日期: /

1. 计算T(n)的渐近上下界 并证明合法

解: (1) $T(n) = 4T(n/3) + n\lg n$

$f(n) = n\lg n$ 且有 $a=4,\ b=3$

$n^{\log_b a} = n^{\log_3 4} \approx n^{1.2618}$

$\therefore \forall\ 0<\varepsilon \le 0.26$, 有 $n\lg n = \Omega(n^{\log_3 4-\varepsilon})$, 由主方法,

$\therefore T(n) = \Theta(n^{\log_3 4})$

(补充): 反 $\lim\limits_{n\to\infty} \dfrac{n^a}{n\lg n} = \lim\limits_{n\to\infty} \dfrac{n^{a-1}}{\lg n} = \lim\limits_{n\to\infty} \dfrac{(a-1)n^{a-2}}{\frac{1}{n\ln 10}} = \lim\limits_{n\to\infty} (a-1)\ln 10 \cdot n^{a-1}$

考 $\lim\limits_{n\to\infty} \dfrac{n^a}{n\lg n} = \infty$ 时 不可说 $n\lg n = O(n^a)$

而 $\lim\limits_{n\to\infty} (a-1)\ln 10 \cdot n^{a-1} = \infty$ 时, 需满足 $a>1$

即 $\lg 3-\varepsilon > 1$

(2) $T(n) = T(n-2) + n^2$

猜想: $T(n) = \Theta(n^3)$. 证明如下:

① 证 $T(n) = O(n^3)$

若 $T(n) = O(n^3)$, 则 $T(n-2) \le c\cdot(n-2)^3$

则 $T(n) \le c\cdot(n-2)^3 + n^2$

$\le cn^3 + (1-6c)n^2 + 12cn - 8c \le c'n^3$

$\therefore T(n) \le O(n^3)$

证毕

② 证 $T(n) = \Omega(n^3)$.

若满足, 则 $T(n-2) \ge c\cdot(n-2)^3$

则 $T(n) \ge c\cdot(n-2)^3 + n^2$

$\ge cn^3 + (1-6c)n^2 + 12cn - 8c \ge c'n^3$

$\therefore T(n) = \Omega(n^3)$

综上, $T(n) = \Theta(n^3)$

$(n^2-4n+4)(n-2)$

$c(n^3-6n^2+12n-8)$

$+n^2$

日期: /

(3) $T(n)=2T(\frac{n}{2})+n^4$

令 $a=2,\ b=2,\ f(n)=n^4,\ \varepsilon=1>0,\ c=\frac{1}{8}<1$

则 $f(n)=n^4=\Omega(n^{\log_b a+\varepsilon})=\Omega(n^2)$

且 $af(\frac{n}{b})=2\cdot(\frac{n}{2})^4=\frac{n^4}{8}\le\frac{1}{8}\cdot n^4$

由主定理.

$T(n)=\Theta(f(n))=\Theta(n^4)$

(4). $T(n)=T(7n/10)+n$

令 $a=1,\ b=\frac{10}{7},\ \varepsilon=1>0,\ c=\frac{7}{10}<1,\ f(n)=n$

则: $f(n)=n=\Omega(n^{\log_b a+\varepsilon})=\Omega(n)$

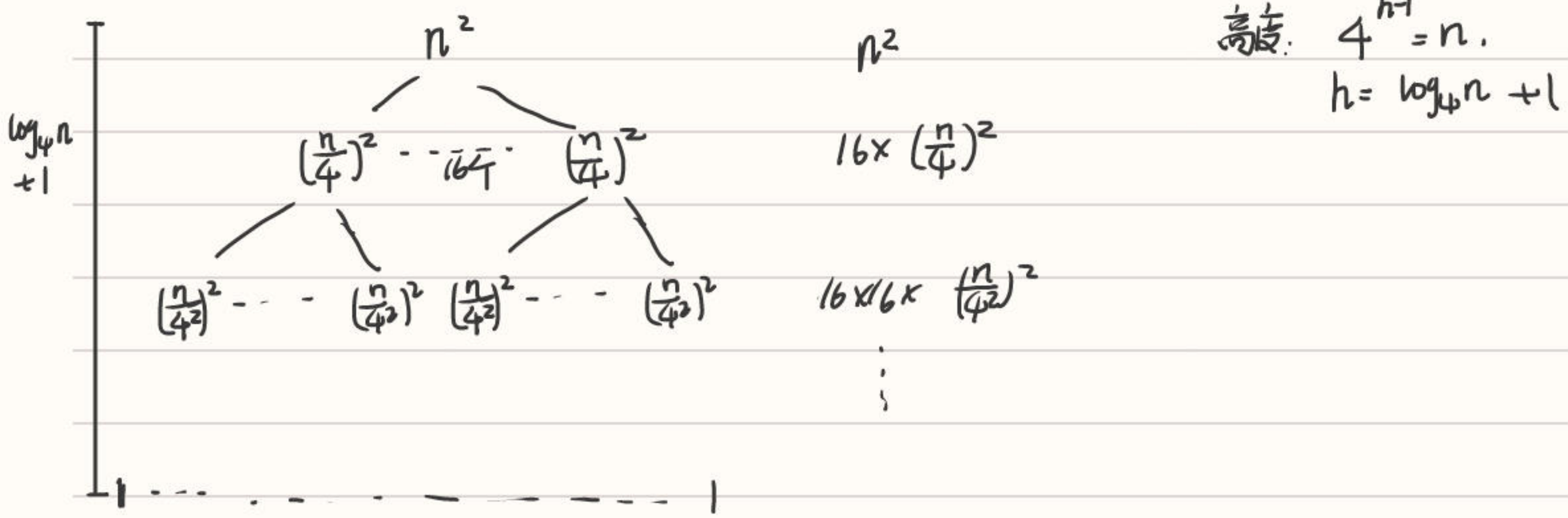
且 $af(\frac{n}{b})=\frac{7}{10}n\le cf(n)=\frac{7}{10}n$

由主定理

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

(5) $T(n)=16T(n/4)+n^2$

树方法:



$$\sum_{i=0}^{\log_4 n} 16^i\times(\frac{n}{4^i})^2=(\log_4 n+1)\cdot n^2=\Theta(n^2\lg n)$$

日期: /

2. (1) 贝尔曼方程:

设 $f(n)$ 是 目标值为 n 时的最少步骤

则

$$f(n)=\begin{cases} f(n-4)+1, & n\%3!=0,\ n-4>0 \\ f(n/3)+1, & n\%3=0,\ n-4\le 0 \\ \min\{f(n-4), f(n/3)\}+1, & n\%3=0,\ n-4>0 \\ \text{无解} & \\ 0 & \end{cases}$$

无解: $n=2$ 或 $n=4$ } $n\%3!=0,\ n-4\le 0$

$n=1$

(2) 伪代码 (自底向上):

```
MAIN( n , f )
  for x ← 1 to n
      do f[x] ← -1
  f[1] ← 0
  for x ← 1 to n
      do if x%3 = 0 and (x-4) > 0
      then f[x] = min{f[n-4], f[n/3]} +1
      else if x%3 =0 and (x-4) ≤ 0      //x=3
      then f[x] = f[n/3] +1
      else if x%3 != 0 and (x-4) > 0
      then f[x] = f[n-4] +1
return f[n]          //若-1表示无路径. 否则为最小步数.
```

日期: /

贪心算法

3. 证明分数背包问题有贪心选择性质

在分数背包问题中，我们每次取单位重量 (v_i/w_i) 最大的商品，取 $\min\{w_i, W\}$ 重量（其中 W 为背包剩余容量）

反证法：设按上述方式取得物品顺序为 $g_i, g_{i+1}, \cdots, g_{i+k}$

假设在某时刻未取 w 重量 g_{i+k}，而是取了另一个 w 重量的 $g_{i+j} \neq g_{i+k}$，得到最大价值

根据上述算法，此时满足 $v_{i+k}/w_{i+k} > v_{i+j}/w_{i+j}$

即 $w \cdot (v_{i+k}/w_{i+k}) > w \cdot (v_{i+j}/w_{i+j})$

显然用 g_{i+j} 代替得到的方案不是最大价值，与上述假设矛盾

所以分数背包问题有贪心选择性质

---

4. 集合 A, B 都含 n 个正整数，对其重排使 $\prod_{i=1}^{n} a_i^{b_i}$ 最大，给出算法证明正确性并写出运行时间。

解：此算法：将正整数集合 A, B 按降序排列：

```
MAX_RESULT (A,B)
    Q_SORT (A, n)
    Q_SORT (B, n)
Q_SORT (S, begin, end)   //快速排序
    key ← S[begin] ;   i ← begin ;   j ← end
    while i < j
        while i<j and S[j] >key     j--
        if ic j
        then  S[i] = S[j] ; i++
        while icj and S[i]<key     i++
        if i cj
        then  S[j] = S[i] ;  j--
    S[i] = key
    Q_SORT(S, begin, i-1)
    Q_SORT(S, i+1, end)
```

日期: /

(2)运行时间:

对快速排序来说, $T(n) = 2T(\frac{n-1}{2}) + n$,

由主定理, $T(n) = \Theta(n\log n)$

而算法为两次快速排序, 故时间复杂度为 $\Theta(n\log n)$

(3)正确性证明:

首先将A,B集合中的元素排序.

$A=\{a_1, a_2, \cdots, a_n\}$ $B=\{b_1, b_2, \cdots, b_n\}$ 且 $\forall 0<i<j\le n$, $a_i<a_j$, $b_i<b_j$

从 $\prod_{i=1}^{n} a_i^{b_i}$ 中任取两组, $a_i^{b_i}a_j^{b_j}$ 且 $i<j$

假设顺序改为 $a_i^{b_j}a_j^{b_i}$

则 $\frac{a_i^{b_i}a_j^{b_j}}{a_i^{b_j}a_j^{b_i}} = a_i^{(b_i-b_j)}a_j^{(b_j-b_i)} = \frac{a_j^{(b_j-b_i)}}{a_i^{(b_j-b_i)}} = \left(\frac{a_j}{a_i}\right)^{(b_j-b_i)}$

$\because j>i \quad \therefore \frac{a_j}{a_i}>1$ 且 $b_j-b_i>0$

$\therefore \frac{a_i^{b_i}a_j^{b_j}}{a_i^{b_j}a_j^{b_i}} = \left(\frac{a_j}{a_i}\right)^{(b_j-b_i)} > 1$

$\therefore a_i^{b_i}a_j^{b_j} > a_i^{b_j}a_j^{b_i}$ $\therefore$ 不可用 $a_i^{b_j}a_j^{b_i}$ 替换之.

证毕

---

5. 换硬币, 每个硬币的值为整数.

(a) 10分, 5分, 25分, 1分

伪代码: SOLUTION_A(n, array) // array | 0 | 1 | 2 | 3 ; number | 1分 | 5分 | 10分 | 25分

```
while n != 0
  do if n >= 25 then n -= 25 ; array[3]++ ;
     else if n >= 10 then n -= 10 ; array[2]++ ;
     else if n >= 5 then n -= 5 ; array[1]++ ;
     else if n >= 1 then n -= 1 ; array[0]++ ;
```

日期: /

b) 给出证明，

首先，此贪心算法的准则是从所有硬币中挑出能给出的最大面值的硬币.

由于使用 C 个 C^i 的硬币可以用 1 个 C^{i+1} 硬币代替，因此 C^i 硬币数 $n_i \le C-1$

C^0	C^1	C^2	...	C^k
n_0	n_1	n_2	...	n_k

对于需要找按 n 价的钱，若用贪心选择，则会选面值 $j=\max\{0\le i\le k = C^i \le n\}$

若不用贪心，则会以 $\sum_{i=0}^{j+1} n_i C^i = n$ 方式选择硬币

$\because n \ge C^j$，所以 $\sum_{i=0}^{j+1} n_i C^i = n \ge C^j$。由于 $n_i \le C-1$

故 $\sum_{i=0}^{j+1} n_i C^i \le \sum_{i=0}^{j+1} (C-1) C^i$

$\le (C-1) \cdot \frac{1-C^j}{1-C}$

$\le C^j - 1 \le C^j$，矛盾

因此必须用贪心

证毕.

c) 举例：1分、5分，7分.

找 10分，若用贪心为 {7, 1, 1, 1}

而存更优解 {5, 5}.

d) 伪代码：自底向上　　任意 k 种：不一定满足贪心 —— 动态归划.

设 $f(n)$ 为找值为 n 时的最少硬币数. 则：

$f(j) = \min\{f(j), f(i-C^j)+1\}$ // 情况当用上一枚 C^i 作比较

```
MIN-RESULT (C, n, f)   //C[i] 表示第 i 种 硬币的面额
    for i ← 1 to n
    then  f[i] ← +∞
    f[1] ← 1      f[0] ← 0
    for i ← 1 to n
    do  for j ← 1 to k
        if (i - c[j]) ≥ 0  and  f[i] > (f[i - c[j]] + 1)
        then  f[i] = f[j - c[i]] + 1
```

日期: /

BFS & DFS:

6. DFS 在 22.6上 如何运作？（考察顶点搜索再表顺序）.

顶点的发现与完成时间；边的分类.

流程分析：

condition	time
find q	d[q]=1
find s	d[s]=2
find v	d[v]=3
find w	d[w]=4
end w	f[w]=5
end v	f[v]=6
end s	f[s]=7
find t	d[t]=8
find x	d[x]=9
find z	d[z]=10
end z	f[z]=11
end x	f[x]=12
find y	d[y]=13
end y	f[y]=14
end t	f[t]=15
end q	f[q]=16
find r	d[r]=17
find u	d[u]=18
end u	f[u]=19
end r	f[r]=20

结果：

node	begin	end
q	1	16
r	17	20
s	2	7
t	8	15
u	18	19
y	13	14
w	4	5
x	9	12
v	3	6
z	10	11

边的分类：

树边：<q,s>、<s,v>、<v,w>、<q,t>、<t,x>、<x,z>、<t,y>、<r,u>

反向边：<w,s>、<y,q>

正向边：<q,w>

交叉边：<r,y>、<u,y>

7. 强连通分支算法过程：

(1)用DFS计算各点结束时间

(2)计算 G^T

(3)在 G^T 上按结束时间降序运行DFS：

visit r | visit u | visit q → visit y → visit t |

visit x → visit z | visit s → visit w → visit v

∴各树有：(q, y, t)、(s, v, w)、(x, z)、r、u.

日期: /

(1) 第10行算的结束时间

node	end
q	16 ✓
r	20 ✓
s	7
t	15✓
u	19 ✓
y	14 ✓
w	5
x	12 ✓
v	6
z	11 ✓

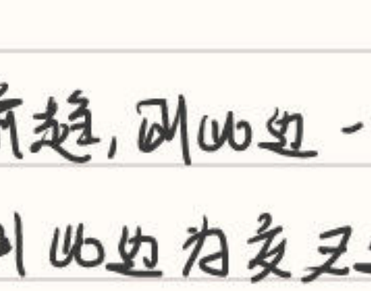
(2) 产生的森林：

(r) , (u) ,

q → y → t

s → w → v

x → z



8. BFS — 广度优先

a. 无向图的 BFS 中

1. 无反向边和正向边：

即证 图中边 要么是树边，要么是交叉边

任取边 (u,v) ∈E， 点 v 必在 u 完成前被发现.

当 u 被发现后：① v 是白色， 此时 v 被加入队列，边 (u,v) 为树边

② v 是灰色，说明 u 与 v 同属一个已探寻的父节点

此时 (u,v) 为交叉边

③ v 是黑色，若 v 是 u 的前趋，则此边一定已被探寻，此边为树边；若 u,v 是同一层，则此边为交叉边

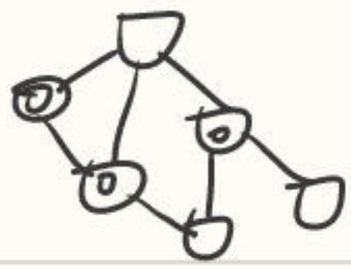
证毕

2. 每个树边 (u,v)，有 v.d = u.d+1 ：

根据算法，当 u 为灰色并探寻到白色顶点 v 时，v 被加入队列，边 (u,v) 对会成为树边。此时算法保证了 d[v] ← d[u]+1

证毕

日期: /



3. 每个交叉边 (u,v) 有 v.d = u.d 或 v.d = u.d+1

根据 a.1 条证明的情况，发现点 u 后，(u,v) 为树边可能.

① v为灰色，此时 u,v 皆灰色，说明它们来自同一个节点，即 v.d=u.d

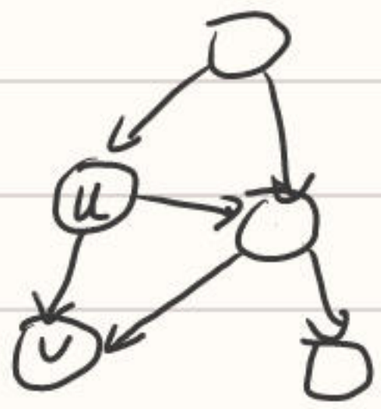
② v为黑色，此时 v 已被探寻完毕，v 可能来自同一节点，或高一层，v.d=u.d+1

或反证：①若 d[v]-d[u] >1，即表示 v 的深度比 u 多1以上，但根据算法，不会存在这样的边

②若 d[u]-d[v] >1，表示 u 深度多，这条边会成为树边

∴ |d[u]-d[v]| ≤ 1

b. 有向图 BFS:



1. 无正向边

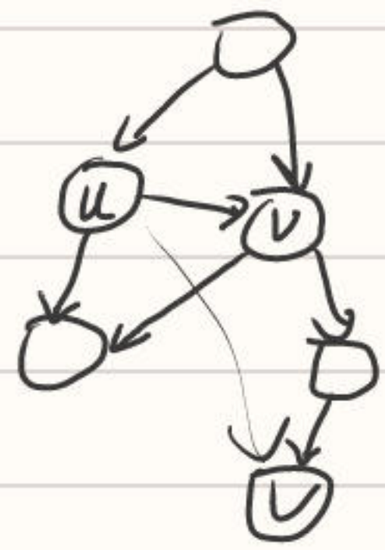
假设 BFS 后存在正向边 (u,v)，即 u 是 v 的前趋但 (u,v) 非树边

当开始探寻点 u 时，由于 (u,v) 非树边，故 v 此时为灰色或黑色

①若 v 为灰色，证明此时 u,v 皆在队列中，又因为 u 是 v 的前趋，只能是探寻 u 时发现的 v，此时 (u,v) 为树边，矛盾

②若 v 为黑色，而 u 是 v 的前趋，说明探寻 u 时 v 已探寻结束，不满足 u 是 v 的前趋. 矛盾

综上，不存在正向边



2. 树边 (u,v) v.d = u.d+1:

同a.2，根据算法，若 (u,v) 是树边，说明探寻 u 时发现 v，会执行 d[v]=d[u]+1 操作

证毕

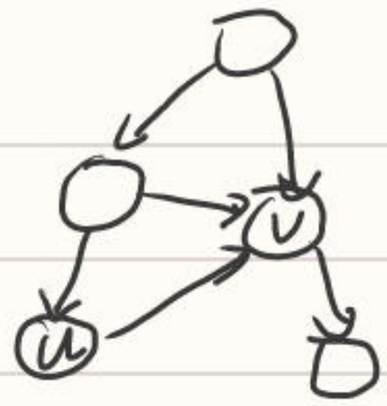
3. 横向边 (u,v) v.d ∈ u.d+1

假设 v.d > u.d+1，当探寻到 u 时，根据推论，v 不会在队列中，但 (u,v) 边存在，所以 v 只可能为黑色

而 v 为黑色说明探寻 u 时 v 已探寻完，则 d[v] ≤ d[u]，与上述矛盾

证毕

日期: /



4. 反向边 (u,v) : $0 \le v.d \in u.d$

假设 $v.d > u.d$, 说明探寻到 u 时,

v 为灰色或黑色 (若为白色, 不会为反向边)

①若 v 为灰色, 此时 v 与 u 皆在队列中, 而 $\because v$ 是 u 的祖先(反向边),

只可能 v 探寻到了 u, 为树边 (u,v) 矛盾

②若 v 为黑色, 说明探寻 u 时 v 已探寻完毕. 但由于 $v.d > u.d$ 这与

v 先探寻完毕矛盾.

证毕.

日期:　　/