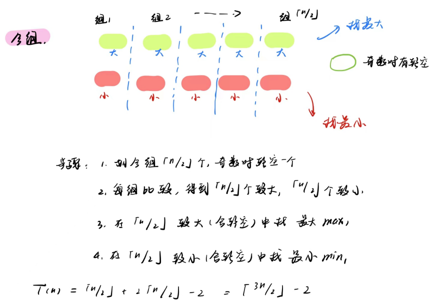
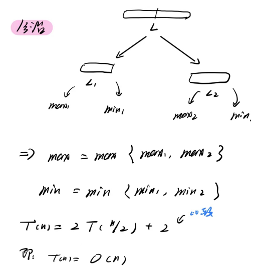
算法设计与分析期末

- **基础知识**
  - 算法分析
  - 函数增长
    - 渐近记号
      - 上界O
        - 存在常数c使，$0 \le f(n) \le cg(n)$,$g(n)$为上界
          - 更简单的理解是：
            - $T(n) = \begin{cases} cg(n_0), n = n_0 \\ \le cg(n), n > n_0 \end{cases}$
      - 下界Ω
        - 存在常数c使，$cg(n) \le f(n)$,$g(n)$为下界
      - 紧界Θ
        - 存在常数c_1, c_2，使$c_1 g(n) \le f(n) \le c_2 g(n)$,g(n)为紧界
      - 非紧上界o
        - 存在常数c使，$0 \le f(n) < cg(n)$,$g(n)$为非紧上界
      - 非紧下界ω
        - 存在常数c使，$cg(n) < f(n)$,$g(n)$为非紧下界
  - 分治
    - 求解递归式
      - 代入法
        - 数列求和
          - 等差
          - 等比
          - 调和数列：$ln(n+1) < 1 + 1/2 + 1/3 + ... + 1/n < ln(n) + 1$
          - 逐项积分 $\int_1^k f_n(x)$
          - 换元，令$n = 2^k$或$n = 3^k$
        - EXAMPLE
          - $T(n) = T(n/3) + T(n/4) + 5n \ \ \Theta(n)$
          - $T(n) = T(\lceil n/2 \rceil) + 1$
            - 放缩法：$T(n) \le T(\frac{n}{2} + 1) + 1$
            - 猜想：$T(n) \le clg(n-2)$
          -

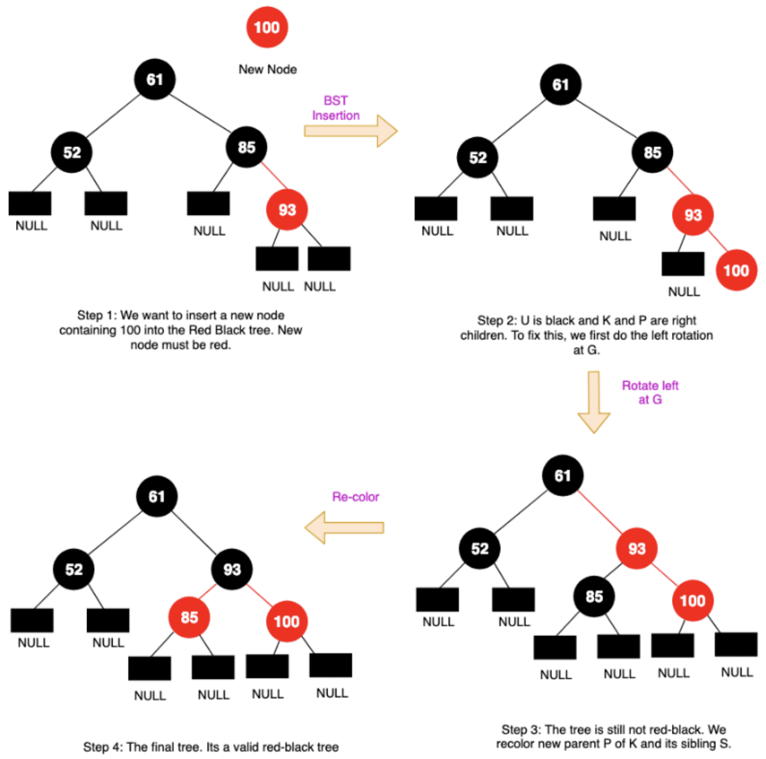
      - 主定理
        - Let $T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d)$ be a recurrence where $a \geq 1, b > 1$. Then,
          $$T(n) = \begin{cases} O(n^d \log n) & \text{if } a = b^d \\ O(n^d) & \text{if } a < b^d \\ O(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$
        - Let $T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$ be a recurrence where $a \geq 1,\ b > 1$. Then,
          - If $f(n) = O\left(n^{\log_b(a) - \epsilon}\right)$ for some constant $\epsilon > 0$, $T(n) = \Theta\left(n^{\log_b(a)}\right)$.
          - If $f(n) = \Theta\left(n^{\log_b(a)}\right)$, $T(n) = \Theta\left(n^{\log_b a} \log n\right)$.
          - If $f(n) = \Omega\left(n^{\log_b(a) + \epsilon}\right)$ for some constant $\epsilon > 0$ and if $af(n/b) \leq cf(n)$ for some c and all sufficiently large n, then $T(n) = \Theta(f(n))$.
      - 递归树
        - **EXAMPLE**
          - $T(n) = \begin{cases} 8T(\frac{n}{2}) + \Theta(1), n^2 > M \\ M, n^2 \leq M \end{cases}$
            - 层数：$h = lgn - lgM^{1/2}$ 最底层节点数$8^h = (n/M^{\frac{1}{2}})^3$,所以$\Theta(n^3/M^{1/2})$
      - 假设+数学归纳法
        - $T(n) = 4T(n/2) + n^2$
          - by the master method, we know $T(n) = \Theta(n^2 lgn)$. So the introduction hypothesis is $T(m) \geq cm^2 lgm$ for all $m < n$. And m = 1, valid. And then $m = k$ $T(n) \geq 4c(\frac{n}{2})^2 lgn + (1-c)n^2$
    - **EXAMPLE**
      - 最大子数组
      - 矩阵乘法Strassen
  - 概率分析和随机算法
    - 随机算法
    - **EXAMPLE**
      - 雇佣问题
      - 生日悖论
      - 球与箱子
      - 特征序列
- **排序和顺序统计量**

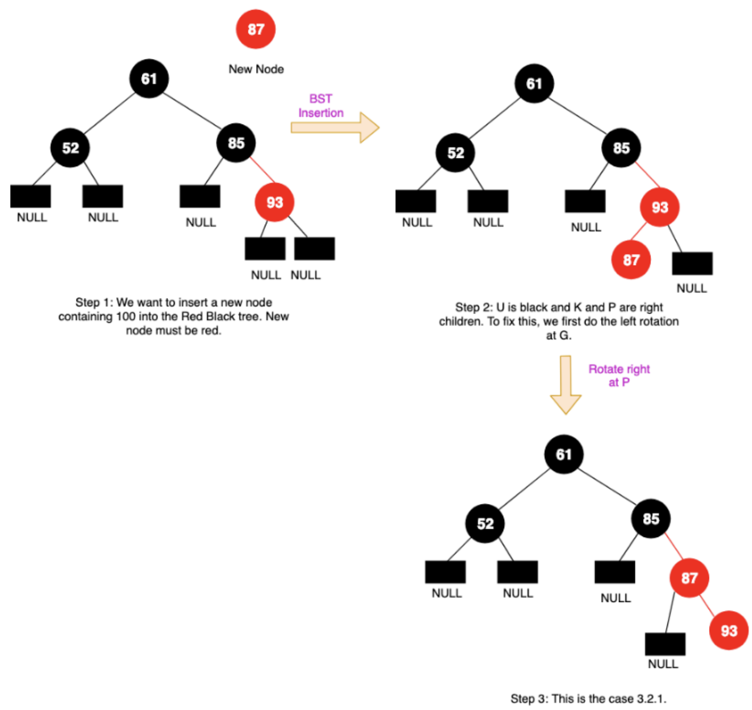
- 快速排序（$QuickSort$）
  - pivot和partition，手写遍历各步骤，游标i和j的变化，以及终止条件
- 归并排序($MergeSort$)
- 中位数和顺序统计量（竞标赛算法和$RandmizedSelect$）
  - 同时找最小值和最大值
    - normal：$T(n) = 2n - 3$
    - optimized:$T(n) = [\frac{3}{2}n] - 2$
      - 分组查询：
        -  
  - $RandmizedSelect$
    - random_pivot--->partition--->search
  - $BFTPR$ 寻找中位数的”中位数“算法
    - 5 5 分组，每组找中位数
- **二叉搜索树**
  - 查询
    - best：$T(n) = O(lgn)$
    - worst:$T(n) = O(n)$
  - 插入
    - 小在左，大在右，每个节点对下去找，新插入节点一定是叶子节点
  - 删除（三种情况）
    - 1.删除叶子节点（直接删除就行了）
    - 2.删除带有一个子节点的节点
      - 带有左节点
        - ”父连左“，让删除节点的父亲节点直接和删除节点的左孩子相连
      - 带有右节点
        - ”父连右“，让删除节点的父亲节点直接和删除节点的右孩子相连
    - 3.删除带有两个子节点的节点
      - 找到删除节点的右子树中的最左节点（其右子树中序遍历过程中的 第一个节点）
      - 把”右最左"节点放到删除节点位置上
- **AVL树**

  - 调平衡
    - 左旋
    - 右旋
  - 调平衡的四种情况
    - **LL**
      - 右旋中间节点
    - **RR**
      - 左旋中间节点
    - **LR**
      - 先变成**LL**，再右旋中间节点
    - **RL**
      - 先变成**RR**，再左旋中间节点
- **红黑树**
  - 基本原则：
    - 1.根节点为黑色
    - 2.**NIL**节点为黑色
    - 3.红节点的孩子和父亲都必须是黑色的
    - 4.黑高相同（任意一条从根节点出发走到NIL节点的路径，这条路径中包含的黑色节点的个数一定是相同的）
  - 性质：
    - $h \leq log_2(n+1)$
  - 插入：

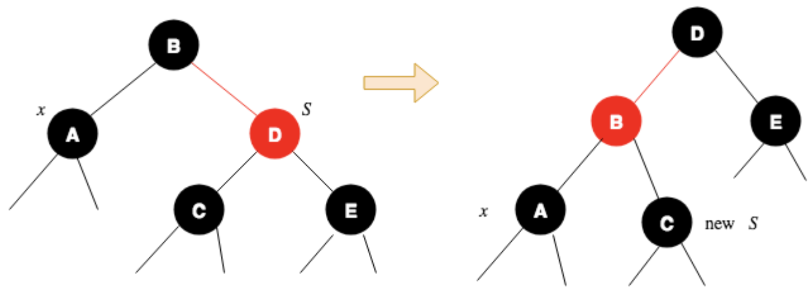
    

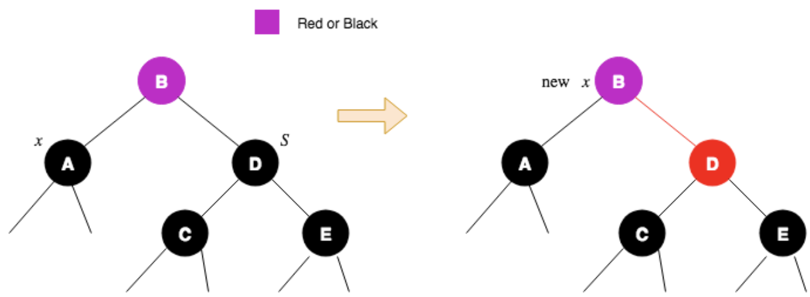
    - $Case1$：树空，直接插入为**黑**
    - $Case2$：**父节点为黑**，直接插入为红
    - $Case3$：**父节点为红**
      - $Case3.1$：**父亲节点和舅节点（父亲节点的兄弟）均为红色**；使父、舅节点调整为黑色，爷节点为红色；特例：爷节点为根节点的话不调整颜色
      - $Case3.2$：**父节点为红色，舅节点为黑色**；需要旋转来调整平衡状态

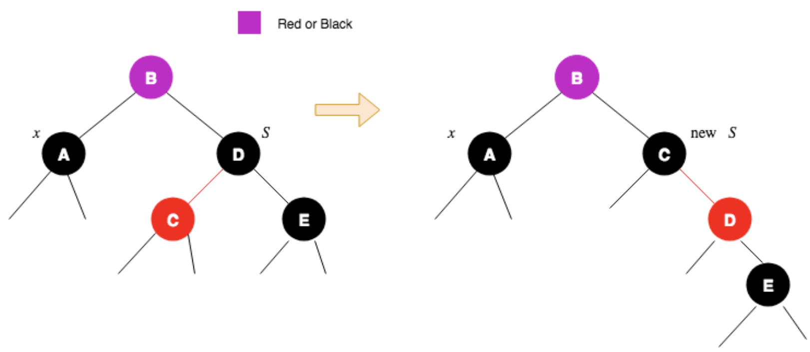
- $Case3.2.1:LL$ 爷为黑色，父（在左）为红，插入在父节点的右子树 红色
- $Case3.2.2:LR$ 爷为黑色，父（在左）为红，插入在父节点的右子树 红色
- $Case3.2.3:RR$ 爷为黑色，父（在右）为红，插入在父节点的右子树 红色
  - 

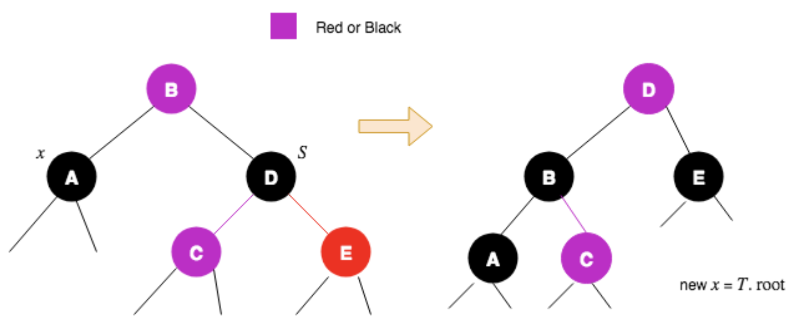
- $Case3.2.4:RL$ 爷为黑色，父（在右）为红，插入在父节点的左子树 红色
  - 

- 删除：
  - $Case1$: xx is a red node
    - 直接删除

- $Case2$: xx has a red child
  - 用红孩子换掉x然后把
- $Case3$: xx is a black node
  - $Case3.1$: xx's sibling SS is red
    - 

  - $Case3.2$: xx's sibling SS is black, and both of SS's children are black.
    - 

  - $Case3.3$: xx's sibling SS is black, SS's left child is red, and SS's right child is black.
    - 

  - $Case3.4$: xx's sibling SS is black, and SS's right child is red.
    - 

- 动态规划
  - 本质：分治递归+记忆存储

- **操作：**
    - 1.确定问题状态
        - 提取问题的最后一步
        - 转化子问题
    - 2.写出动态转移方程
    - 3.确定边界条件和终止条件---明确 base case
    - 4.确定计算顺序并计算
- **EXAMPLE**
    - 钢条切割
    - 矩阵链乘法
    - 最长公共子序列
        - LCS(string s1,int i,string s2,int j)
        - base:dp(s1,0,s2,0)
        - dp_equation:三种情况
    - 最优二叉搜索树

- **线性规划**
    - 线性规划的直觉
        - 线性规划的最优解会出现在可行区域的一个顶点上（当有所求规划和原可行域中线或面有平行时，仍然满足）---> n维线性空间的情况下，每个约束定义了n维空间的一个半平面，所有半平面的交集形成的可行域为单纯形，最优解在顶点上
    - 单纯性算法
        - 1.从单纯形的某个顶点开始遍历
        - 2.遍历顺序：沿n维空间的边移动到目标值不小于当前顶点的相邻顶点
        - 3.终止条件：所有相邻顶点的目标值都小于该顶点的目标值
    - 标准型和松弛型
        - 标准型

          最大化 $2x_1 - 3x_2 + 3x_3$

          满足约束

          $$\begin{array}{rcrcrcl} x_1 & + & x_2 & - & x_3 & \leqslant & 7 \\ -x_1 & - & x_2 & + & x_3 & \leqslant & -7 \\ x_1 & - & 2x_2 & + & 2x_3 & \leqslant & 4 \\ & & & & x_1, x_3, x_3 & \geqslant & 0 \end{array}$$

            - 求一个目标函数的最大值
            - 每一变量都有一个非负约束$x_1, x_2, x_3, x_4, x_5 \geq 0$
            - 所有约束都必须是线性的不等式
            - 除非负约束外，其余约束都必须是$\leq$ 而不是$\geq$

- 松弛型

$$\begin{aligned} \textit{maximize:} \quad & z = x_1 + 2x_2 \\ \textit{subject to:} \quad & x_3 = 3 + 3x_1 - 2x_2 \\ & x_4 = 2 - x_1 - x_2 \\ & x_5 = 1 - x_1 + x_2 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

  - 非负约束
  - 除非负约束之外，其余约束条件都是=
- 标准型化为松弛型：（增元替换，强加条件）
  - 举一个简单的例子：

$$\begin{gathered} maximize: z = x_1 + x_2 \\ subject\ to: -3x_1 + 2x_2 \leq 3 \\ x_1 + x_2 \leq 2 \\ x_1 + x_2 \leq 1 \\ x_1, x_2 \geq 0 \\ \text{标准型} --> \text{松弛型} \\ \text{令：} x_3 = 3x_1 - 2x_2 + 3 \geq 0 \\ x_4 = -x_1 - x_2 + 2 \geq 0 \\ x_5 = -x_1 - x_2 + 1 \geq 0 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \\ so\ we\ have\ antoher\ one: \\ \\ maximize: z = x_1 + x_2 \\ subject\ to: x_3 = 3x_1 - 2x_2 + 3 \\ x_4 = -x_1 - x_2 + 2 \\ x_5 = -x_1 - x_2 + 1 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{gathered}$$

- 目标值的具体求解
  - 单纯型算法的目的是为了能最直观的求出目标函数的最大值，以及导致目标函数最大值的个变量的值，因而当我们的目标函数满足如下形式时：$z = 20 - x_1 - x_2$ 由于 x_1 ,$x_2 \geq 0$ 所以当两个变量同时为0时就有最大值20，若他们中任意有一个变量有一个正数，那么函数值就是$20 - a\ constant$;所以综合的目的是把目标函数化成 $C - \Sigma_{i=1}^{n} c_i x_i$的形式
  - 具体操作可以是，如下初始化：

最大化 $3x_1 + x_2 + 2x_3$
满足约束：
$x_1 + x_2 + 3x_3 <= 30$
$2x_1 + 2x_2 + 5x_3 <= 24$
$4x_1 + x_2 + 2x_3 <= 36$
$x_1, x_2, x_3 >= 0$

⇒

z= $3x_1 + x_2 + 2x_3$
$x_4 = 30 - x_1 - x_2 - 3x_3$
$x_5 = 24 - 2x_1 - 2x_2 - 5x_3$
$x_6 = 36 - 4x_1 - x_2 - 2x_3$

- 基本解：$(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_6)$=(0, 0, 0, 30, 24, 36)
- 目标值z = (3 * 0 + 1 * 0 + 2 * 0 = 0
- 考虑增加x1的值，且使得所有值都仍然是正数
  - 易知：x1 超过 30时，x_4变负；x_1超过12时，x_5变负；x_1超过9时，x_6变负
  - 因此，互换x_1和x_6

- 然后找到一个限制最小的$x_1 = max[20(从x_4的式子得到), 12(从x_5的式子得到), 9(从x_6的式子得到)]$所以就换$x_1$和$x_6$就可以了，得到：（狗屁转动不要理他，就是变量替换吗，记住试卷上写pivot的你怎么做就行了，没那么复杂）
  - $z = 27 + x_2/4 + x_3/2 - 3x_6/4$
  - $x_1 = 9 - x_2/4 - x_3/2 - x_6/4$
  - $x_4 = 21 - 3x_2/4 - 5x_3/2 + x_6/4$
  - $x_5 = 6 - 3x_2/2 - 4x_3 + x_6/2$
- 算法具体实现（python）
  -

- NP问题
  -
- **难题和错题**
  -

  (b) If a data structure supports an operation `foo` such that a sequence of n `foo`'s takes $\Theta(n \log n)$ time to perform in the worst case, then the amortized time of a `foo` operation is $\Theta\left(\boxed{}\right)$, while the actual time of a single `foo` operation could be as high as $\Theta\left(\boxed{}\right)$.

  Answer: `Amortized time` $\Theta(\log n)$, `worst-case time` $\Theta(n \log n)$.

  - 奇怪的变量就换元，比如$lglgn\ \sqrt{M}$

  (d) $T(n) = T(\sqrt{n}) + \Theta(\lg \lg n)$

  let m = lglgn也可

  Solution: Change of variables: let $m = \lg n$. Recurrence becomes $S(m) = S(m/2) + \Theta(\lg m)$. Case 2 of master's theorem applies, so $T(n) = \Theta((\lg \lg n)^2)$.

  -

(g) $T(n) = T(n/2 + \sqrt{n}) + \sqrt{6046}$

I try by substitution method, but find it difficult to prove if T(n) = T(n/2), then T(n) = Ω(lgn)

Solution: By induction, $T(n)$ is a monotonically increasing function. Thus, for large enough n, $T(n/2) \le T(n/2+\sqrt{n}) \le T(3n/4)$. At each stage, we incur constant cost $\sqrt{6046}$, but we decrease the problem size to atleast one half and at most three-quarters. Therefore $T(n) = \Theta(\lg n)$.

T(n) >= T(n/2), 然后用master method.

-

(i) $T(n) = T(n/5) + T(4n/5) + \Theta(n)$

Solution: Master's theorem doesn't apply here. Draw recursion tree. At each level, do $\Theta(n)$ work. Number of levels is $\log_{5/4} n = \Theta(\lg n)$, so guess $T(n) = \Theta(n \lg n)$ and use the substitution method to verify guess.

In the $f(n) = \Theta(n)$ term, let the constants for $\Omega(n)$ and $O(n)$ be n_0, c_0 and c_1, respectively. In other words, let for all $n \ge n_0$, we have $c_0 n \le f(n) \le c_1 n$.

- First, we show $T(n) = O(n)$.
  For the base case, we can choose a sufficiently large constant d_1 such that $T(n) < d_1 n \lg n$.
  For the inductive step, assume for all $k < n$, that $T(k) < d_1 n \lg n$. Then for $k = n$, we have

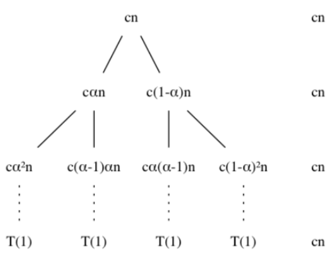
$$
\begin{aligned}
T(n) &\le T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + c_1 n \\
&\le d_1 \frac{n}{5} \lg\left(\frac{n}{5}\right) + d_1 \frac{4n}{5} \lg\left(\frac{4n}{5}\right) + c_1 n \\
&= d_1 n \lg n - \frac{d_1 n}{5} \lg 5 - \frac{4 d_1 n}{5} \lg\left(\frac{5}{4}\right) + c_1 n \\
&= d_1 n \lg n - n\left(\left(\frac{\lg 5 + 4\lg(5/4)}{5}\right) d_1 - c_1\right).
\end{aligned}
$$

  The residual is negative as long as we pick $d_1 > 5c_1/(\lg 5 + 4\lg(5/4))$. Therefore, by induction, $T(n) = O(n \lg n)$.

-

Use a recursion tree to give an asymptotically tight solution to the recurrence T(n) = T(αn) + T((1 - α)n) + cn, where α is a constant in the range 0 <α < 1 and c > 0 is also a constant.

`Answer`



可以假设α < 1/2,因此树的高度有$\log_{1/\alpha} n$

$$T(n) = \sum_{i=0}^{\log_{1/\alpha} n} cn + \Theta(n) = cn \log_{1/\alpha} n + \Theta(n) = \Theta(n \lg n)$$

-

$$T(n) = 5T(n/5) + n/\lg n$$

b.
 $T(n) = 5T(n/5) + \frac{n}{\lg n} = 25T(n/25) + 5\frac{n/5}{\lg(n/5)} + \frac{n}{\lg n} = 25T(n/25) + \frac{n}{\lg n - \lg 5} + \frac{n}{\lg n} = nT(1) + \sum_{i=0}^{\lg n - 1} \frac{n}{\lg n - i \lg 5} = nT(1) + n\sum_{i=1}^{\lg n} \frac{1}{\lg n} = \Theta(n \lg \lg n)$

-

$$T(n) = 3T(n/3+5) + n/2$$

d.
 by applying master method $\Theta(n \lg n)$

-

2、复数a+bi可以用数对(a, b)表示（其中$i^2=-1$）。描述一个方法，只需构造进行三次实数乘法（可采用有限次实数加减），即可计算a+bi和c+di相乘的结果数对(e, f)。请写出采用了哪三次实数乘法？（5分）

参考解答：

$$
\begin{aligned}
(a,b)(c,d) &= ac+(ad+bc)i+bd(-1) \\
&= ac+[(a-b)(d-c)+ac+bd]i-bd
\end{aligned}
$$

三次实数乘法分别是：$ac, (a-b)(d-c), bd$

-

---

d. $T(n)=3T(n/3-2)+n/2$

n足够大时，原式转化为

$$T(n)=3T(\frac{n}{3})+\frac{n}{2}$$

根据主定理 case2

$$T(n)=\Theta(n\lg n)$$

-

i. $T(n)=T(n-2)+1/\lg n$

与前两题同理，研究对数积分$li(x)$，当$x\to\infty$时，函数有如下渐近表现：

$$li(x)=O(\frac{x}{\ln(x)})$$

(其完整的渐近展开式为$li(x)=\frac{x}{\ln x}\sum_{k=0}^{\infty}\frac{k!}{(\ln x)^k}$)

不难验证：

$$\sum_{i=1}^{n}\frac{1}{\lg i}\geq\frac{n}{\lg n}$$

因此

$$T(n)=\Theta(\frac{n}{\lg n})$$

-
-
-
-
-