解决若干“小”问题比解决一个“大”问题easier

4.3 分治策略

- 对于一个规模为*n*的问题，若该问题可以容易地解决（比如说规模*n*较小）则直接解决
- 否则将其分解为*k*个规模较小的子问题，这些子**问题互相独立**且与**原问题形式相同**，递归地解这些子问题，然后将各子问题的解**合并**得到原问题的解。

4.3 分治策略

- 如果原问题可分割成k个子问题，$1<k\leqslant n$，且这些子问题都可解，并可利用这些子问题的解求出原问题的解，那么这种分治法就是可行的。
- 由分治法产生的子问题往往是原问题的较小模式，这就为使用递归技术提供了方便。在这种情况下，反复应用分治手段，可以使子问题与原问题类型一致而其规模却不断缩小，最终使子问题缩小到很容易直接求出其解。这自然导致递归过程的产生。
- 分治与递归像一对孪生兄弟，经常同时应用在算法设计之中，并由此产生许多高效算法。

4.3 分治策略

- 用分治法求解的条件：

1. 小规模时，问题容易求解
2. 问题可以分解成若干个规模较小的相同问题。子问题的解可以合并为该问题的解
3. 子问题相互独立，即不包含公共的子问题

4.3 分治策略

- 三个步骤：
- 划分（divide）:将原问题分解成若干规模较小、相互独立、与原问题形式相同的子问题。
- 解决（conque）：若子问题规模较小，则直接求解；否则递归求解个子问题。
- 合并（combine）：将各子问题的解合并为原问题的解。

分治策略一般流程

Solve(I)

{ n=size(I);

if (n<=SmallSize) Solution=directlySolve(I);

else Divide I into $I_1, I_2, \ldots, I_k$;

for each i $\in$ {1,2,…,k} S_i=solve(I_i);

Solution= Combine(S_1, S_2,…, S_k);

return Solution;}

$$T(n) = \begin{cases} B(n), & n \le SmallSize \\ D(n) + \sum_{i=1}^{k} T(size(I_i)) + C(n), & n > SmallSize \end{cases}$$

4.3分治策略

1. 快速排序--***hard*** division, ***easy*** combination
2. 归并排序--***easy*** division, ***hard*** combination

4.4快速排序

E[0]	E[1]	E[2]	E[3]	E[4]	E[5]	E[6]	E[7]
	49	38	65	67	76	13	50

4.4快速排序

E[0]	E[1]	E[2]	E[3]	E[4]	E[5]	E[6]	E[7]
49	49	38	65	67	76	13	50

first *last*

4.4快速排序

E[0]	E[1]	E[2]	E[3]	E[4]	E[5]	E[6]	E[7]
49	13	38	49	67	76	65	50

↑ ↑

first last

n个数据元素的排序经过$n-1$次比较，**划分**为2个与**原问题形式相同**的排序**子**问题

一次比较可能不止消除一对逆序

对每个子问题采用同样的策略求解

4.4快速排序

E[0]	E[1]	E[2]	E[3]	E[4]	E[5]	E[6]	E[7]
49	13	38	49	67	76	65	50

fitasty

4.4快速排序

E[0]	E[1]	E[2]	E[3]	E[4]	E[5]	E[6]	E[7]
13	13	38	49	67	76	65	50
	↑ *first*	↑ *last*					

4.4快速排序

E[0]	E[1]	E[2]	E[3]	E[4]	E[5]	E[6]	E[7]
13	13	38	49	67	76	65	50

↑ ↑

first *last*

4.4快速排序

E[0]	E[1]	E[2]	E[3]	E[4]	E[5]	E[6]	E[7]
67	13	38	49	67	76	65	50
				first			*last*

4.4快速排序

E[0]	E[1]	E[2]	E[3]	E[4]	E[5]	E[6]	E[7]
67	13	38	49	50	65	67	76

↑↑

first *last*

4.4快速排序

E[0]	E[1]	E[2]	E[3]	E[4]	E[5]	E[6]	E[7]
50	13	38	49	50	65	67	76
				↑ *first*	↑ *last*		

4.4快速排序

E[0]	E[1]	E[2]	E[3]	E[4]	E[5]	E[6]	E[7]
50	13	38	49	50	65	67	76

first *last*

合并没有做任何工作

4.4快速排序

```
void QSort (Element [] E, int first, int  last)
{    int t;
     if (first<last)
     {
          t = partition( E, first, last);
          QSort (E,first,  t-1);
          QSort (E, t+1, last);
     }
}
```

```
Solve(I)
{   n=size(I);
   if (n<=SmallSize)
    Solution=directlySolve(I);
   else  Divide I into I1,I2,…,Ik;
         for each i ∈{1,2,…,k} Si=solve(Ii);
         Solution=  Combine(S1, S2,…, Sk);
   return Solution;}
```

4.4快速排序

```
int partition(Element [] E, int first,  int last )
{   E[0]=E[first];
    while(first<last)
    {
      while(( first < last) && (E[last] >E[0]) )  last--;
      if (first < last ) {E[first]=E[last]; first++;}
      while( (first < last) && (E[first]<=E[0]))  first++;
      if(first < last) {E[last]=E[first]; last--;}
    }
    E[first]=E[0]; return first;
}
```

4.4快速排序

- w(n)
- A(n)
- pivot的选取对算法性能的影响

4.4快速排序----w(n)

$$T(n)=\begin{cases} B(n), & n \le SmallSize \\ D(n)+\sum_{i=1}^{k} T(size(I_i))+C(n), & n > SmallSize \end{cases}$$

$$T(n)=\begin{cases} 0, & n \le 1 \\ (n-1)+T(I_1)+T(I_2)+0, & n > 1 \end{cases}$$

存在一种情况，划分的2个子问题规模为0和n-1此时:

$$T(n)=\begin{cases} 0, & n \le 1 \\ (n-1)+T(0)+T(n-1)+0, & n > 1 \end{cases}$$

递推求解该式得: $T(n)=n(n-1)/2$

4.4快速排序---- A(*n*)

- assuming all input permutation are equally likely

$$T(n)=\begin{cases}0, & n\le 1\\ (n-1)+T(I_1)+T(I_2)+0, & n>1\end{cases}$$

根据假设，I_1, I_2的所有可能情况：

I_1=0, I_2=n-1;

I_1=1, I_2=n-2;

…

I_1=n-1, I_2=0;

$$A(n)=(n-1)+\frac{1}{n}\sum_{i=1}^{n}(A(i-1)+A(n-i)),n\ge 2$$

4.4快速排序---- A(*n*)

- assuming all input permutation are equally likely

A(1)=0, A(0)=0

$$A(n) = (n-1) + \frac{1}{n}\sum_{i=1}^{n}(A(i-1) + A(n-i)), n \geq 2$$

$$A(n) = (n-1) + \frac{1}{n}(A(0) + A(n-1) + A(1) + A(n-2) + \ldots\ldots + A(n-1) + A(0)), n \geq 2$$

$$A(n) = (n-1) + \frac{2}{n}(A(0) + A(1) + A(2) + \ldots\ldots + A(n-1)), n \geq 2$$

$$A(n) = (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} A(i), n \geq 2$$

4.4快速排序---- A(*n*)

- 求解方法：

1. 给出（***guess!***）A(n)的“估计值”，再进行证明
2. 直接推理计算

4.4快速排序---- A(n)---- *guess!*

- 估算快速排序的时间复杂度$Q(n) \approx n+2Q(n/2)$
- Theorem 3.17 $Q(n) \in \Theta(n\log n)$

Theorem 3.17 (Master Theorem) With the terminology of the preceding discussion, the solution of the recurrence equation

$$T(n) = b\,T\left(\frac{n}{c}\right) + f(n) \tag{3.9}$$

(restated from Equations 3.3 and 3.8) has forms of solution as follows, where $E = \lg(b)/\lg(c)$ is the *critical exponent* defined in Definition 3.6.

1. If $f(n) \in O(n^{E-\epsilon})$ for any positive ϵ, then $T(n) \in \Theta(n^E)$, which is proportional to the number of leaves in the recursion tree.
2. If $f(n) \in \Theta(n^E)$, then $T(n) \in \Theta(f(n)\ \log(n))$, as all node depths contribute about equally.
3. If $f(n) \in \Omega(n^{E+\epsilon})$ for any positive ϵ, and $f(n) \in O(n^{E+\delta})$ for some $\delta \geq \epsilon$, then $T(n) \in \Theta(f(n))$, which is proportional to the nonrecursive cost at the root of the recursion tree.

(Possibly none of these cases applies.)

4.4快速排序---- A(n)---- *guess!*

- Theorem 4.2： Let $A(n)=(n-1)+\frac{2}{n}\sum_{i=1}^{n-1}A(i), n\geq 1$

  then for n≥1, $A(n)\leq cn\ln n$ for some constant c.

证明A(n) ≤cnlnn

- 数学归纳法，n
- Base case:n=1, A(1)=0,c×1 ×ln(1)=0
- n>1,假设A(i)≤c ×i ×ln(i), 0<i<n

$$A(n)=(n-1)+\frac{2}{n}\sum_{i=1}^{n-1}A(i)\le(n-1)+\frac{2}{n}\sum_{i=1}^{n-1}ci\ln(i)$$

Equation 1.16: $$\sum_{i=1}^{n-1}ci\ln(i)\le c\int_1^n x\ln(x)dx$$

Equation 1.15: $$\int_1^n x\ln(x)dx=\frac{1}{2}n^2\ln(n)-\frac{1}{4}n^2$$

$$A(n)\le(n-1)+\frac{2c}{n}(\frac{1}{2}n^2\ln(n)-\frac{1}{4}n^2)=cn\ln(n)+n(1-\frac{c}{2})-1$$

证明A(*n*) ≤cnlnn

$$A(n) \le (n-1) + \frac{2c}{n}\left(\frac{1}{2}n^2\ln(n) - \frac{1}{4}n^2\right) = cn\ln(n) + n\left(1 - \frac{c}{2}\right) - 1$$

c≥2， **A(*n*)** ≤cnlnn

c=2, A(*n*) ≤2nlnn □

同样可证， **A(*n*)** >cnlnn

Note： ln n≈0.693logn

推论： On average, assuming all input permutation are equally likely, the number of comparisons done by Quicksort one sets of size n is approximately 1.386nlogn, for large n.

➢直接推理计算A(n)

$$A(n)=(n-1)+\frac{2}{n}\sum_{i=1}^{n-1}A(i),n\geq 2\cdots\cdots\cdots\cdots(1)$$

$$A(n-1)=(n-2)+\frac{2}{n-1}\sum_{i=1}^{n-2}A(i),n\geq 2\cdots\cdots\cdots\cdots(2)$$

(1)*n-(2)*(n-1)可得:

$$nA(n)-(n-1)A(n-1)=n(n-1)+2\sum_{i=1}^{n-1}A(i)-(n-1)(n-2)-2\sum_{i=1}^{n-2}A(i),\cdots\cdots\cdots\cdots(3)$$

$$nA(n)-(n-1)A(n-1)=2(n-1)+2A(n-1),\cdots\cdots\cdots\cdots(4)$$

$$nA(n)=(n+1)A(n-1)+2(n-1),\cdots\cdots\cdots\cdots(5)$$

$$\frac{A(n)}{n+1}=\frac{A(n-1)}{n}+\frac{2(n-1)}{n(n+1)},\cdots\cdots\cdots\cdots(6)$$

$$Let\quad B(n)=\frac{A(n)}{n+1},\quad B(1)=0$$

$$B(n)=B(n-1)+\frac{2(n-1)}{n(n+1)},\cdots\cdots\cdots\cdots(7)$$

➢直接推理计算A(n)

$$B(n) = B(n-1) + \frac{2(n-1)}{n(n+1)}, \cdots\cdots\cdots\cdots (7)$$

$$B(n) = B(n-1) + \frac{2(n-1)}{n(n+1)} = B(n-2) + \frac{2(n-2)}{(n-1)n} + \frac{2(n-1)}{n(n+1)} = B(n-3) + \frac{2(n-3)}{(n-2)(n-1)} + \frac{2(n-2)}{(n-1)n} + \frac{2(n-1)}{n(n+1)}$$

$$B(n) = \sum_{i=1}^{n} \frac{2(i-1)}{i(i+1)}$$

$$B(n) = 2\sum_{i=1}^{n} \frac{1}{i} - 4\sum_{i=1}^{n} \frac{1}{i(i+1)}$$

公式1.11: $$\sum_{i=1}^{n} \frac{1}{i} \approx \ln(n) + 0.577$$

$$B(n) \approx 2(\ln(n) + 0.577) - \frac{4n}{n+1}$$

$$A(n) = (n+1)B(n) \approx 2(n+1)(\ln(n) + 0.577) - 4n$$

$$A(n) \approx 2n\ln(n) + 2\ln(n) + 1.154 - 2.846n$$

$$A(n) \approx 1.386n\log(n) + 1.386\log(n) + 1.154 - 2.846n$$

$$A(n) \approx 1.386n\log(n) - 2.846n$$

改进措施

- 递归
- 小排序问题
- 枢轴pivot的选取----随机打乱
  - 随机选择
  - E[first],E[last],E[(first+last)/2] , (E[first]+E[last])/2
  - E[first],E[last],E[(first+last)/2]三者取中