2 \rceil} (N/2) \ / \ \log_2 N \ = \ 1/(\log_2 m - 1)$$

若取m = 256，树高约降低至1/7

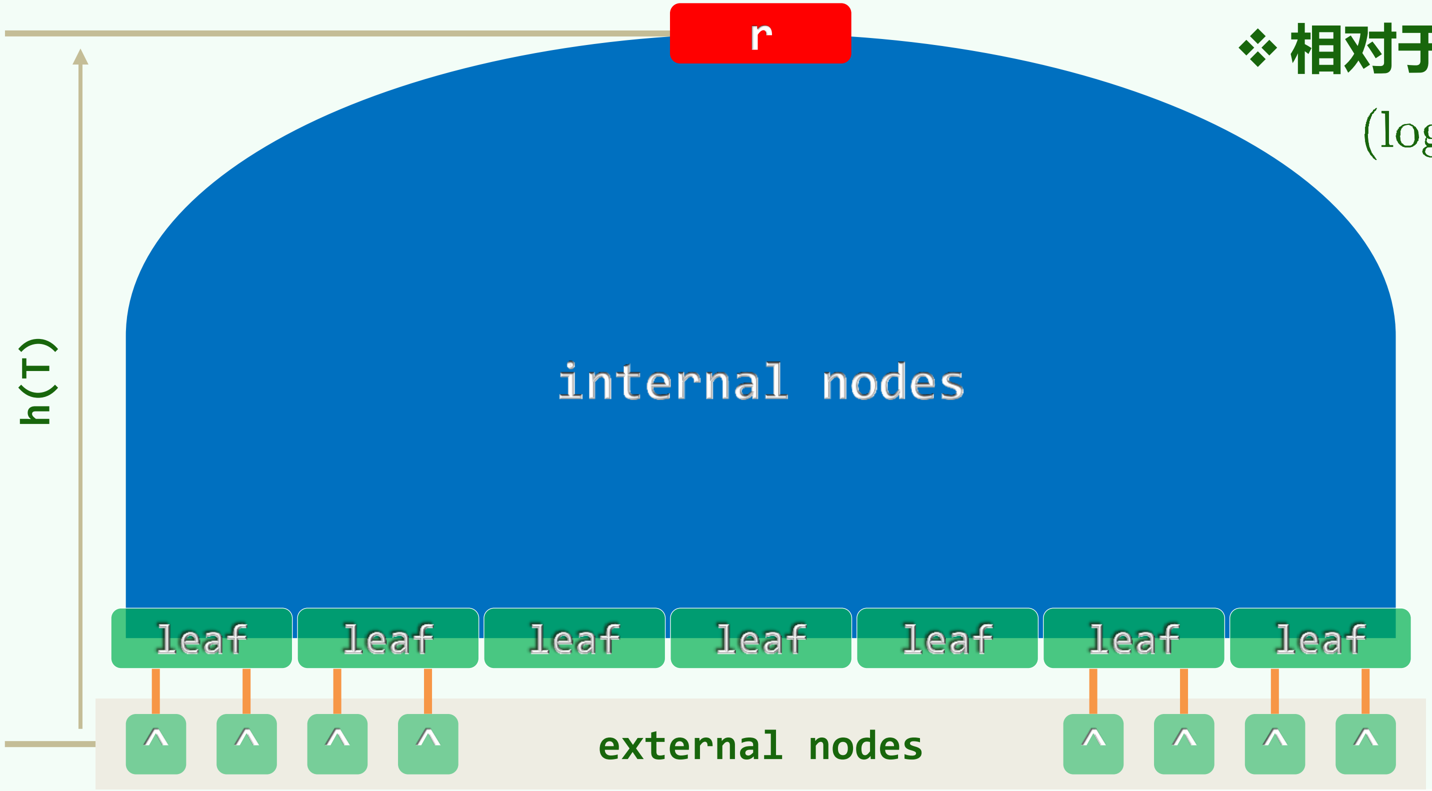
//用4年上完大学，还是28年？

最小树高

❖ 含N个关键码的m阶B-树，可能有多矮？

❖ 为此，内部节点应尽可能“胖”

$$n_k \leq m^k, \quad \forall\, k \geq 0$$



❖ 依然，考查（所有）外部节点所在的那层

$$N+1 = n_h \leq m^h$$

$$h \geq \log_m (N+1) = \Omega(\log_m N)$$

❖ 相对于BBST：

$$(\log_m N - 1) \,/\, \log_2 N = \log_m 2 - \log_N 2$$

$$\approx 1/\log_2 m$$

若取m = 256，树高约降低至1/8