chapter 2

2022 年 12 月 3 日

目录

1 阶

时间复杂度 通常我们使用同阶函数的符号来描述一个算法的时间复杂度 viz. $\Theta(f(n))$ 就是说, 当输入规模为 n 的时候, 运行时间和 $f(n)$ 同阶. 这个算法的运行时间和 $f(n)$ 有着相同的增长率.

我们于是可以使用 Θ 来比较某些算法的时间效率.

渐进 渐进就是考虑无穷远处附近的情况. 比如说我们有一个**渐进正的函数** f 那么 $\exists c$ s.t. $x > c \implies f(x) > 0$. 输入规模很大的时候, 我们就可以只考虑最高阶的项

增长符号 这些符号有五种, 我们现在给出其严格定义.

渐进上界 O

$$O(f) = \{g : \exists c, n_0, n > n_0 \implies g(n) \le cf(n)\}$$

这个符号说明 f 是一个渐进上界, $O\left(f\right)$ 就表示那些, 在远处来看, 比 f 小的函数. 我们在描述一些算法的复杂度的时候, 也会使用这个符号, 因为一个算法的最坏情况和最好情况是不一样. 描述整个算法的时候常用 O , 也能用 Θ , 只不过要注明是一般情况还是最坏情况.

渐进紧界 $\mathit{\Theta}$:

$$\mathit{\Theta}\left(f\right)=\left\{g:\exists c_1,c_2,n>n_0\implies c_1f\left(n\right)\le g\left(n\right)\le c_2f\left(n\right)\right\}$$

这个是同阶函数的意思, 我们说在远处, f 和 g 之间的区别就是一个常数倍的关系. 我们很容易知道 $n^2+\dfrac{1}{2}n^{0.5}+n^{0.6}\in\Theta\left(n^2\right)$ 我们只用考虑这里面的最大项数, 其中的那些小项可以直接忽略不计. 证明需要从定义出发: 比如说证明 $\dfrac{1}{2}n^2-3n\in\Theta\left(n^2\right)$.

取 $c_2=\frac{1}{2}$, 然后 $\dfrac{1}{2}n^2-3n=n\left(\dfrac{1}{2}n-3\right)=n\left(\dfrac{1}{3}n+\dfrac{1}{6}n-3\right)$ 取 $n_0=9$ 那么 $n>n_0$ 有 $\dfrac{1}{3}n+\dfrac{1}{6}n-3>\dfrac{1}{6}n$

于是取 $c_1=\dfrac{1}{6}$, c_1,c_2,n_0 都取定了, 那么就证明完了.

渐进下界 Ω

$$\Omega\left(f\right)=\left\{g:\exists c,n_0,n>n_0\implies g\left(n\right)>cf\left(n\right)\right\}$$

和 O 很类似. 我们可以通俗的将 Θ 理解为 $=$, 将 Ω 视为 $\ge$, O 看作 $\le$. O 用来表示渐进上界, Θ 表示的是渐进紧界, Ω 表示的渐进下界.

多项式界限的. $f\left(n\right)=O\left(p\left(n\right)\right)$ where p is a polynomial, 那么我们称呼 f 是多项式界限的.

严格渐进上界 o: 小 o 符号, 是严格小于的意思. 实际上就是将那些同阶的函数给去掉了, 但是这种去掉的方法值得注意. 其严格的数学定义如下.

$$o\left(f\right)=\left\{g:\forall c>0,\exists n_0,\forall n>n_0\implies g\left(n\right)<f\left(n\right)\right\}$$

证明 $2n^2\ne o\left(n^2\right)$ 即我们要证明存在 c 使得 "$\forall n>n_0,2n^2<n^2$" 不成立. 注意到 $c=1>0$ 则对于任意的 n, $2n^2<cn^2$ 都不成立.
证明有点乱, 勉强看把. 同时我们有一种等价描述法:

$$f\left(n\right)=o\left(g\left(n\right)\right)\iff\lim_{n\to\infty}\frac{f\left(n\right)}{g\left(n\right)}=0$$

至少我认为这是等价的.

严格渐进下界 ω: 小 ω 符号, 是严格大于的意思, 也是将那些同阶的函数去掉了. 和上面的定义完全类似.

$$\omega\left(f\right)=\left\{g:\forall c>0,\exists n_0,\forall n>n_0\implies g\left(n\right)>cf\left(n\right)\right\}$$

类似的, 我们也有等价的描述法:

$$\lim_{n\to\infty}\frac{f\left(n\right)}{g\left(n\right)}=\infty$$

增长符号之间的关系 我们将他们考虑为那些小于等于号, 等于号, 大于等于号就行了, 并不是很难处理的. 我们著需要理解就行了.

Remark 1. *我们这里定义的这种关系实际上并不是良序的. 就是说, 存在两个函数, 他们是不可比的. 比如说*

$$n, \quad n^{1+\sin n}$$

The latter is a BAD function.

Example 1. *Is $f(n)+o(f(n))=\Theta(f(n))$ correct?*

2 和式的估计和界限

我们有几种方法, 1. 硬算; 2. 数学归纳法; 3. 放缩

2.1 1. 直接算

Proposition 1. *有限的求和能够拆开来:*

$$\sum_{k=1}^{n}(ca_k+b_k)=c\sum_{k=1}^{n}a_k+\sum_{k=1}^{n}b_k$$

这当然是显然的, 因为是有限的.

Example 2. *算术级数: $\displaystyle\sum_{k=1}^{n}k=\frac{n(n+1)}{2}=\Theta\left(n^2\right)$*

几何级数: $\displaystyle\sum_{k=0}^{n}=1+x+\cdots+x^n=\frac{x^{n+1}-1}{x-1}$

前后项抵消: $\displaystyle\sum_{k=1}^{n}(a_k-a_{k-1})=a_n-a_0$

裂项求和: $\displaystyle\sum_{k=1}^{n-1}\frac{1}{k(k+1)}=\sum_{k=1}^{n-1}\left(\frac{1}{k}-\frac{1}{k+1}\right)=1-\frac{1}{n}$

2.2 2. 数学归纳法

主要是使用第二数学归纳法, 证明一个上界而已.

Example 3. *证明 $\displaystyle\sum_{k=0}^{n}3^k=O(3^n)$*

证明. 步骤 1. $n=1$ 的时候, 我们取 c 是一个大于 1 的数就行了, 具体地, 我们取 $c=\frac{3}{2}$ 等会就能知道为什么了.

步骤 2. $n=i$ 的时候, 根据假设, $\displaystyle\sum_{k=0}^{n}3^k=O(3^n)$ 成立

步骤 3. 我们现在要证明 $n=i+1$ 的时候, 上面的式子也成立. $\sum_{k=0}^{i+1}3^k=\sum_{k=0}^{i}3^k+3^{i+1}\le c3^i+3^{i+1}$ 有

$$c3^i+3^{i+1}=c3^{i+1}\left(\frac{1}{3}+\frac{1}{c}\right)\le c3^{i+1}$$

得证. 这里用到了 $c=\frac{3}{2}$ 的性质. □

Example 4. *一个错误示范: $\sum_{k=1}^{n}k=O(n)$*

证明. 1. $n=1$ 的时候当然成立.

2. $n=m$ 的时候成立, 根据假设

3. $n=m+1$ 的时候

$$\sum_{k=1}^{m+1} k = \sum_{k=1}^{m} + (m+1) \leq cm + (m+1) = (c+1)\,m + 1 = O\,(m)$$ □

这个证明是错误的, 因为其我们考虑这个 $n=m+1$ 的时候, 因为我们已经给定了这个 c , 那么需要维持这个 c . 就是说, 我们要得到 cn *viz.* $c\cdot(m+1)$ 而不是上面这个 $(c+1)\cdot m+1$

我们需要有不等式:

$$cm + m + 1 \leq c\,(m+1)$$

就有

$$c \geq m+1$$

这个时候能够看出 c 是一个随着 n 变化而变化的函数值 (并且这个函数没有上界), 并不是一个常数.

2.3 放缩法

Example 5.

$$\sum_{k=1}^{n} k \leq \sum_{k=1}^{n} n = n^2$$

Example 6.

$$\sum_{k=1}^{n} a_k \leq n \times \max\{a_k\}$$

Example 7 (使用等比数列作为上界)**.** *If* $\forall k>0, a_0 \geq 0, 0 \leq \dfrac{a_{k+1}}{a_k} \leq r < 1$, 求 $\sum_{k=0}^{n} a_k$ 的上界

证明. 明显这个数列的每一项, 都是小于一个 r 为公比, a_0 为首项的等比数列.

$$a_k \leq a_0 r^k$$

那么 $\sum\limits_{k=0}^{n} a_k \leq \sum\limits_{k=0}^{\infty} a_0 r^k = a_0 \sum\limits_{k=0}^{\infty} r^k = a_0 \dfrac{1}{1-r}$ $(|r| \leq 1)$ □

Example 8. 求 $\sum\limits_{k=1}^{\infty}\left(\dfrac{k}{3^k}\right)$ 的上界.

证明. 能够注意到

$$a_{k+1}/a_k = \frac{(k+1)\,/3^{k+1}}{k/3^k} = \frac{1}{3} \times \frac{k+1}{k} \leq \frac{1}{3} \times 2 = \frac{2}{3}$$

我们设 r 为 $\dfrac{2}{3}$ 就能够使用上面的方法 □

2.3.1 裂项求和

裂开!

Example 9. 求 $\sum_{k=1}^{n} k$ 的下界

证明.

$$\sum_{k=1}^{n} k = \sum_{k=1}^{\lceil n/2 \rceil} k + \sum_{k=\lceil n/2 \rceil}^{n} k$$

我们将这个东西放小. $\sum\limits_{k=1}^{\lceil n/2 \rceil} k \geq 0$, 并且 $\sum\limits_{k=\lceil n/2 \rceil+1}^{n} k \geq \sum\limits_{k=\lceil n/2 \rceil+1}^{n} \frac{n}{2}$

就能够得到

$$\sum_{k=1}^{n} k \geq \sum_{\lceil n/2 \rceil+1}^{n} \frac{n}{2} \geq \left(\frac{n}{2}\right)^2 = \frac{n^2}{4}$$

这就说明这个逼登东西的下界是 $\Omega\left(n^2\right)$ □

Example 10. $\sum_{k=0}^{\infty} \frac{k^2}{2^k}$ 的上界

证明. 对于类似这样的东西, 我们可以联想到放缩为等比数列的那个东西.

注意到 $k \geq 3$ 的时候

$$\frac{(k+1)^2 / 2^{k+1}}{k^2 / 2^k} = \frac{(k+1)^2}{2k^2} \leq \frac{8}{9}$$

于是我们将这个东西拆分为两个部分, 一个部分是前面 $\sum_{k=0}^{2} \dfrac{k^2}{2^k}$ 以及从 $k=3$ 开始的后面的那些东西.

那么有

$$\begin{aligned}
\sum_{k=0}^{\infty} \frac{k^2}{2^k} =& \sum_{k=0}^{2} \frac{k^2}{2^k} + \sum_{k=3}^{\infty} \frac{k^2}{2^k} \\
\leq& 0 + \frac{1}{2} + 1 + \sum_{k=3}^{\infty} \frac{9}{8} \left(\frac{8}{9}\right)^k \\
\leq& \frac{3}{2} + \left(\frac{8}{9}\right)^k \sum_{k=0} \left(\frac{8}{9}\right)^k \\
=& \frac{3}{2} + \left(\frac{8}{9}\right)^2 \cdot \frac{1}{1-\frac{8}{9}} \\
=& \text{不知道多少} = O(1)
\end{aligned}$$

□

Example 11. 求调和级数 $H_n = \sum_{k=1}^{n} \frac{1}{k}$ 的上界

证明. Not so good the proof.

$$\begin{aligned}
\sum_{k=1}^{n} \frac{1}{k} = \frac{1}{1} + \left(\frac{1}{2} + \frac{1}{3}\right) + (\frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7}) \\
+ \left(\frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \frac{1}{11} + \frac{1}{12} + \frac{1}{13} + \frac{1}{14} + \frac{1}{15}\right) \\
+ \cdots
\end{aligned}$$

我们进行一个观察, 这个括号[1]里面的几把东西, 可以看出 $\left(\dfrac{1}{2} + \dfrac{1}{3}\right) \leq \left(\dfrac{1}{2} + \dfrac{1}{2}\right) = 1$ 每一个括号都是类似的. 所以说

$$\sum_{k=1}^{n} \frac{1}{k} \leq \sum_{i=0}^{\lceil \log n \rceil} 1 \leq \log n + 1 = O(\log n)$$

[1]你是否记得什么情况下括号是合法的?

$\square$

2.3.2 积分近似法

其实就是画长方形.

我们有近似方法: 如果说 f 是单调的, 那么有

$$\int_{m-1}^{n} f(x)\ \mathrm{d}x \leq \sum_{k=m}^{n} f(k)$$
$$\int_{m}^{n+1} f(x)\ \mathrm{d}x \geq \sum_{k=m}^{n} (k)$$

这个好像叫什么 darboux 上和, darboux 下和什么的. 没有什么必要进行死记, 理解就行

我个人建议画一个图, 反正我画不了.

Example 12. *如果 f 是单调递减的, 那么*

$$\int_{m}^{n+1} f(x)\ \mathrm{d}x \leq \sum_{k=m}^{n} f(k) \leq \int_{m-1}^{n} f(x)\ \mathrm{d}x$$

证明. 略 $\square$

Example 13.

$$\log(n+1) = \log x|_1^{n+1} = \int_1^{n+1} \frac{1}{x}\ \mathrm{d}x \leq \sum_{k=1}^{n} \frac{1}{k}$$

while

$$\sum_{k=2}^{n} \frac{1}{k} \leq \int_1^{n} \frac{1}{x}\ \mathrm{d}x = \log x|_1^{n} = \log n$$

Example 14. *$H_n = o(n)$ 是对的吗?*

3 递归方程

Example 15. *比如说这时一个递归方程:*

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + 11 \cdot n & n > 1 \\ 1 & n \leq 1 \end{cases}$$

这个东西就描述了一种递归关系. 对于这个 $T(n)$ 我们希望找到这个东西的阶.

上述这个递归方程的就有 $T(n) = O(n\log n)$

一般来说, 对于递归方程的初始条件, $n \leq 1$ 的时候的值, 都是给定的. 不是一般性, 我们都假定这个东西是 $O(1)$. 因为就算说, 阿, 这个东西, 哈, 初始的时候就有 $T(1) = O(n)$ 什么的, 其实没什么用, 我只需要照常求, 然后在结果上面加上一个 $O(n)$ 就行了.

我们求解递归方程有三个主要的方法

$$
\begin{cases}
1.\text{替换方法} \\
2.\text{递归树方法} \\
3.\text{Master 定理}
\end{cases}
$$

图 1: 三个主要方法

进行一点说明: 1. 其实就是猜, 然后使用数学归纳法. 2. 画出递归树, 然后将方程转化为一个 series 然后使用估计的方法机型求解. 3. 可以求解形如 $T(n) = a \cdot T(n/b) + f(n)$

3.1 替换方法

Example 16. *求解* $T(n) = 2 \cdot T(\lfloor n/2 \rfloor) + n, T(1) = 1$ *的上界.*

证明. 根据这个经验 (master 定理也行) 猜测这个结尾 $T(n) = O(n \log n)$.

根据这个 O 的定义, 我们需要证明

$$\exists c, n_0 > 0, \forall n \geq n_0, T(n) \leq cn \log n$$

而后, 使用归纳法证明:

step 1. $n = 1$ 的时候显然成立, 随后

step 2. 设 $n \leq m$ 的时候这个结论都成立.

step 3. 验证 $n = m + 1$ 的时候

$$
\begin{aligned}
T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n \leq \left(c\frac{n}{2} \log \frac{n}{2}\right) + n \leq cn \log \frac{n}{2} + n \\
= cn \log n - cn \log 2 + n \\
= cn \log n - n(c \log 2 - 1)
\end{aligned}
$$

这个时候, 如果说 $c \log 2 - 1$ 是大于 0 的, 就有

$$T(n) \leq cn \log n$$

得证 □

Comment 1. *如果说初始条件不成立的的话, 就往后推理, 看看是否是矛盾得.*

比如说这里就有: $n \log n|_{n=1} = \log 1 = 0$ *这个和* $T(n) = 1$ *矛盾了, 于是我们找到* $n = 2$ *得时候*

对于大多数得递归式而言, 扩展边界条件使得归纳假设对较小的 n *成立, 是一种简单直接得方法*

Example 17. *求解* $T(n) 2 \cdot T\left(\frac{n}{2} + 17\right) + n$

证明. 设 $T(n)$ 和 $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$ 只相差了一个常数 17

啥, 为什么?

当 n 充分大的时候, $T\left(\frac{n}{2} + 17\right)$ 和 $T(n/2)$ 之间得差别并不是很大. 这个时候我们可以猜测, $T(n) = O(n \log n)$. 猜测完了之后就是使用数学归纳法的时候了

step 1. 初始显然成立

step 2. 设 $n \leq m$ 得时候成立, viz.

$$T(n) \leq cn \log n$$

step 3. 下面验证 $n = m + 1$ 得时候的情况, 并且这里有 $n/2 + 17 \leq m$. 这里设 $\log = \log_2$

$$\begin{aligned}
T(n) &= 2T(n/2+17) + n \leq 2c\left(\frac{n}{2}+17\right) \cdot \log\left(\frac{n}{2}+17\right) + n \\
&= (cn + 34c) \cdot (\log(n+34) - 1) + n \\
&\leq (cn + 34c) \cdot (\log(1.5n) - 1) + n && (n + 34 < 1.5) \\
&= (cn + 34c) \cdot (\log n + \log 1.5 - 1) + n \\
&= cn \log n + ((\log 1.5 - 1)c + 1)n + 34c(\log 1.5 - 1) + 34c \log n \\
&\leq cn \log n && ((\log 1.5 - 1)c + 1 \leq 0, \log 1.5 - 1 < 0)
\end{aligned}$$

shit！$c \log n$ is literally ignored. □

3.2 替换方法 2

我们证明比较松的上下界, 然后缩小范围.

Example 18. 求解 $T(n) = 2 \cdot T(n/2) + n$

证明. 先证明 $T(n) = \Omega(n), T(n) = O\left(n^2\right)$ 然后降低上界, 提高上界.

$\Omega(n)$ 上一阶是 $\Omega(n \log n)$ 而 $O\left(n^2\right)$ 的下一个阶是 $O(n \log n)$ □

但是, 就算说这个猜测是正确的, 也有可能不适合使用归纳法. 于是说我们需要从猜测中减去一个低价的玩意, 就可能解决.

Example 19 (减取低阶的玩意)**.** $T(n) = T\left(\lfloor \frac{n}{2} \rfloor\right) + T(\lceil n/2 \rceil) + 1$

证明. 猜测 $T(n) = O(n)$

有

$$T(n) \leq c\lfloor \frac{n}{2} \rfloor + \lceil \frac{n}{2} \rceil + 1 = cn + 1 \neq cn$$

就是因为这个 1 , 现在不能证明出来.

这个时候, 我们需要从猜测中减去一个低价的玩意, 那么 $T(n) \leq cn - b$, 其中 $b \geq 0$ b 是随便一个数字, 就有:

$$\begin{aligned}
T(n) &\leq c\lfloor \frac{n}{2} \rfloor + \lceil \frac{n}{2} \rceil + 1 - 2b \\
&= cn - 2b + 1 = cn - b - b + 1 \leq cn - b
\end{aligned}$$

其中我们令 $b \geq 1$, 就能够得到答案了 viz. $T(n) \leq cn - b$ □

Comment 2. *为什么说, 是增加一个呢? 并且为什么说上面是 $cm - b$ 的形式呢? 也就是说这个减去一个项并不是说 $T(n) = O(n - b)$?*

3.3 变量替换法

使用变量替换把递归方程变换为熟悉的方程.

Example 20. $T(n) = 2 \cdot T(\sqrt{n}) + \log n$ 考虑 $\sqrt{n}$ 为整数.

证明. 令 $m = \log n$ 那么 $n = 2^m$ 然后, 就有 $T(n) = T(2^m)$ 将这个玩意视为 m 的函数就有

$$T(2^m) = S(m)$$

并且还有

$$S(m) = T\left(2^{m/2}\right) + m = 2S\left(\frac{m}{2}\right) + m$$

明显这个逼登东西是我们熟悉的, 然后就有 $S(m) = O(m \log m)$, $T(2^m) = O(m \log m)$

那么就有 $T(n) = O(\log n \log(\log n))$ □

Very good the example

3.4 递归数方法

下面是一个步骤.

$$\begin{cases} \text{1.画出递归树} \\ \text{2.循环的展开方程} \\ \text{3.吧递归方程转化为 series} \\ \text{4.求解 series} \end{cases}$$

根节点就是 $T(n)$.

内部的节点就是不同层次调用的那些东西产生的代价.

并且这个树的分支的数量取决于这个子问题的数量. 其中叶子节点是边界条件的那些东西.

Example 21. $T(n) = 3 \cdot T(n/4) + \Theta\left(n^2\right)$ 能够看出这是一个 *3 branching* 的一棵树.

我们进行一个手算好吧.

先看 *level 0*, 明显有 cn^2 , 这个是根据 $\Theta\left(n^2\right)$ 来的, 就一个节点, *viz.* 根节点.

level 1. 有三个节点, 都是 $c(n/4)^2$. 总和就是 $\frac{3}{16}n^2$

level 2. 类似的, 有 9 个节点, 每一个都是 $c\left(n/4^2\right)^2$, 总和就是 $\left(\frac{3}{16}\right)^2 cn^2$

每一个 *level* 都是差不多的, 于是我们要知道 *level* 的数目. 其实就是 $\log_4 n$, 这时明显的. 那么就有:

$$T(n) = \sum_{k=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^k cn^2$$

这里其实还需要特别注意那个, 叶子节点的那一层, 反正就是那层不太一样, 我们找出那层的节点个数: $3^{\log_4 n} = n^{\log_4 3}$, 那么那一层的代价就是 $cn^{2 \log_4 3}$

就有:

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log_4 n-1}\left(\frac{3}{14}\right)^i cn^2+\Theta(n\log_4 3) \\ &= \frac{1-\left(\frac{3}{16}\right)^{\log_4 n}}{1-\frac{3}{16}}cn^2+\Theta\left(n^{\log_4 3}\right) \end{aligned}$$

the deduction above use $\sum_{k=0}^{n} x^k = \frac{1-x^{n+1}}{1-x}$

Example 22. $T(n) = n + 3T(n/4)$

证明. 容易知道

$$\begin{aligned} T(n) &\le \sum_{i=0}^{\log_4 n-1}\left(\frac{3}{4}\right)^i n+\Theta\left(n^{\log_4 3}\right) \\ &\le n\sum_{i=0}^{\infty}\left(\frac{3}{4}\right)^i \Theta\left(n^{\log_4 3}\right) \\ &= n\times\frac{1}{1-\frac{3}{4}}+\Theta\left(n^{\log_4 3}\right) = 4n+\Theta\left(n^{\log_4 3}\right) = O(n) \end{aligned}$$

□

3.5 Master theorem

Master theorem is used to solve the recurrence function with the form

$$T(n) = aT(n/b) + f(n)$$

where $a \ge 1$, $b > 1$, they are 常数. $f(n)$ 渐进正函数

Theorem 1. *Give a recurrence function* $T(n)$ *then*

1. *if* $f(n) = O\left(n^{\log_b a-\varepsilon}\right)$ *for some* $\varepsilon > 0$ *, then*

$$T(n) = \Theta\left(n^{\log_b a}\right)$$

2. *if* $f(n) = \Theta\left(n^{\log_b a}\right)$ *, then*

$$T(n) = \Theta\left(n^{\log_a b}\log n\right) = \Theta(f(n)\log n)$$

3. *if* $f(n) = \Omega\left(n^{\log_b a+\varepsilon}\right)$ *for some* $\varepsilon > 0$*, then*

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

with additional condition $\exists c < 1$ *we have* $af(n/b) \le cf(n)$ *asymptotically*

You may notice the condition $f(n) = \Omega\left(n^{\log_a b+\varepsilon}\right)$ somehow weird. Why bother to use ε ? In fact , there are some conditions that Master can not tackle with viz. f suit that

$$\forall\varepsilon > 0, f(n) = o(p(n)\cdot n^{\varepsilon})$$

while that $f = \omega(p(n))$

Example 23. $T(n) = 9T(n/3) + n$

$T(n) = T(2n/3) + 1$

Example 24. $T(n) = 3T(n/4) + n\log n$

Example 25. $T(n) = 2T(n/2) + n\log n$