

暑期算法培训

辛济远

8月4日

KMP算法

字典树

AC自动机

CSTRING和STL库

KMP 算法

参考：（next数组） http://www.ruanyifeng.com/blog/2013/05/Knuth%E2%80%93Morris%E2%80%93Pratt_algorithm.html
（匹配） <http://blog.csdn.net/yutianzuijin/article/details/11954939/>
<http://www.matrix67.com/blog/archives/115>

朴素（BF）字符串匹配

1、匹配

BBC ABCDAB ABCDABCDABDE
ABCDABD

BBC ABCDAB ABCDABCDABDE
ABCDABD

搜索词	A	B	C	D	A	B	D
-----	---	---	---	---	---	---	---

部分匹配值	0	0	0	0	1	2	0
-------	---	---	---	---	---	---	---

	A	B	C	D	A	B	D	
next	0	0	0	0	0	1	2	0

BBC ABCDAB ABCDABCDABDE
ABCDABD

BBC ABCDAB ABCDABCDABDE
ABCDABD

BBC ABCDAB ABCDABCDABDE
ABCDABD

BBC ABCDAB ABCDABCDABDE
ABCDABD

移动位数 =
已匹配的字符数 - 对应的部分匹配值

相较于朴素匹配，
KMP失配后会尽可能减少重复匹配过程。

2、NEXT数组

“部分匹配值”（NEXT）就是“前缀”和“后缀”的最长的共有元素的长度。

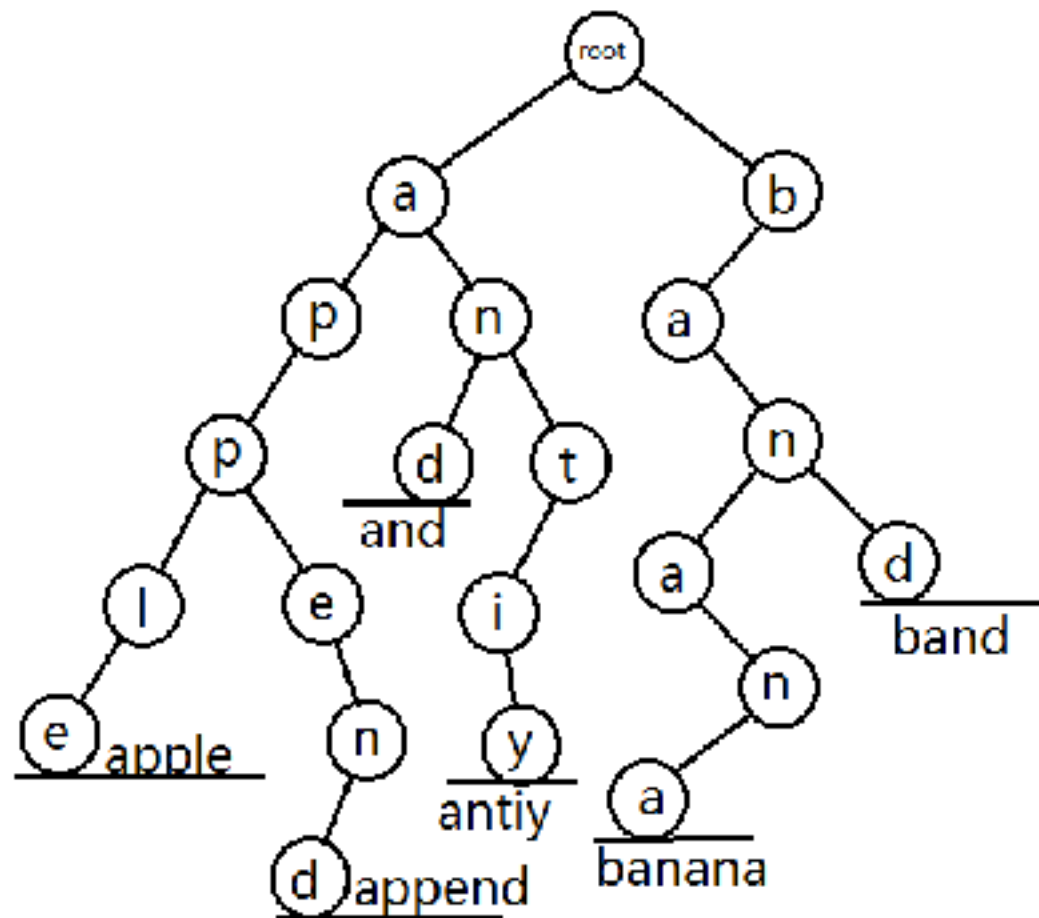
字符串：**“bread”**

前缀：**b , br , bre , brea**

后缀：**read , ead , ad , d**

- "A"的前缀和后缀都为空集，共有元素的长度为0；
- "AB"的前缀为[A]，后缀为[B]，共有元素的长度为0；
- "ABC"的前缀为[A, AB]，后缀为[BC, C]，共有元素的长度0；
- "ABCD"的前缀为[A, AB, ABC]，后缀为[BCD, CD, D]，共有元素的长度为0；
- "ABCDA"的前缀为[A, AB, ABC, ABCD]，后缀为[BCDA, CDA, DA, A]，共有元素为"A"，长度为1；
- "ABCDAB"的前缀为[A, AB, ABC, ABCD, ABCDA]，后缀为[BCDAB, CDAB, DAB, AB, B]，共有元素为"AB"，长度为2；
- "ABCDABD"的前缀为[A, AB, ABC, ABCD, ABCDA, ABCDAB]，后缀为[BCDABD, CDABD, DABD, ABD, BD, D]，共有元素的长度为0。

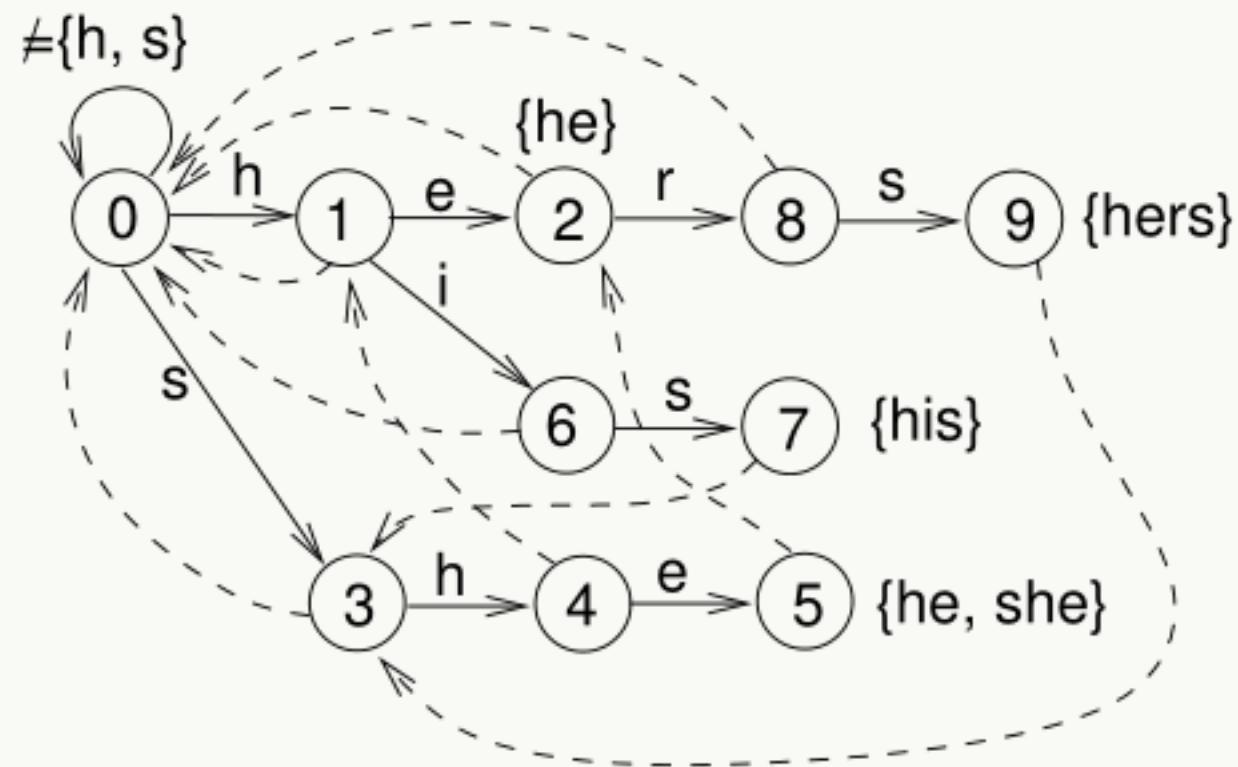
字典树



```
struct node {
    bool end;        //单词结束标记
    node *s[26];     //子节点指针
    char c;          //节点字符 (可以去除)
}
```


AC自动机

参考: <http://www.cppblog.com/mythit/archive/2009/04/21/80633.html>
<http://blog.csdn.net/niushuai666/article/details/7002823>



原理

● 前缀 & 后缀

● 广搜

```
struct node {  
    bool end;        // 单词结束标记  
    node *s[26];     // 子节点指针  
    char c;          // 节点字符 (可以去除)  
    node *fail;      // 失配指针  
}
```


CSTRING和STL库

<http://www.cnblogs.com/xFreedom/archive/2011/05/16/2048037.html>

- `string(const char *s);`
- `int length();`
- `string substr(int pos = 0,int n = npos)`
- `int find(const char *s, int pos = 0) const;`
- `string &replace(int p0, int n0,const char *s);`

- HDU 1711
- HDU 2594
- 洛谷 KMP模版
- HDU 2222
- HDU 2243
- 洛谷 AC自动机（两道）

【补充】其他字符串匹配算法

- BF算法（朴素、暴力匹配算法）
- BM算法 <http://www.cnblogs.com/xubenben/p/3359364.html>
- Sunday算法 <http://blog.csdn.net/qq575787460/article/details/40866661>
- BNDM算法 <http://www.itkeyword.com/doc/9616517912688594289/BNDM>

8月5日

树的遍历

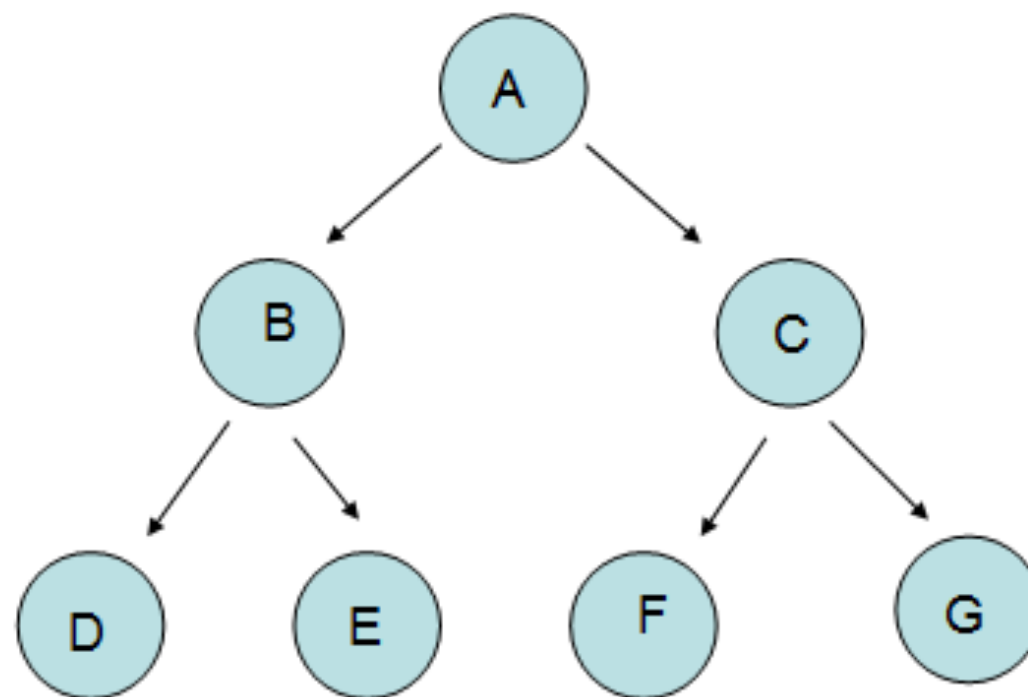
前缀（后缀）和数组

树状数组

线段树

STL库

二叉树的 遍历



先序 中序 后序

CodeVS 1029

前缀 (后缀) 和数组

构造

index	1	2	3	4	...	n
value	$s[1]$	$s[1]+s[2]$	$s[1]+s[2]+s[3]$	$s[1]+s[2]+s[3]+s[4]$		$s[1]+\dots+s[n]$

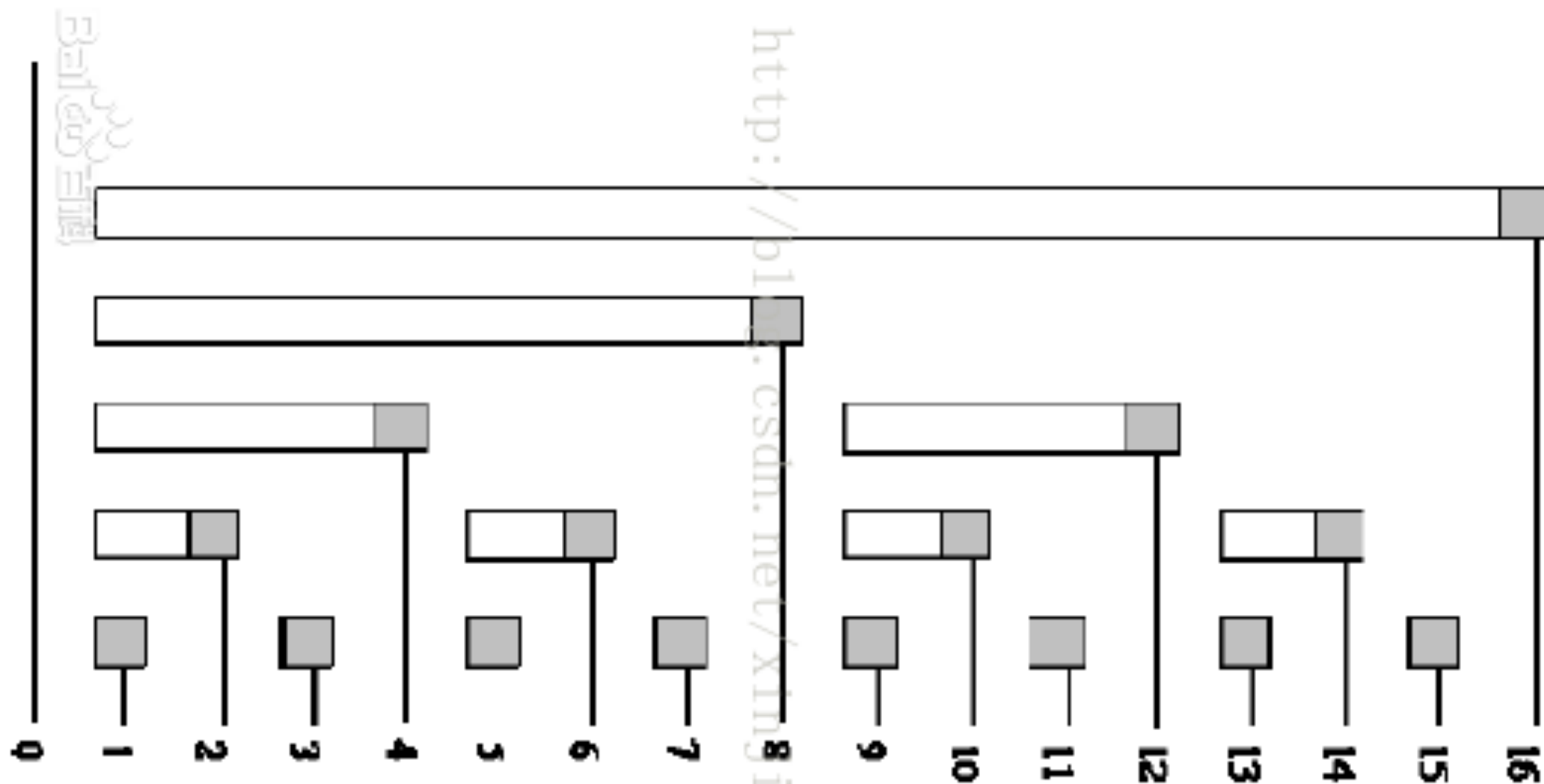
查询

查询 $s[n]+\dots+s[m]$ ($n \leq m$) 的值，等价于计算 $v[m] - v[n - 1]$ 。

NOIP2011 提高组 选择客栈

树状数组

- 修改操作 `add(int x);`
- 求和操作 `sum(int x, int y);`



LOWBIT函数

```
int lowbit(int x) {  
    return x & -x;  
}
```

参考：算法竞赛训练指南

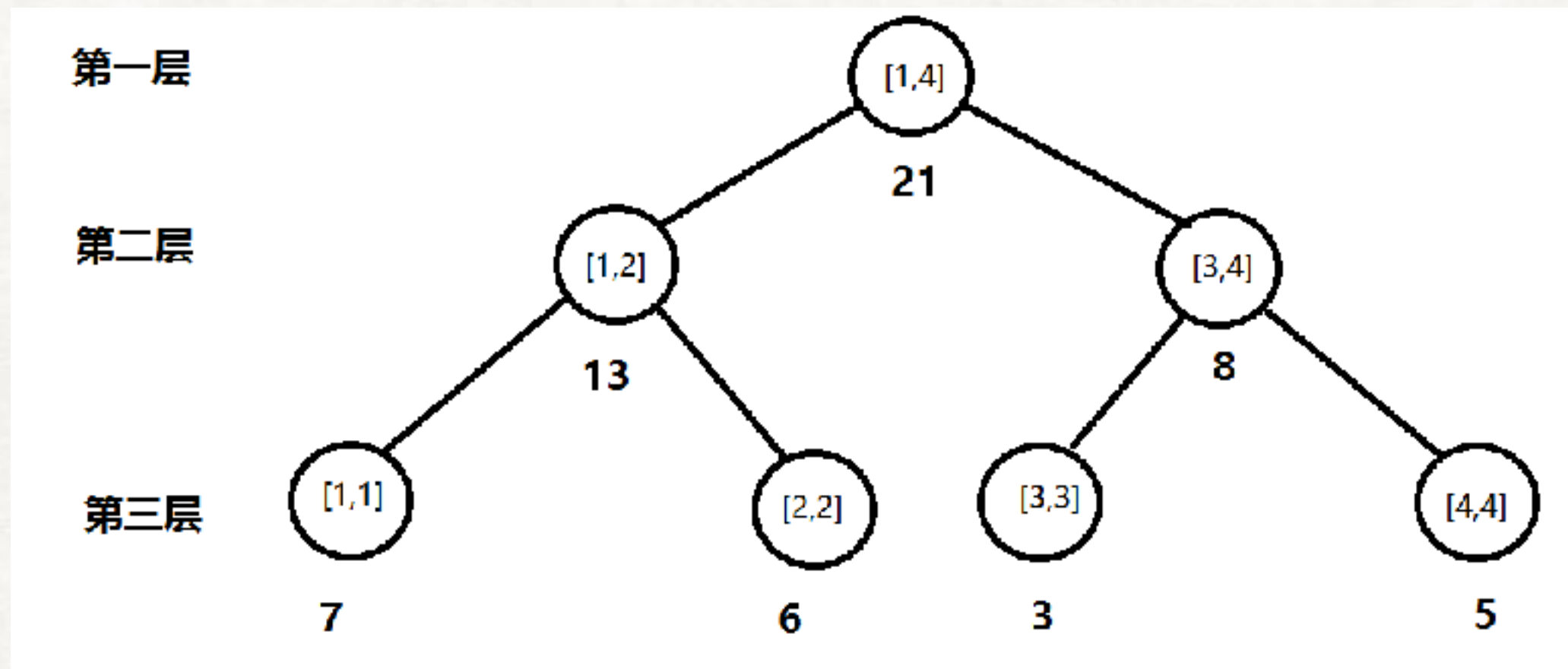
<http://blog.csdn.net/xinjiyuan97/article/details/53576678>

线段树

功能

- 修改操作 `add(int x);`
- 询问操作 `query(int x, int y);` (包括询问最大值、和、最小值)
- 段修改 `set(int x, int y);`
- 段修改 `add(int x, int y, int v)`

线段树的表示



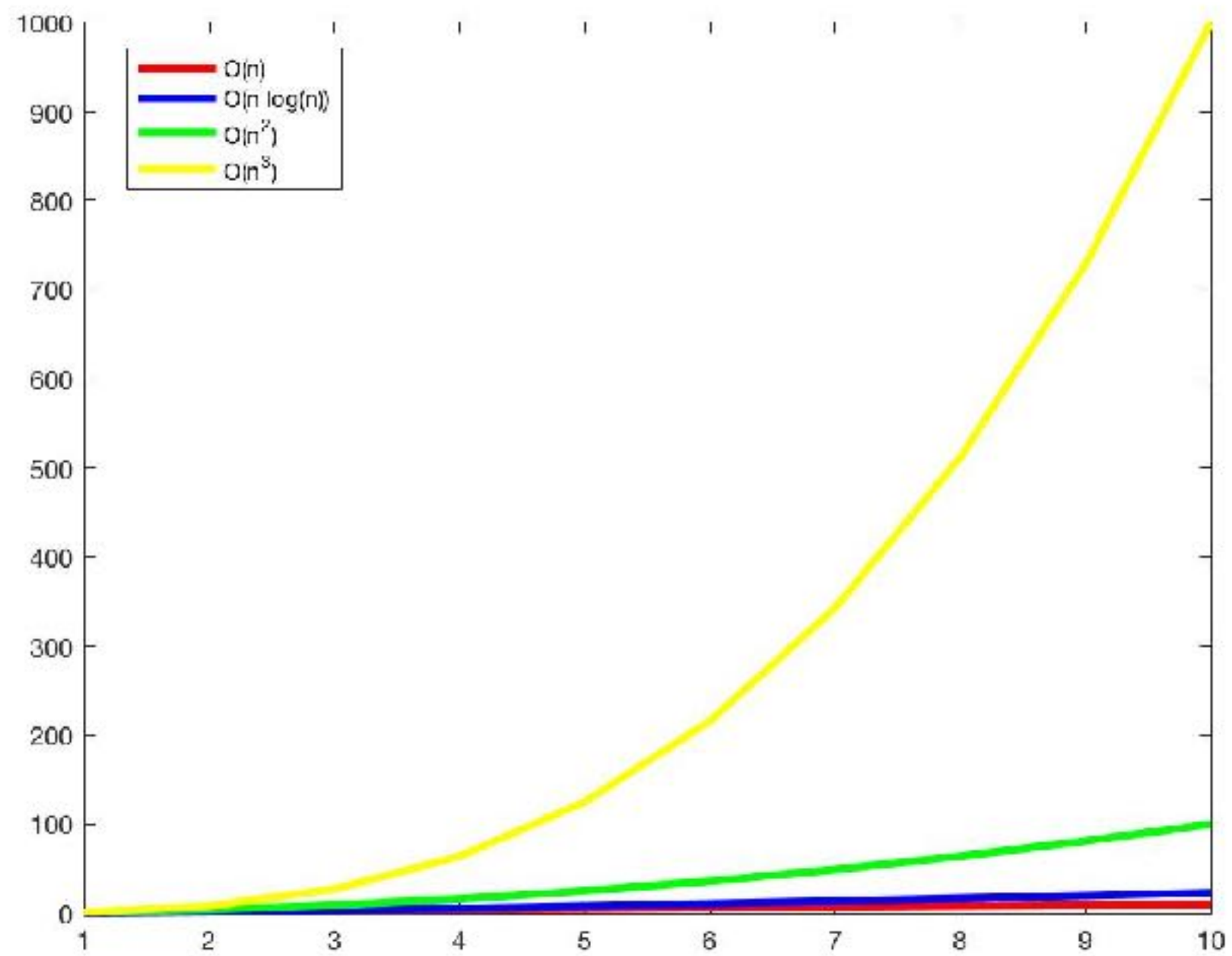
- 根结点 $s[1]$
- 父节点 $s[n]$, 左子节点 $s[2n]$, 右子节点 $s[2n + 1]$

线段树的段操作 (LAZY)

- `void add(int l, int r, int v)`
- `void set(int l, int r, int v)`

维护操作

```
void maintain(int o, int l, int r) {  
    int left = 2 * o, right = 2 * o + 1;  
    if (l < r) {  
        _sum[o] = _sum[left] + _sum[right];  
        _min[o] = min(_min[left], _min[right]);  
        _max[o] = max(_max[left], _max[right]);  
    }  
    _min[o] += _add[o];  
    _max[o] += _add[o];  
    _sum[o] += _add[o] * (R - L + 1);  
}
```



8月6日

二分查找

快速幂

二分答案

*有限自动机

*启发式搜索 (A*算法)

快速幂

NOIP2003 麦森数

二分答案

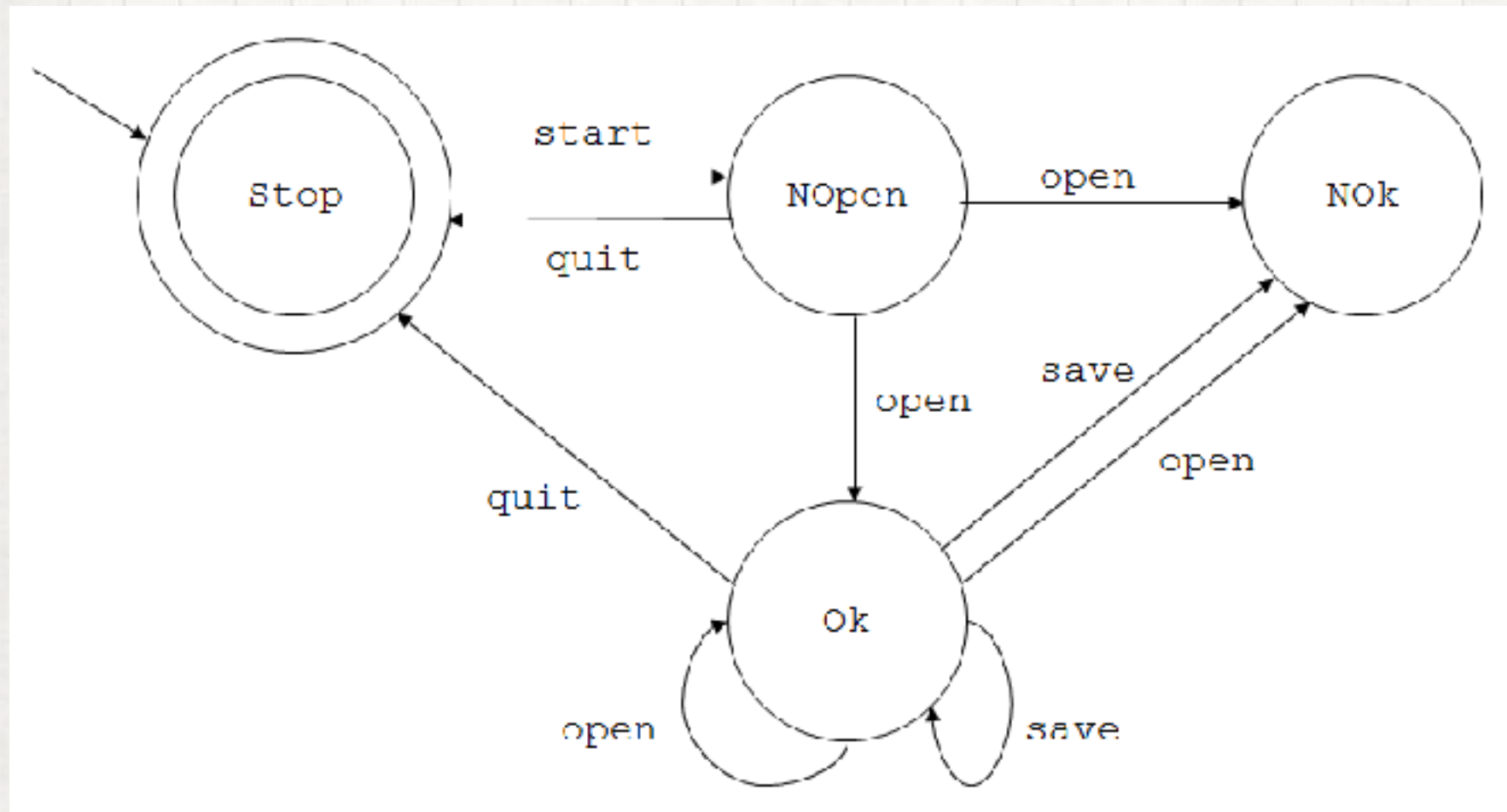
- 二分答案是对二分的一个扩展。每找到一个答案，判断其是否成立，若满足，则求更优解，否则，产生相对较劣解。
- “求最小的最大”
- 主要用于答案有明显的区间范围。

NOIP2011 提高组 聪明的质检员

NOIP2012 提高组 借教室

NOIP2010 提高组 关押罪犯

有限自动机



状态

停止 (HALT)

接受 (ACCEPT)

拒绝 (REJECT)

中断 (EXHAUSTED)

8月7日

二叉查找树

TREAP树

STL库 (SET和MAP)

二叉查找树

性质

- 若它的左子树不为空，则左子树上所有结点的值均小于它的根结点的值；
- 若它的右子树不为空，则右子树上所有结点的值均大于它的根结点的值；
- 它的左、右子树也分别为二叉查找树。

基本运算

插入

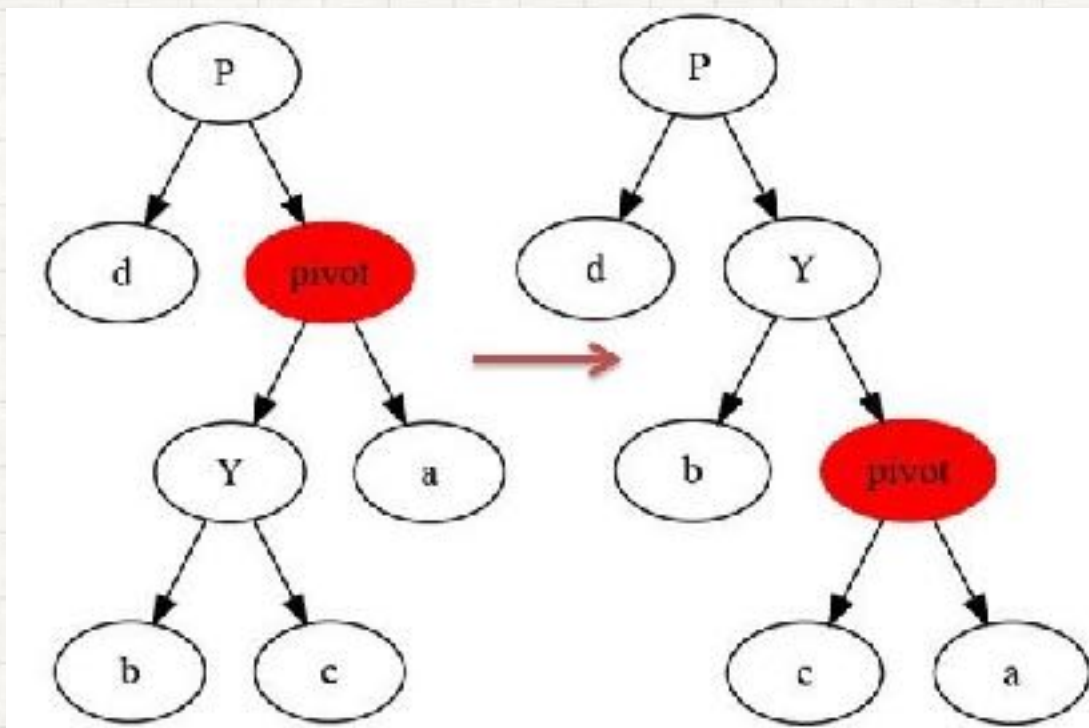
遍历

查找

删除

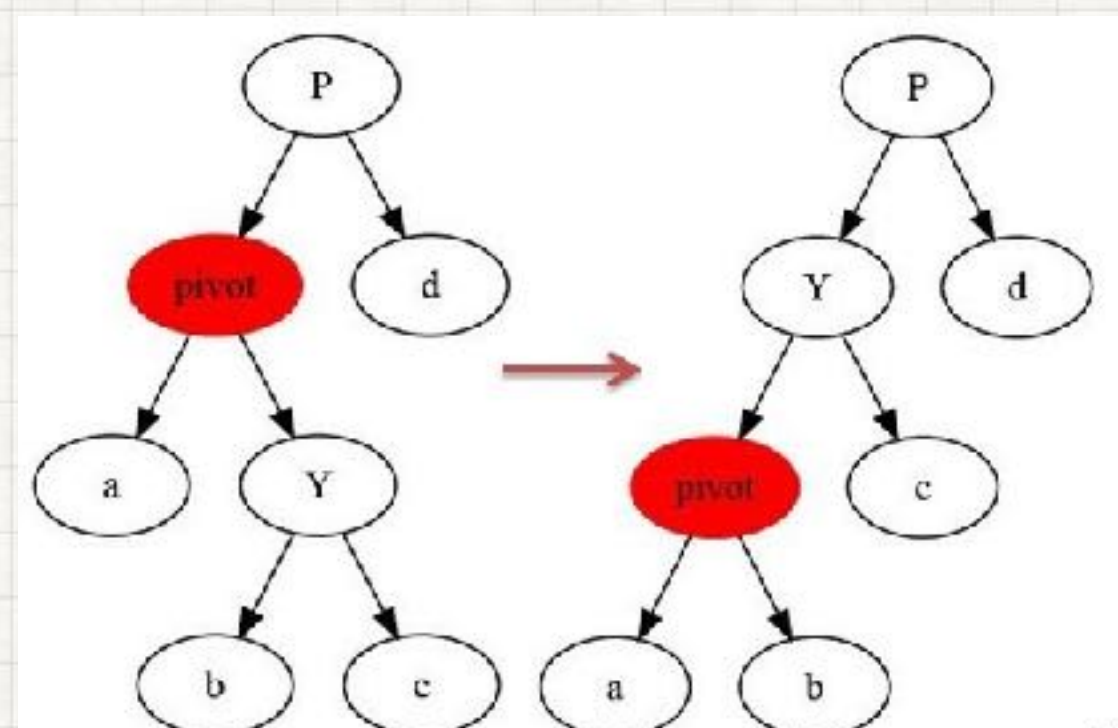
删除

- 若删除点是叶子结点，则设置其双亲结点的指针为空。
- 若删除点只有左子树，或只有右子树，则设置其双亲结点的指针指向左子树或右子树。
- 若删除点的左右子树均不为空，则
 - 查询删除点的右子树的左子树是否为空，若为空，则把删除点的左子树设为删除点的右子树的左子树。
 - 若不为空，则继续查询左子树，直到找到最底层的左子树为止。



右旋

```
void RRotate(node* p) {
    node lc = *p -> left;
    *p -> left = lc -> left;
    lc -> left = *p;
    *p = lc;
}
```



左旋

```
void LRotate(node* p) {
    node rc = *p -> right;
    *p -> right = lc -> right;
    lc -> right = *p;
    *p = rc;
}
```

TREAP树

-

<http://blog.csdn.net/u014634338/article/details/49612159>

STL库SET和MAP

- 头文件 <set>
- 声明 set<数据类型> S;
- 插入 S.insert(1);
- 查找 S.find(200)
- 遍历 迭代器
- 头文件 <map>
- 声明 set<数据类型1, 数据类型2> m
- m[x] = y;
- 查找 m.find(200)
- 遍历 迭代器

8月8日

拓扑排序

LCA及LCA倍增

强连通

TARJAN

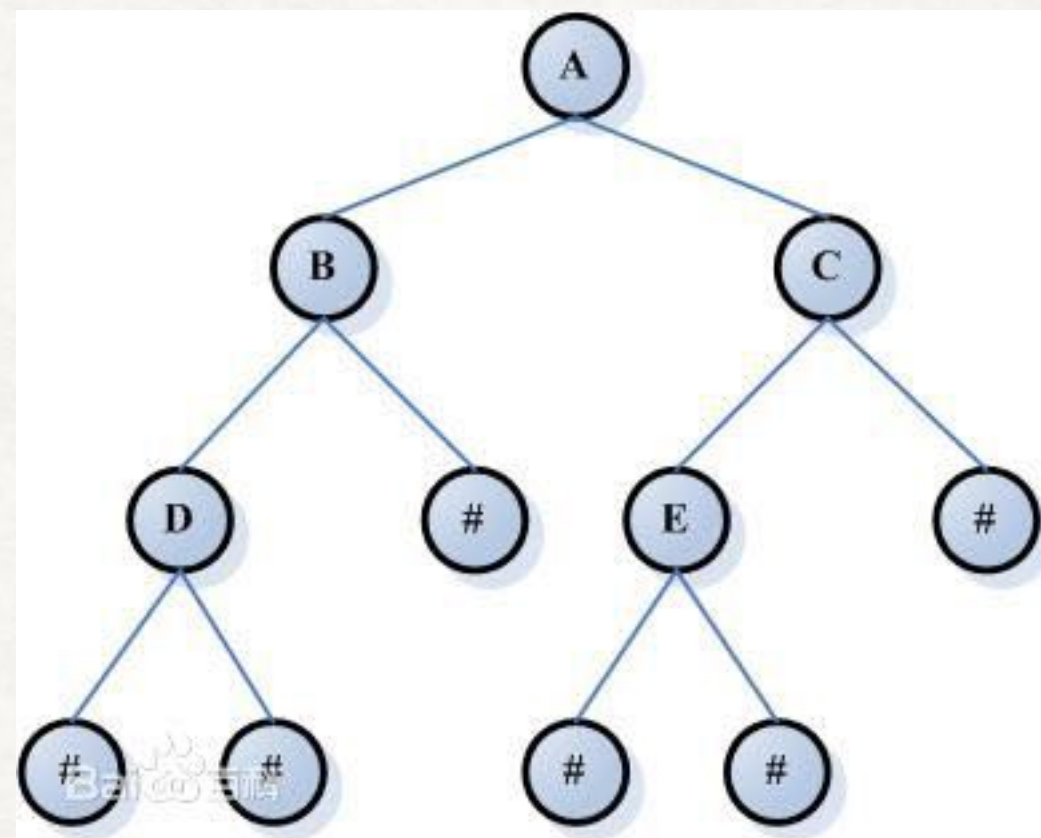
拓扑排序

- 拓扑排序是一个由偏序求全序的过程。
- 由AOV网构造拓扑序列的拓扑排序算法主要是循环执行以下两步，直到不存在入度为0的顶点为止。
 - (1) 选择一个入度为0的顶点并输出之；
 - (2) 从网中删除此顶点及所有出边。

课程代号	课程名称	先修课程
C1	高等数学	无
C2	程序设计基础	无
C3	离散数学	C1,C2
C4	数据结构	C3,C5
C5	算法语言	C2
C6	编译技术	C4,C5
C7	操作系统	C4,C9
C8	普通物理	C1
C9	计算机原理	C8

最小公共祖先(LCA)

- 朴素方法
- 树上倍增
- RMQ
- Tarjan算法



树

树上倍增

- $f(i, j)$ 表示从节点*i*向上爬 2^j 的高度所能够到达的节点。
- 二进制拆分

$$f(i, j) = f(f(i, j - 1), j - 1)$$

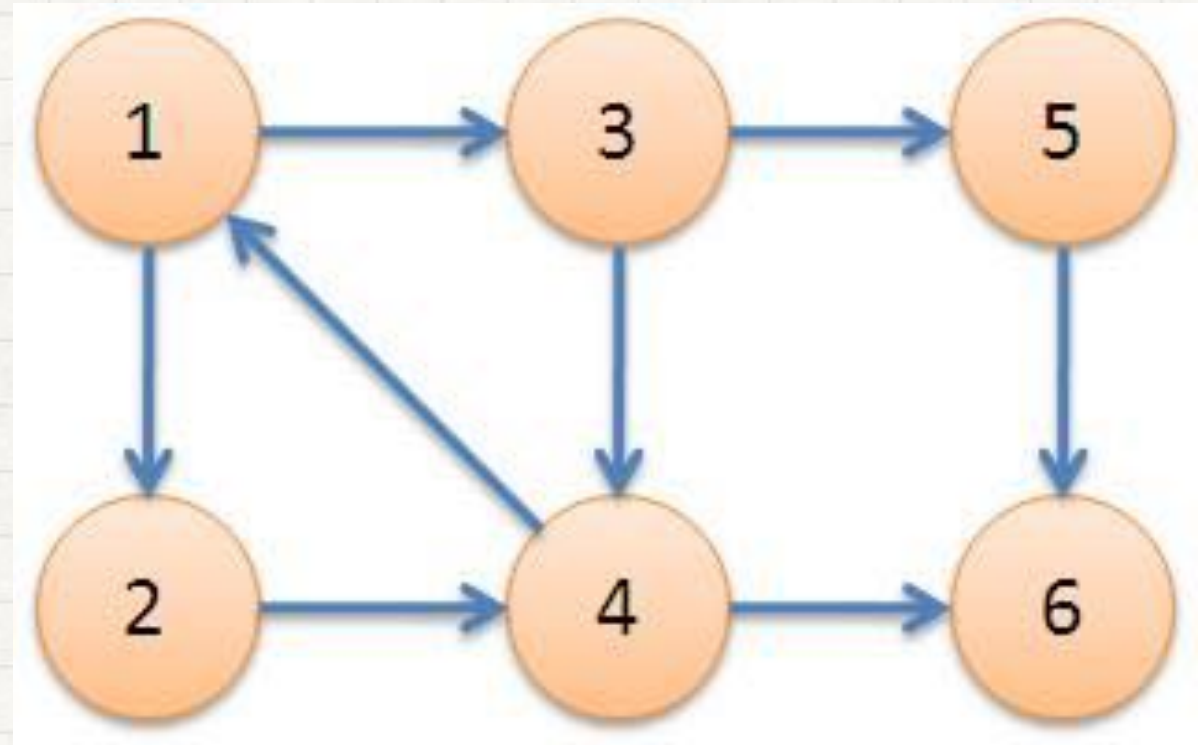
NOIP2016 天天爱跑步

NOIP2015 运输计划

NOIP2013 火车运输

强连通

- 强连通是指一个有向图中任意两点 v_1 、 v_2 间存在 v_1 到 v_2 的路径及 v_2 到 v_1 的路径。
- 有向图的极大强连通子图，称为强连通分量。



TARJAN算法 (强连通分量)

- <http://www.cnblogs.com/uncle-lu/p/5876729.html>
- <https://www.byvoid.com/zhs/blog/scc-tarjan>