§2.2 递归

*递归可能分解为规模不等的子问题
求解递归方程的方法：代入法（猜测上界后证明）、递归树法（转化乘树，节点表示不同层次产生的代价，再采用边界求和）、主方法（求解 $T(n)=aT(n/b)+f(n)$，$a\geq 1, b>1$，$f(n)$ 是某给定函数（并非对任何都可解））
**技术细节：

1. 假定自变量整数，忽略上下取整。
2. 对足够小的 n 假设 $T(n)$ 为常数，忽略边界。

   （一些特殊情况可能导致技术细节非常重要，上面两种条件仍然值得重视）
3. 有时会存在不等式情况，如 $T(n)\leq 2T(n/2)+O(n)$，此时一般用 O 描述上界，反之对大于等于可用 Ω 描述下界。

例：最大子数组问题（给定数组，求和最大的连续子数组）
分治策略：找到数组中央位置，任何连续子数组必然在其左侧、右侧，或包含它。左侧与右侧可直接通过递归，由此只需要找到包含中间位置的最大子数组后取最大值即可。

```
def find_max_crossing_subarray(A,low,mid,high):
  left_sum = -infty
  sum = 0
  for i = mid downto low
    sum += A[i]
    if (sum > left_sum)
      left_sum = sum
      max_left = i
  right_sum = -infty
  sum = 0
  for i = mid -> low
    sum += A[i]
    if (sum > right_sum)
      right_sum = sum
      max_right = i
  return (max_left, max_right, left_sum+right_sum)
```

整体算法：利用上方的算法进行递归，low=high 即为终止条件。

复杂度分析：包含中间位置的部分的复杂度为 $\Theta(n)$，递归方程为 $T(n)=\begin{cases}\Theta(1) & n=1\\ 2T(n/2)+\Theta(n) & otherwise.\end{cases}$，

可发现复杂度与归并排序相同，为 $\Theta(n\log n)$。

*算法改进（$\Theta(n)$ 算法）：从左侧开始，找到第一个大于 0 的位置 i1 开始，依次求和（sum+=A[j]），max1 记录目前的最大值，并记录当前的 j1。直到 sum<0 时中止，然后继续向右找到下一个大于 0 的位置 i2，清空 sum，重复此过程，在比较中得到 maxk 中的最大值即可（证明思路：反证，若否则可以拼接为更大）。

```
def find_max_subarray(A, low, high):
  now = low
  summing = 0
  max = -infty
  for now = low to high
    if (summing == 1)
      sum += A[now]
      if (sum > max_now)
        j_now = now
        max_now = sum
      if (sum <= 0 or now == high)
        if (max_now > max)
          i_max = i_now
          j_max = j_now
          max = max_now
        summing = 0
    else if (A[now] > 0)
      summing = 1
      max_now = sum = A[now]
      i_now = j_now = i
  return (max,i_max,j_max)
```

§2.3 替代法

包含两个步骤：猜测解的形式、归纳常数（直接替代）并证明解正确（要求易于猜得）

例：针对 $T(n)=2T(\lfloor n/2\rfloor)+n$，先猜测解为 $T(n)=O(n\log n)$，选取常数 $C>T(2)+1$ 即可。

更复杂的例子：求 $T(n)=2T(\lfloor n/2\rfloor+17)+n$ 的解的上界。

解法（平移的思路）：令 $n=m+34$，可发现 $T(m+34)=2T(\lfloor m/2\rfloor+34)+m+34$，由此 $T(m+34)+34=2(T(\lfloor m/2\rfloor+34)+34)+m$ 类似上一种情况可直接估算出上界。

*猜测出渐近界未必能归纳成功，有时是因为归纳假设偏弱，可以尝试加强假设、调整低阶项、初等变换等，如 $T(n)=2T(\lfloor n/2\rfloor)+1$，归纳假设 cn 无法继续，假设为 $cn-2$ 即可。

*不应将猜测无理由放大

*变量代换：对 $T(n)=2T(\lfloor\sqrt{n}\rfloor)+\log n$，取 $m=\log n$ 可发现最终结果为 $O(\log n\log\log n)$。

**弊端：猜测、证明都可能困难

§2.4 递归树

每个节点代表相应子问题代价，行和代表某层代价，总和得总代价

*一般用来获得猜测解，省略大部分细节。也可细化直接解出结果。

例：$T(n)=3T(\lfloor n/4\rfloor)+\Theta(n^2)\Longrightarrow T(n)=3T(n/4)+cn^2$ 简化，接着画出递归树求得复杂度大约为 $\sum_{i=0}^{\infty}\frac{3^i}{16^i}cn^2=c'n^2$，因此为 $\Theta(n^2)$ 量级。

代价不相同：$T(n) = T(n/3) + T(2n/3) + O(n)$，作出递归树可发现每层的和为 cn，再由行数（通过解方程可计算出层数具体值，不过估算中没有精确必要）为 $O(\log n)$ 可知复杂度 $O(n \log n)$。同理，对于 $T(n) = T(n/4) + T(n/2) + n^2$，可类似算得结果为等比级数的 n^2 倍，仍为 n^2 量级。
*关键为求行和与总代价，需要上下行和的关联