

计算机算法设计与分析

动态规划

易凯

2017 年 4 月 18 日

班 级 软件 53 班

学 号 2151601053

邮 箱 williamyi96@gmail.com

联系电话 13772103675

个人网站 <https://williamyi96.github.io>
williamyi.tech

实验日期 2017 年 4 月 25 日

提交日期 2017 年 6 月 6 日

目录

1	贪心算法相关概念	5
2	贪心算法基本要素	5
3	贪心算法与动态规划的区别	5
4	哈夫曼编码问题	5
4.1	试题描述	5
4.2	题目分析与求解	5
5	磁盘文件最优存储问题	6
5.1	试题描述	6
5.2	算法设计	7
5.3	数据输入	7
5.4	数据输出	7
5.5	问题分析	7
5.6	程序代码	8
5.7	样例运行	9
6	程序存储问题	9
6.1	试题描述	9
6.2	算法设计	9
6.3	数据输入	9
6.4	数据输出	9
6.5	问题分析	10
6.6	程序代码	10
6.7	样例运行	11
7	最优服务次序问题	11
7.1	试题描述	11
7.2	算法设计	11
7.3	数据输入	11
7.4	数据输出	11
7.5	问题分析	11

目录	3
7.6 程序代码	11
7.7 样例运行	12
8 汽车加油问题	12
8.1 试题描述	12
8.2 算法设计	13
8.3 数据输入	13
8.4 数据输出	13
8.5 问题分析	13
8.6 程序代码	13
8.7 样例运行	14
8.8 算法最优解证明	14
9 磁带最大利用率问题	14
9.1 试题描述	14
9.2 算法设计	15
9.3 数据输入	15
9.4 数据输出	15
9.5 程序代码	15
10 最优分解问题	15
10.1 试题描述	15
10.2 算法设计	15
10.3 数据输入	15
10.4 数据输出	15
10.5 问题分析	16
10.6 程序代码	16
10.7 样例运行	17

插图

1	斐波那契哈夫曼编码 1	6
2	斐波那契哈夫曼编码 2	6
3	斐波那契哈夫曼编码 3	7
4	磁盘文件最优存储问题运行示例	9
5	磁盘存储问题样例运行	11
6	最优服务次序问题样例输出	12
7	汽车加油问题样例运行	14
8	最优分解问题样例运行	17

1 贪心算法相关概念

贪心算法是一种逐步逼近最优解的算法，仅考虑当前如何决策可以达到最优，而不考虑全局最优。虽然由其得到的解不一定为最优解，但是其逼近最优解，或者说是最优解的一个近似解。

2 贪心算法基本要素

1. 贪心选择性质
2. 最优子结构性质

3 贪心算法与动态规划的区别

贪心算法与动态规划的相同点是两者都需要研究的问题具有最优子结构。在动态规划算法中，每步所做的选择往往依赖于相关子问题的解，只有在解出相关子问题之后才能够做出选择；但是在贪心算法中，仅在当前状态下做出最好的选择，即局部最优选择。

4 哈夫曼编码问题

4.1 试题描述

字符 a-h 出现的频率恰好是前 8 个 Fibonacci 数，它们的哈夫曼编码是什么？将结果推广到 n 个字符的频率恰好是前 n 个 Fibonacci 数的情形。

4.2 题目分析与求解

哈夫曼编码是一种将使用字符在文件中出现的频率表构建的不定长码。

其中前八个 Fibonacci 数分别为 1,1,2,3,5,8,13,21。

如果是 n 个字符的频率满足 Fibonacci 数列的情况，则最终的结果为：

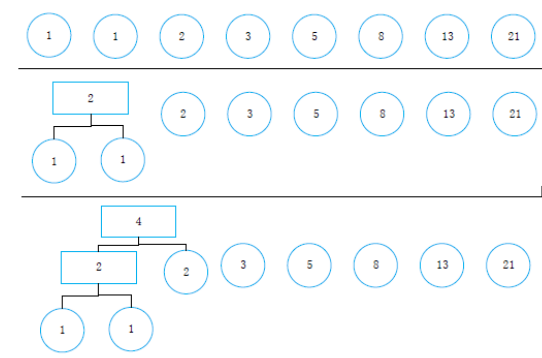


图 1: 斐波那契哈夫曼编码 1

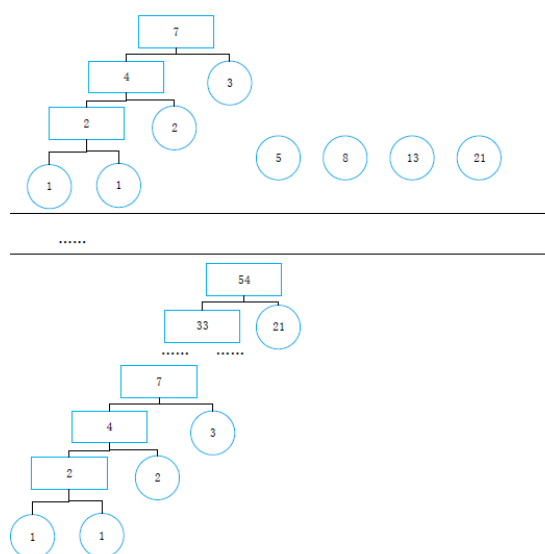


图 2: 斐波那契哈夫曼编码 2

5 磁盘文件最优存储问题

5.1 试题描述

设磁盘上有 n 个文件 f_1, f_2, \dots, f_n , 每个文件占用磁盘上的 1 个磁道。这 n 个文件的检索概率分别是 p_1, p_2, \dots, p_n , 且 $\sum_{i=1}^n p_i = 1$ 。磁头从当前磁道移到被检信息磁道所需的时间可用这 2 个磁道之间的径向距离来度量。如果文件 f_i 存放在第 i 道上, $1 \leq i \leq n$, 则检索这 n 个文件的期望时间对于所有的

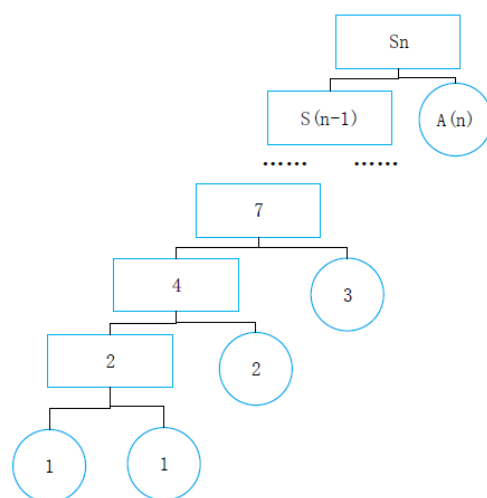


图 3: 斐波那契哈夫曼编码 3

$i < j, \text{time} += p_i * p_j * d(i, j)$ 。其中 $d(i, j)$ 是第 i 道与第 j 道之间的径向距离 $|i - j|$ 。磁盘文件的最优存储问题要求确定这 n 个文件在磁盘上的存储位置, 使期望检索时间达到最小。

5.2 算法设计

对于戈丁的文件检索概率, 计算磁盘文件的最优存储方案。

5.3 数据输入

第 1 行是正整数 n , 表示文件个数。第 2 行有 n 个正整数 a_i , 表示文件的检索概率。

5.4 数据输出

计算的最小期望检索时间

5.5 问题分析

此题是使用贪心算法求解的典型, 其中先将 n 个文件按访问概率从大到小进行排序, 其中概率最大的放在中间磁道上, 次大和次次大的放在最大

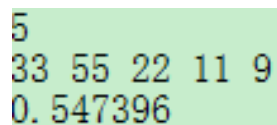
的两边，逐级递推，最后访问概率最小的放在最边上。

5.6 程序代码

```
1 #include <iostream>
2 #include <stdio.h>
3 #include <algorithm>
4 using namespace std;
5
6 int cmp (const void *a , const void *b) {
7     return *(double *)a - *(double *)b;
8 }
9
10 double greedy(double a[], int n) {
11     qsort(a,n,sizeof(double),cmp);
12     int mid = (n - 1) / 2;
13     double x[n];
14     x[mid] = a[n-1];
15     for(int i = mid+1; i < n; i++)
16         x[i] = a[n - 2*(i-mid)];
17     for(int i = mid-1; i >= 0; i--)
18         x[i] = a[n - 2*(mid-i) - 1];
19     double sum = 0, exp = 0;
20     for(int i = 0; i < n; i++) {
21         sum += a[i];
22         for(int j = i+1; j < n; j++)
23             exp += x[i]*x[j]*(j-i);
24     }
25     return exp/sum/sum;
26 }
27
28 int main() {
29     int i, j, n;
30     double a[1000], exp;
31     scanf("%d",&n);
32     for(i = 0; i < n; i++)
33         scanf("%lf",&a[i]);
34     exp = greedy(a,n);
35     printf("%lf\n",exp);
```


36 }

5.7 样例运行



```
5
33 55 22 11 9
0.547396
```

图 4: 磁盘文件最优存储问题运行示例

6 程序存储问题

6.1 试题描述

设有 n 个程序 $1, 2, 3, \dots, n$ 要存放在长度为 L 的磁带上，程序 i 存放在磁带上的长度为 l_i 。程序存储问题要求确定这 n 个程序在磁带上的一种存储方案，使得能够在磁带上存储尽可能多的程序。

6.2 算法设计

对于给定的 n 个程序存放在磁带上的长度，计算磁带上最多可以存储的程序数。

6.3 数据输入

输入的格式是第一行是两个正整数，分别表示文件个数 n 和磁带的长度 l ，接下来的一行中，有 n 个正整数，表示程序存放在磁盘上的长度。

6.4 数据输出

将计算的最多可以存储的程序数输出到文件中。

6.5 问题分析

此题可以使用贪心算法求解近似最优解。其满足贪心算法使用的贪心性质和最优子结构性质。因此其贪心策略就是将文件长度最小的文件存入磁带上。

6.6 程序代码

```
1  #include<iostream>
2  #include<algorithm>
3  using namespace std;
4
5  int a[100000];
6
7  int most(int *a, int n, int s) {
8      int i=0, sum=0;
9      while(i<n) {
10         sum=a[i]+sum;
11         if(sum<=s) i++;
12         else return i;
13     }
14     return n;
15 }
16
17 int main() {
18     int i, n, s;
19     scanf("%d %d\n", &n, &s);
20     for(i=0; i<n; i++) scanf("%d", &a[i]);
21     sort(a, a+n);
22     printf("%d", most(a, n, s));
23     return 0;
24 }
```

```
6 50
2 3 13 8 80 20
5
-----
Process exited after 13.86 seconds with return value 0
```

图 5: 磁盘存储问题样例运行

6.7 样例运行

7 最优服务次序问题

7.1 试题描述

设有 n 个顾客同时等待一项服务。顾客 i 需要的服务时间为 $t_i, 1 \leq i \leq n$ 。应该如何安排 n 个顾客的服务次序才能使平均等待时间达到最小？平均等待时间是 n 个顾客等待服务时间的总和除以 n 。

7.2 算法设计

对于给定的 n 个顾客需要的服务时间，计算最优服务次序。

7.3 数据输入

输入的格式中第一行为正整数 n ，表示 n 个顾客，接下来的一行中，有 n 个正整数，表示 n 个顾客需要的服务时间。

7.4 数据输出

计算出最小平均等待时间输出到屏幕之上。

7.5 问题分析

此题可以使用贪心算法来求得最优解。其中该题目具有贪心选择性质以及最优子结构性质。

7.6 程序代码

```
1 #include<iostream>
2 #include<cstring>
3 #include<algorithm>
4 using namespace std;
5
6 int main()
7 {
8     int n;
9     cin >> n;
10    int a[100] = { 0 };
11    for (int i = 1; i <=n; i++)
12        cin >> a[i];
13    sort(a, a+n);
14    double sum=0;
15    for (int i = 1; i <= n; i++)
16        sum +=(n+1-i)*a[i];
17    cout << sum / n<<endl;
18    return 0;
19 }
```

7.7 样例运行

```
10
56 12 1 99 1000 234 33 55 99 812
550.8
-----
Process exited after 16.1 seconds with return value 0
```

图 6: 最优服务次序问题样例输出

8 汽车加油问题

8.1 试题描述

一辆汽车加满油后能够行驶 n km，旅途中有若干个加油站。设计一个有效的算法，指出应该在那些加油站停靠加油，使沿途加油次数最少，并证明算法能够产生一个最优解。

8.2 算法设计

对于给定的 n 和 k 个加油站位置，计算最少加油次数。

8.3 数据输入

文件的输入格式为第一行有两个正整数 n 和 k ，其中表示加满油可行驶 n km，且旅途中有 k 个加油站。接下来的一行中，有 $k+1$ 个整数，表示第 k 个加油站与第 $k-1$ 个加油站之间的距离。第 0 个加油站表示出发地，汽车已经加满了油。第 $k+1$ 个加油站表示目的地。

8.4 数据输出

将计算的最少加油次数输出到屏幕上。如果无法叨叨目的地，则输出“No Solution”。

8.5 问题分析

8.6 程序代码

```
1 #include <stdio.h>
2
3 void greedy(int d[], int n, int k) {
4     int num = 0;
5     for(int i = 0; i < k; i++) {
6         if(d[i] > n) {
7             printf("no solution/n");
8             return;
9         }
10    }
11    for(int i = 0, s = 0; i < k; i++) {
12        s += d[i];
13        if(s > n) {
14            num++;
15            s = d[i];
16        }
17    }
18    printf("%d\n", num);
```

```
19 }
20
21 int main() {
22     int i,n,k;
23     int d[1000];
24     scanf("%d %d",&n,&k);
25     for(i = 0; i < k; i++)
26         scanf("%d",&d[i]);
27     greedy(d,n,k);
28 }
```

8.7 样例运行

```
7 7
1 2 3 4 5 1 6 6
4
-----
Process exited after 8.065 seconds with return value 0
```

图 7: 汽车加油问题样例运行

8.8 算法最优解证明

使用反证法。假设 a 为起始位点, b 为终止位点, 如果存在中间位点 c 使 a - c 加油次数更少, 那么 a - c , c - b 联合所得到的加油次数最少, 则原策略不是 a - b 的最优策略。但是可以使用 a - c 策略进行替换, 使其具有最优策略。综上所述, 算法能够产生最优解。这么证明是否具有说服力??

9 磁带最大利用率问题

9.1 试题描述

设有 n 个程序 $1, 2, \dots, n$ 要存放在长度为 L 的磁带上。程序 i 存放在磁带上的长度是 l_i , $1 \leq i \leq n$ 。程序存储问题要求确定这 n 个程序在磁带上的一个存储方案, 使得能够在磁带上存储尽可能多的程序。在保证存储最多程序的前提下还要求磁带的利用率达到最大。

9.2 算法设计

对于给定的 n 个程序存放在磁带上的长度，编程计算磁带上最多可以存储的程序数和占用磁带的长度。

9.3 数据输入

第一行是 2 个正整数，分别表示文件个数 n 和磁带的长度 L 。接下来的 1 行中，有 n 个正整数，表示程序存放在磁带上的长度。

9.4 数据输出

第 1 行输出最多可以存储的程序数和占用磁带的长度；第 2 行输出存放在磁带上的每个程序的长度。

9.5 程序代码

此题暂时没有找到一种好的贪心方式进行求解。

10 最优分解问题

10.1 试题描述

设 n 是一个正整数。现在要求将 n 分解成若干个互不相同的自然数的和，且使这些自然数的乘积最大。

10.2 算法设计

对于给定的正整数 n ，计算最优分解方案。

10.3 数据输入

输入仅有一行，表示的是待分解的自然数。

10.4 数据输出

输出是计算得到的最大乘积。

10.5 问题分析

对整数分析可有结论：若 $a+b=\text{const}$ ，则 $|a-b|$ 越小， $a*b$ 越大。根据原问题的描述，需要将正整数 n 分解为若干互不相同的自然数的和，同时又要使自然数的乘积最大。当 $n<4$ 时，对 n 的分解的乘积是小于 n 的；当 n 大于或等于 4 时， $n=1+(n-1)$ 因的乘积也是小于 n 的，所以 $n=a+(n-a)$ ， $2 \leq a \leq n-2$ ，可以保证乘积大于 n ，即越分解乘积越大。因此可以采用如下贪心策略：将 n 分成从 2 开始的连续自然数的和，如果最后剩下一个数将此数在后项优先的方式下均匀地分给前面各项。该贪心策略首先保证了正整数所分解出的因子之差的绝对值最小，即 $|a-b|$ 最小；同时又可以将其分解成尽可能多的因子，且因子的值较大，确保最终所分解的自然数的乘积可以取得最大值。

10.6 程序代码

```
1 #include<iostream>
2 using namespace std;
3 int main() {
4     int n,sum=0,i,d;
5     cin>>n;
6     for(i=2;;++i) {
7         sum+=i;
8         d=sum-n;
9         if(d>=0) break;
10    }
11    unsigned int result=1;
12    if(d==1) {
13        for(int j=3;j<i;++j) result*=j;
14        i++;
15        result *= i;
16    } else {
17        for(int j=2;j<=i;++j) {
18            if(j==d) continue;
19            result*=j;
20        }
21    }
22    //cout<<"The result is "<<result<<endl;
23    cout << result << endl;
```



```
24     return 0;  
25 }
```

10.7 样例运行

```
10  
30  
-----  
Process exited after 2.117 seconds with return value 0
```

图 8: 最优分解问题样例运行