第一章

分 治 算 法

在计算机科学中，分治算法是一种很重要的算法。字面上的解释是“分而治之”，就是把一个复杂的问题分成两个或更多的相同或相似的子问题，再把子问题分成更小的子问题……直到最后子问题可以简单地直接求解，原问题的解即子问题的解的合并。这个技巧是很多高效算法的基础，如排序算法(快速排序，归并排序)，傅立叶变换(快速傅立叶变换)……

折半查找法

【题目描述】 折半查找法(half.cpp/c/pas)

大魔导师培根曾经说过：“读史使人明智，读诗使人聪慧，演算使人精密，哲理使人深刻，伦理学使人有修养，逻辑修辞使人善辩。”由此可见书籍的重要性是不言而喻的。而与书籍天天打交道的图书管理员，更是夺天地之造化，吸日月之精华的“神之职业”。据史料记载，魔法世界从古至今诞生的众多不平凡的人物中，有不少人都曾经做过“图书管理员”，如道家学派创始人老子，威软公司创始人比耳、少林藏经阁的扫地神僧等等。所以，作为以马虎自负出名的楚继光，在魔法学院的社会实践活动中又怎么会放过这“天将降大任于斯人也”的必经锻炼呢。但想成为一个合格的图书管理员并不容易，他必须能够在一排(10 000以内)已按编号大小排好序的图书中，快速地按编号查找到某本书所在的位置。

【输入格式】

输入文件第一行是N，表示有N个元素，第二行是N个数，第三行是M表示要查找的数。

【输出格式】

一个数，即如找到该数，则输出位置，否则输出－1。

【输入样例】

3

2 4 6

4

【输出样例】

2

递归二分算法★

将已排好序的数列依次存入数组 a[ ],设查找数值为 X,用指针 bot 指向数列最左端位置(最小值),指针 top 指向数列最右端位置(最大值),取 bot 和 top 的中间值 mid 指向数列中间,如图 1.1 所示。

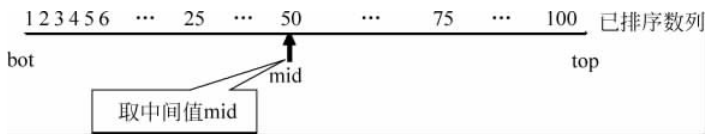

图 1.1

当 top>bot 时,比较查找 X 与 a[mid],有 3 种可能:

(1) 若 X=a[mid],则表示找到,退出比较查找;

(2) 若 X<a[mid],则选择前半段继续比较查找,bot 不变,top 变成 mid−1;

(3) 若 X>a[mid],则选择后半段继续查找,bot 变成 mid+1,top 不变。

结束的过程有两种:一种是找到了 X=a[mid];另一种是没找到,即 top<bot。

完整的参考代码如下所示:

```
1  //折半查找法——递归二分算法
2  #include <iostream>
3  #include <stdio.h>
4  #include <stdlib.h>
5  #define Max 10001
6  using namespace std;
7  int a[Max],key;
8
9  int search(int bot,int top)
10 {
11   int mid;
12   if(top>=bot)
13   {
14     mid=(top+bot)/2;                    //取中间值 mid
15     if(key==a[mid])                     //如果相等,则打印该数
16     {
17       cout<<mid<<endl;
18       return 0;
19     }
20     else if(key<a[mid])                 //如 x 小于中间值, 则取前半段
21       search(bot,mid-1);
22     else                                //如 x 大于中间值,则取后半段
23       search(mid+1,top);
24   }
```

```
25      else
26      {
27          printf("-1\n");
28          return 0;
29      }
30  }
31
32  int main()
33  {
34      freopen("half.in","r",stdin);
35      freopen("half.out","w",stdout);
36      int n;
37      cin>>n;
38      for(int i=1;i<=n;++i)
39          cin>>a[i];
40      cin>>key;
41      search(1,n);
42      return 0;
43  }
```

考虑一个猜数字游戏：即取任何一个大于0小于1 024的自然数，提问方最多问10次这个数比某数大吗？被提问方只需回答“是”或者“不是”，提问方就可以猜出这个数字。呵呵，你能明白其中的奥秘吗？

非递归二分法★

折半查找法除了可以用递归的方法外，还可以用非递归的方法，完整代码如下所示：

```
1   //折半查找法——非递归二分法
2   #include <iostream>
3   #include <cstdlib>
4   #define MAXN 10001
5   using namespace std;
6   int key,top,bot,mid,n,a[MAXN];
7
8   void half()                    //二分查找法
9   {
10      top=1;
11      bot=n;
12      while (top<=bot)
13      {
14          mid=(bot+top)/2;
15          if (key==a[mid])       //如果正好找到
16          {
17              cout<<mid<<endl;
18              exit(0);
19          }
20          else if (key<a[mid])   //选择左半段
```

```
21          bot=mid-1;
22        else                          //选择右半段
23          top=mid+1;
24      }
25      cout<<-1<<endl;
26  }
27
28  int main ()
29  {
30      freopen("half.in","r",stdin);
31      freopen("half.out","w",stdout);
32      cin>>n;
33      for(int i=1;i<=n;i++)
34          cin>>a[i];
35      cin>>key;
36      if (key<a[1] || key>a[n])
37          cout<<-1<<endl;
38      else
39          half();
40      return 0;
41  }
```

考虑一个放魔法石的游戏：把1 000个魔法石装在10个袋子中，任取其中的一袋，或把几个袋中的魔法石数加起来，都能凑成1～1 000中的任何一种魔法石数。你知道这10个袋中分别装了多少个魔法石吗？

拓展与练习

【题目描述】 神族文字(dictionary.cpp/c/pas)POJ 2503

楚继光发现图书馆里收藏有许多上古时代的魔法书，这些上古时代的魔法书使用一种传说中的“神族文字”来书写，幸运的是，楚继光手边恰巧有一本词典可以帮助他。

【输入格式】

输入的词典内容最多包含有100 000个词条，每一个词条包含一个英文单词，其次是一个空格和一个对应的“神族文字”。没有一个“神族文字”在词典中出现一次以上。词典词条全部输入完毕后是一个空行，之后是需要翻译的“神族文字”，每一个词一行，每个单词是一个最多为10个小写字母的字符串。

【输出格式】

输出翻译好的英文，每行一个字。若词典中查找不到，输出“eh”。

【输入样例】

dog ogday

cat atcay

pig igpay

froot ootfray

loops oopslay

atcay
ittenkay
oopslay

【输出样例】

cat
eh
loops

魔法石的诱惑

【题目描述】 魔法石的诱惑(rob.cpp/c/pas)

修罗王远远地看见邪狼狂奔而来,问道:“慌慌张张地跑什么?”

邪狼大口大口地喘气:“我路过一家魔法石店,看到摆着那么多高阶魔法石,我就跑进去抢了一大袋。”

修罗王怒道:“光天化日,朗朗乾坤,众目睽睽之下,你也敢抢?”

邪狼:“我抢魔法石的时候,压根儿就没看见人,眼里只看见魔法石了。”

修罗王:“……”

其实邪狼的贪婪很容易理解,因为高阶魔法石有一个特征,即它的重量进行阶乘运算后末尾有几个0,就拥有同等重量普通魔法石几倍的魔法力。例如5!=5×4×3×2×1=120,而120结尾包含1个零,这意味着该魔法石拥有同等重量的普通魔法石1倍的魔法力。你的任务是找到最小自然数N,使N!在十进制下包含Q个零。

【输入格式】

一个数Q($0\leqslant Q\leqslant 10^8$)。

【输出格式】

如果无解,输出"No solution",否则输出N。

【输入样例】

2

【输出样例】

10

分治算法★

这个很简单啊,就好像猜1～100之间的一个数字,我先猜50,如果大了,我就缩小范围到0～49,如果小了,我就缩小范围到51～100。

本题中,N!随着N的增加,0的个数也在增加,所以是一个单调递增序列,显然可以用二分法很快求出答案。

至于判断N!有多少个零,本书的第一部里曾经提到过一个公式是$N/5+N/(5^2)+N/(5^3)+\cdots$

参考程序如下所示：

```
//魔法石的诱惑——分治算法
#include <iostream>
#include <cstdio>
#include <cstdlib>
using namespace std;

int solve(int n)
{
  int ans = 0;
  while (n > 0)
  {
    ans = ans + n / 5;
    n = n / 5;
  }
  return ans;
}

void run()
{
  int Q, i;
  scanf("%d", &Q);
  int start = 1;
  int end = 500000000;
  int ans = 500000001;
  int mid;
  int t;
  while (start <= end)        //二分查找
  {
    int mid = (end - start) / 2 + start;
    int t = solve(mid);
    if (t == Q && mid < ans)
      ans = mid;
    if (t > Q)
      end = mid - 1;
    else if ( t < Q )
      start = mid + 1;
    else
      end = mid - 1;
  }
  if (ans != 500000001)
    printf("%d\n", ans);
  else
    printf("No solution\n");
}

int main()
{
  run();
```

```
49    return 0;
50  }
```

数学方法★

这道题还可以用数学方法解决。

题目给出的是Q,要求出的是N,由于是要求出最小的自然数,所以N必定是5的倍数。又有:

$Q = N/5 + N/(5^2) + N/(5^3) + \cdots$

$\Rightarrow Q = N(5^k - 1) / [4 \times (5^k)]$(根据等比数列的求和公式)

$\Rightarrow N = 4Q \times [(5^k)/(5^k - 1)]$

注意到$1 < (5^k)/(5^k-1) \leqslant 5/4$,且当k趋于无穷时,$(5^k)/(5^k-1)$趋于1,所以可先算出N=4Q的末尾零的个数与所给的Q比较,显然所求的数就在4Q的附近。

参考程序如下所示:

```
1   //魔法石的诱惑——数学方法
2   #include <iostream>
3   #include <cstdio>
4   #include <cstdlib>
5   using namespace std;
6
7   int ZeroTrail(int n)              //计算 n!的 0 的个数
8   {
9     int count = 0;
10    while(n)
11    {
12      count+=n/5;
13      n/=5;
14    }
15    return count;
16  }
17
18  int main()
19  {
20    int q;
21    scanf ("%d", &q);
22    if (!q)
23    {
24      printf ("1\n");
25      return 0;
26    }
27    int i = 4 * q/5 * 5;
28    while (ZeroTrail(i) < q)
29      i += 5;
30    if (q == ZeroTrail(i))
31      printf ("%d\n", i);
32    else
```

```
33        printf ("No solution\n");
34    return 0;
35  }
```

拓展与练习

【题目描述】 近似整数(Approximation. cpp/c/pas) POJ 1650

给定一个浮点数 A 和一个整数 L,求在范围[1,L]内的两个整数 n 和 d,使得 n/d 能近似等于 A,且使误差|A－n/d|最小。

【输入格式】

第一行为一个浮点数 A,第二行为一个整数 L。

【输出格式】

两个整数 n 和 d。

【输入样例】

3.141 592 653 589 79

10 000

【输出样例】

355 113

逃亡

【题目描述】 逃亡(escape. cpp/c/pas)

邪狼紧张地说:“老大,警察快追过来了,我们快逃跑吧!”

修罗王傲然道:“在我的字典里没有逃跑……”

邪狼内心崇敬地想:“老大实在是太有领袖范了……”

修罗王接着说:“只有战略转移。”

邪狼:“……”

现在,修罗王和邪狼两人需要从 A 地出发尽快到达 B 地。出发时 A 地有一辆可带一人的自动驾驶悬浮车。又知两人步行速度相同。问怎样利用小车才能使两人尽快同时到达 B 地。

【输入格式】

输入文件为 escape. in,有三个 int 类型整数,分别表示 A、B 两地的距离,步行速度和车速。

【输出格式】

输出文件为 escape. out,有一个小数位数为 2 的浮点数,即最短时间。

【输入样例】

100 5 10

【输出样例】

14.00

分治算法★

如图 1.2 所示，设两人分别为甲、乙，则最优方案应该是甲先乘车到达 C 后下车步行，小车回头接已经走到 E 的乙，假设在 D 相遇，乙乘车到达 B 时正好甲也步行到达，这样花费的时间最短。

图　1.2

那就是求 C 点的位置了。设 A、B 距离为 S，步行速度为 a，车速为 b，甲耗时 T_1，乙耗时 T_2。

再设 C_0 为起点位置，C_1 为终点位置，取中点位置 $C=(C_0+C_1)/2$ 作为测试点，即计算甲耗时 T_1 与乙耗时 T_2，若 $T_1<T_2$，则取 C 与 C_0 的中点，否则取 C 与 C_1 的中点，如此反复循环直到 $T_1=T_2$ 即可。

参考程序如下所示：

```
1  //逃亡——分治算法
2  #include <cstdio>
3  #include <cstdlib>
4  #include <math.h>
5  #include <iostream>
6  using namespace std;
7
8  int main()
9  {
10   freopen("escape.in","r",stdin);
11   freopen("escape.out","w",stdout);
12   float s,a,b,c,c0,c1,t1,t2,t3,t4 ;
13   cin>>s>>a>>b;
14   c0=0;c1=s;
15   do
16   {
17     c=(c0+c1)/2.0;
18     t3=c/b;                          //甲乘车到C的时间
19     t1=t3+(s-c)/a;                   //甲用的总时间
20     t4=(c-t3*a)/(a+b);               //小车从C回头与乙相遇的时间
21     t2=t3+t4+(s-(t3+t4)*a)/b;        //乙用的总时间
22     if(t1<t2)
23       c1=c;
24     else
25       c0=c;
26   }while(fabs(t1-t2)>1e-4);
27   printf("%4.2f",t1);
28   return 0;
29 }
```

数学方法 1★★

分治算法是好，但如果数学功底够好，完全可以用数学推导算出结果。另外需要注意的是，由于涉及浮点数运算，所以不同的算法或数学运算过程，其结果可能会略有不同。

如图 1.3 所示：设 S＝总路程，V1＝步行速度，V2＝车速，t1＝甲的乘车时间，t2＝车返回与乙相遇的时间，t3＝甲步行到终点的时间(即乙乘车到终点的时间)，P1 为车返回处，P2 为车与乙相遇入，Q1 为车返回时乙当时所处的位置。

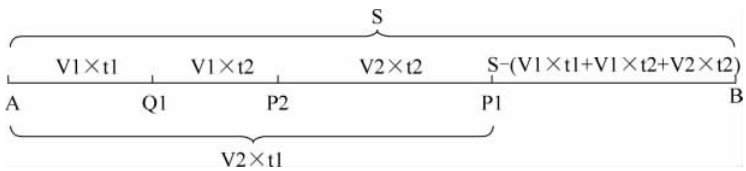

图 1.3

首先求 Q1 到 P1 的距离，显然有 Q1P1＝AP1－AQ1＝V2t1－V1t1＝t1(V2－V1)

则车返回与乙相遇的时间：$t2=\frac{Q1P1}{V1+V2}\Rightarrow t2=\frac{t1(V2-V1)}{V1+V2}$ ①

则乙乘车到终点的时间：$t3=\frac{S-V1t1-V1t2}{V2}$

$=\frac{S-V1t1-V1t2-V2t2}{V1}-t2$（即甲步行到终点的时间）

$=\frac{S-V1t1-V1t2-V2t2}{V1}-\frac{t1(V2-V1)}{V1+V2}$

推出：$\frac{S-V1t1-V1t2}{V2}=\frac{S-V1t1-V1t2-V2t2}{V1}-\frac{t1(V2-V1)}{V1+V2}$ ②

设 $k=\frac{V2-V1}{V1+V2}$，则由式①可得 t2＝kt1 ③

将式③代入式②，得$\frac{S-V1(t1+kt1)}{V2}=\frac{S-V1(t1+kt1)-V2kt1}{V1}-kt1$

⇒V1S－$V1^2$(1＋k)t1＝V2S－V1V2(1＋k)t1－$V2^2$kt1－V1V2kt1（等式两边同乘以 V1V2）

⇒(V1－V2)S＝[$V1^2$(1＋k)－V1V2(1＋k)－$V2^2$k－V1V2k]t1

故 $t1=\frac{(V1-V2)S}{V1^2(1+k)-V1V2(1+2k)-V2^2k}$

$\Rightarrow t1=\frac{(V1-V2)S}{V1^2-V1V2+(V1^2-2V1V2-V2^2)k}$

$\Rightarrow t1=\frac{(V1-V2)S}{V1^2-V1V2+(V1^2-2V1V2-V2^2)\frac{V2-V1}{V1+V2}}$

$\Rightarrow t1=\frac{\frac{S}{V2}-\frac{S}{V1}}{\frac{V1+\frac{(V2-V1)V1}{V1+V2}}{V2}-1-\frac{(V2-V1)\left(\frac{V2}{V1}+2\right)}{V1+V2}}$（上下同除以 V1V2）