算法基础HW1

PB19071535徐昊天

问题1

答:

辗转相除法的时间复杂度上界为O(loga+logb)。

分析过程如下:

①a<b时

a mod b = a，下一轮迭代(b,a)，进入下一种情况。

②a≥b时

根据欧几里得算法，迭代一次后得到(b,a mod b)，迭代两次后得到(a mod b,b mod (a mod b))。

根据 求余符号 的性质，有:

a mod b < a / 2; b mod (a mod b) < b / 2

由以上结论可得，经过两次迭代后，a与b都将至少缩小1/2，a*b至少缩小至1/4。

则有:

$$4^{\frac{T(n)}{2}} \le a * b$$
$$\text{即} T(n) \le log(ab) = loga + logb$$

故欧几里得算法时间复杂度上界为O(loga+logb)，为线性时间。

问题2

答：函数从小到大的排序如下：

$$2^{\Theta(\frac{logn}{loglogn})} < \sqrt{n} < log(n!) < n^2 < n^{\Theta(logloglogn)} < (logn)^{\Theta(logn)}$$

分析过程如下：

$$\begin{aligned}
&\text{显然有：} \sqrt{n} = \Theta(n^{\frac{1}{2}}), n^2 = \Theta(n^2) \\
&\text{根据斯特林公式：} n! = \sqrt{2\pi n} \cdot (\frac{n}{e})^n, \text{有：} \\
&log(n!) = log(\sqrt{2\pi n} \cdot (\frac{n}{e})^n) = \frac{1}{2}log(2\pi n) + nlogn - n = \Theta(nlogn) \\
&2^{\Theta(\frac{logn}{loglogn})} = \Theta(2^{\frac{logn}{loglogn}}) = \Theta(n^{\frac{1}{loglogn}}) \\
&n^{\Theta(logloglogn)} = \Theta(n^{logloglogn}) \\
&(logn)^{\Theta(logn)} = \Theta(logn^{logn}) = \Theta(n^{loglogn}) \\
&\text{根据增长的阶有如下排序：} \\
&n^{\frac{1}{loglogn}} < n^{\frac{1}{2}} < nlogn < n^2 < n^{logloglogn} < n^{loglogn} \\
&\text{故函数从小到大的排序为：} \\
&2^{\Theta(\frac{logn}{loglogn})} < \sqrt{n} < log(n!) < n^2 < n^{\Theta(logloglogn)} < (logn)^{\Theta(logn)}
\end{aligned}$$

问题3

答：

(a)先证：

> 若对于互质的数x,y，存在非负整数a、b使得n=a·x+b·y当且仅当不存在非负整数a′,b′使得xy-x-y-n=a′·x+b′·y。

证明如下：

首先假设存在ax + by = a′x + b′y （a≠a',b≠b'）

故有：(a - a′)x = (b′ - b)y,即存在非零整数m使得a′ = a + my 和 b′ = b - mx。

若a∈[0,y）， 则a' = a + my ∉ [0,y）。

由上可得对于任意整数 n 都至多具有一个 n = ax + by 形式的表示 （其中0 ≤ a < y， b ≥ 0 ）

令$n = a \cdot x + b \cdot y$，假设存在非负整数$a',b'$使得$xy - x - y - n = a' \cdot x + b' \cdot y$，则有：$(a + a')x + (b + b')y = xy - x - y$

根据以上证明得到的结论，$a + a'$在$[0,y)$中仅存在一个解，故：$a + a' = y - 1,b + b' = -1$

由于已知$b≥0$，故$b'<0$，即不存在所求a',b'，原命题得证。

由于$n = a \cdot x + b \cdot y \geq 0$，则最大的满足不存在非负整数$a',b'$使得 `n' = a′ · x + b′ · y` 的**$n'$为$xy - x - y$**。即对于任意整数$n≥xy - x - y + 1=(x - 1)(y - 1)$，存在非负整数a、b使得$n = a \cdot x + b \cdot y$。

根据以上结论，当$j > l + 2h2h3$时，$j - l > 2h2 \cdot h3 > (h2-1)(h3-1)$，故存在非负整数a、b使得$j - l = a \cdot h2 + b \cdot h3$。

根据**shell sort**性质(1)可知$A[l]<A[l + a \cdot h2 + b \cdot h3]=A[j]$，原命题得证。

附加题

同理(a),当$j > l + h2h3$时，$j - l > h2 \cdot h3 > (h2-1)(h3-1)$，故存在非负整数a、b使得$j - l = a \cdot h2 + b \cdot h3$。

根据**shell sort**性质(1)可知$A[l]<A[l + a \cdot h2 + b \cdot h3]=A[j]$，原命题得证。

(b)根据(a)中的结论，在调用InsertSort(h3)和InsertSort(h2)后，对于任意的j和l满足**$j > l + 2h2h3$**时，总会有$A[j]>A[l]$。

由于最外层**for**循环需执行n-h1次。

分析最内层**while**循环：

```
while i>0 and A[i]>key do
    A[i+h1]=A[i]
    i=i-h1
```

当代码中$i≤j-2h2h3$时，满足$A[i]<key$，**while**循环终止。故运行时间满足以下公式：

$$T(n) * h_1 \le (n - h_1) * 2h_2h_3$$
$$T(n) \le \frac{2nh_2h_3}{h_1} + 2h_2h_3$$

故InsertSort(h1)的时间为O(n·h2·h3/h1)。

问题4

答:

(1)由于骰子为均匀，故投掷得到的点数概率相等，为**1/k**，具体概率分布如下表所示:

点数	1	2	...	k
概率	1/k	1/k	...	1/k

根据数学期望的定义，投出点数的期望**EX=(1+2+...+k)*(1/k)=(k+1)/2**。

(2)根据题意，投掷一次骰子得到点数C的概率为**1/k**，不得到C的概率为**1-1/k**。

令投掷次数为N，则有:

$$P(N = n) = \frac{1}{k}(1 - \frac{1}{k})^{n-1} \quad (n = 1, 2, \dots)$$

显然，N服从的分布为几何分布。

根据几何分布的性质，投掷次数的期望**EN=1/(1/k)=k**。

问题5

答:

(a)A(n)函数时间复杂度为Θ(n^3)。

分析过程如下:

要得到该算法的时间复杂度，需分析最内层语句 `t=t+1` 的执行次数。

该语句的执行次数为:

$$
\begin{aligned}
T(n) &= \sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=1}^{j} 1 \\
&= \sum_{i=1}^{n} \frac{(n-i+1)(n+i)}{2} \\
&= \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}
\end{aligned}
$$

故A(n)函数时间复杂度为Θ(n^3)。

(b)B(n)函数时间复杂度为Θ(2^n)。

分析过程如下：

要得到该算法的时间复杂度，需分析最内层语句 `t=t+B(i)` 的执行次数。

$$
\begin{aligned}
&\text{显然有：} T(1) = 1 \\
&\text{且} T(n) = 1 + \textstyle\sum_{i=1}^{n-1} T(i) \\
&\text{假设} T(n) = 2^{n-1}, \text{将其带入以上时间复杂度表达式有：} \\
&T(n) = 1 + \textstyle\sum_{i=1}^{n-1} 2^{i-1} = 2^{n-1} \\
&\text{故原假设成立。}
\end{aligned}
$$

故B(n)函数时间复杂度为Θ(2^n)。