**实验一 集合的表示与操作算法设计**

**xxxxxxx**

目录

[实验目的 1](#_Toc10490347)

[实验内容 1](#_Toc10490348)

[概述 1](#_Toc10490349)

[统计数字问题 1](#_Toc10490350)

[最多约数问题 3](#_Toc10490351)

[字典序问题 4](#_Toc10490352)

[贪心算法 9](#_Toc10490353)

[程序存储问题 9](#_Toc10490354)

[汽车加油问题 10](#_Toc10490355)

[最优分解问题 12](#_Toc10490356)

[递归与分治策略 14](#_Toc10490357)

[众数问题 14](#_Toc10490358)

[整数因子分解问题 16](#_Toc10490359)

# 实验目的

通过这次实验了解体会并掌握基本的递归分治算法以及贪心算法的思想，并有能力解决一些具体的问题，通过c++来实现解题的过程，进一步的熟悉算法的流程。

# 实验内容

实验大致分为三部分： **概述** 、 **递归与分治策略** 、 **贪心算法** 。对于每一类问题，选择至少一道题目进行思考并用代码验证算法的正确性，分析算法的可行性。

## 概述

### 统计数字问题

#### 题目

问题描述：  
  
　　一本书的页码从自然数1 开始顺序编码直到自然数n。书的页码按照通常的习惯编排，每个页码都不含多余的前导数字0。例如，第6 页用数字6 表示，而不是06 或006 等。数字计数问题要求对给定书的总页码n，计算出书的全部页码中分别用到多少次数字0，1， 2，…，9。  
  
编程任务：  
  
　　给定表示书的总页码的10 进制整数n (1≤n≤109) 。编程计算书的全部页码中分别用到多少次数字0，1，2，…，9。  
  
数据输入：  
  
　　输入数据由文件名为input.txt 的文本文件提供。每个文件只有1 行，给出表示书的总页码的整数n。  
  
结果输出:   
  
　　程序运行结束时，将计算结果输出到文件output.txt 中。输出文件共有10 行，在第k 行输出页码中用到数字k-1 的次数，k=1，2，…，10。  
  
输入文件示例   
input.txt   
11  
输出文件示例  
output.txt   
1   
4   
1   
1   
1   
1   
1   
1   
1   
1

#### 算法思路

使用一个10位大小的整型数组保存每一个数字出现的次数，对于 中出现的每一个数，求出最后一位数，个数增一，最后输出所有的 数字的个数即可。

#### 实验程序

#include <bits/stdc++.h>  
using namespace std;  
const int maxn = 10;  
int a[maxn];  
void solve(int n)  
{  
 while(n)  
 {  
 ++a[n % 10];  
 n /= 10;  
 }  
 return;  
}  
int main()  
{  
 freopen("input.txt", "r", stdin);  
 freopen("output.txt", "w", stdout);  
 int n;  
 scanf("%d", &n);  
 memset(a, 0, sizeof a);  
 for(int i = 1; i <= n; ++i)solve(i);  
 for(int i = 0; i <= 9; ++i)printf("%d\n", a[i]);  
}

#### 测试结果

input:  
11  
  
output:  
1   
4   
1   
1   
1   
1   
1   
1   
1   
1

### 最多约数问题

#### 题目

问题描述：  
  
　　正整数x 的约数是能整除x 的正整数。正整数x 的约数个数记为div(x) 。例如，1，2， 5，10 都是正整数10 的约数，且div(10)=4 。设a 和b 是2 个正整数，a≤b，找出a 和b 之间约数个数最多的数x。  
  
编程任务：  
  
　　对于给定的2 个正整数a≤b，编程计算a 和b 之间约数个数最多的数。  
  
数据输入：  
  
　　输入数据由文件名为input.txt 的文本文件提供。文件的第1 行有2 个正整数a 和b。  
  
结果输出:   
  
　　程序运行结束时，若找到的a 和b 之间约数个数最多的数是x，将div(x)输出到文件output.txt 中。  
  
输入文件示例 输出文件示例  
input.txt output.txt   
　1 36 9

#### 算法思路

预处理出所有 的每一个数的因数的个数，然后对每一个测试遍历 找出因数最多的数，输出即可。

#### 实验程序

#include <bits/stdc++.h>  
using namespace std;  
const int maxn = 1e5 + 5;  
int a[maxn];  
void init()  
{  
 for(int i = 2; i <= maxn; ++i)  
 for(int j = i; j <= maxn; j += i)  
 ++a[j];  
 return;  
}  
int main()  
{  
 // freopen("input.in", "r", stdin);  
 // freopen("output.out", "w", stdout);  
 init();  
 int l, r;  
 while(~scanf("%d%d", &l, &r))  
 {  
 int ans = 0;  
 for(int i = l; i <= r; ++i)  
 ans = max(ans, a[i]);  
 printf("%d\n", ++ans);  
 }  
}

#### 测试结果

input.txt output.txt   
　1 36 9

### 字典序问题

#### 题目

问题描述：  
  
　　在数据加密和数据压缩中常需要对特殊的字符串进行编码。给定的字母表A 由26 个小写英文字母组成A={a,b,…,z}。该字母表产生的升序字符串是指字符串中字母按照从左到右出现的次序与字母在字母表中出现的次序相同，且每个字符最多出现1 次。例如，a,b,ab,bc,xyz 等字符串都是升序字符串。现在对字母表A 产生的所有长度不超过6 的升序字符串按照字典序排列并编码如下。  
1 2   
…   
26 27 28   
…   
a b   
…   
z ab ac   
…   
对任意长度不超过6 的升序字符串，迅速计算出它在上述字典中的编码。  
  
编程任务：  
  
　　对于给定的长度不超过6 的升序字符串，编程计算出它在上述字典中的编码。  
  
数据输入：  
输入数据由文件名为input.txt 的文本文件提供文件的第一行是一个正整数k，表示接下来共有k 行接下来的k 行中，每行给出一个字符串。  
  
结果输出:   
  
　　程序运行结束时，将计算结果输出到文件output.txt 中。文件共有k 行，每行对应于一  
个字符串的编码。  
   
输入文件示例 输出文件示例  
input.txt output.txt   
2 1   
a 2   
b

#### 算法思路

##### 法1.找规律递归分段求和

手推出几个字符串的序号后可以看出，要求一个字符串的编号，可以通过 来求得待求字符串的编号，显然三次计算都有一个公共操作： **求一个串长为len且开头为a的所有字符串的数量** ， 定义 表示以字符a开头的长度为k的字符串的数量， 显然可以通过所有长度为 开头字符为大于a的字符与a的拼接可以得到，所以我们求出长度为 开头为 的所有字符串的数量和便可以得到长度为 且开头为a的字符串的数量，也就是说： ，这个可以通过递归的方式得到。

于是整个问题的结果流程为：

* 计算出所有长度为k-1的字符串的数量（此时开头字母为所有）调用26次sum()函数
* 计算出开头字符小于待求字符串开头字符的所有长度为k的字符串：调用 次sum()函数
* 计算长度为 且后面几位的字符小于待求字符串对应的那一位的字符串的和

例如：

s: cefvz

* 先求出所有长度小于5的数量，即： 的数量
* 求出 的数量，这一段就是求长度为5的，开头小于’c’的字符串的数量
* 求出 的数量，这一段就是求长度为k-1=4的从’defg’到’efvz’的所有字符串的数量

##### 法2.暴力深搜保存字典hash

因为题目给出的要求字符串最大长度仅为6，所以可以深搜出所有的符合条件的字符串，（dfs序即为字符串的序号），然后对于每一个字符串给定一个hash值，最后对于每一个询问根据字符串的hash便可得出序号。（通过这个方法可以生成测试数据）

#### 实验程序

##### 法1

#include <bits/stdc++.h>  
using namespace std;  
const int maxn = 1e5;  
char s[10];  
  
int sum(int i, int j)  
{  
 int ret = 0;  
 if(j == 1)return 1;  
 else  
 {  
 for(int k = i + 1; k <= 26; ++k)  
 ret += sum(k, j - 1);  
 }  
 return ret;  
}  
void solve()  
{  
 int ans = 0;  
 int len = strlen(s);  
 for(int i = 1; i <= len - 1; ++i)  
 for(int j = 1; j <= 26; ++j)  
 ans += sum(j, i);  
 // cout << ans << "--" << endl;  
 for(int i = 1; i <= s[0] - 'a' + 1 - 1; ++i)  
 ans += sum(i, len);  
 // cout << ans << "--" << endl;  
 for(int i = 1; i <= len - 1; ++i)  
 {  
 for(int j = s[i - 1] - 'a' + 1 + 1; j <= s[i] - 'a' + 1 - 1; ++j)  
 ans += sum(j, len - i);  
 }  
 printf("%d\n", ++ans);  
}  
  
int main()  
{  
// freopen("input.txt", "r", stdin);  
// freopen("output.txt", "w", stdout);  
  
 int t;scanf("%d", &t);  
 while(t--)  
 {  
 scanf("%s", &s);  
 solve();  
 }  
 return 0;  
}

##### 法2

#include <bits/stdc++.h>  
using namespace std;  
const int maxn = 1e5;  
char s[10];  
const int p = 1e3+7;  
unordered\_map<int, int> mp;  
inline int gethash(int len)  
{  
 int ret = 1;  
 for(int i = 0; i <= len - 1; ++i)  
 ret += ret \* p + s[i] - 'a';  
 return ret;  
}  
int tot = 0;  
void print(int len, int n)  
{  
 if(n == len)  
 {  
// printf("%s", s);  
 // for(int i = 0; i < len; ++i)printf("%c", s[i]);  
 // printf("\n");  
 mp[gethash(len)] = ++tot;  
 return;  
 }  
 if(n == 0)  
 {  
 for(int i = 0; i < 26; ++i)  
 {  
 s[0] = (char)(i + 'a');  
 print(len, n + 1);  
 }  
 }  
 else  
 {  
 for(int i = 0; i < 26; ++i)  
 {  
 if(s[n - 1] < i + 'a')  
 {  
 s[n] = (char)(i + 'a');  
 print(len, n + 1);  
 }  
 }  
  
 }  
  
 return;  
}  
void init()  
{  
 tot = 0;  
 for(int i = 1; i <= 6; ++i)  
 print(i, 0);  
}  
  
int main()  
{  
// freopen("input.txt", "r", stdin);  
// freopen("output.txt", "w", stdout);  
  
 init();  
 int t;scanf("%d", &t);  
 while(t--)  
 {  
 scanf("%s", &s);  
 printf("%d\n", mp[gethash(strlen(s))]);  
 }  
 return 0;  
}

#### 测试结果

input:  
4  
abc  
bciop  
acfuxz  
uvwxyz  
  
output:  
352  
31577  
98306  
313911

## 贪心算法

### 程序存储问题

#### 题目

算法实现题 程序存储问题  
  
问题描述：  
  
　　设有n 个程序{1,2,…, 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li，1≤i≤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。  
  
编程任务：  
  
　　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。  
  
数据输入：  
  
　　由文件input.txt 给出输入数据。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。  
  
结果输出:   
  
　　将编程计算出的最多可以存储的程序数输出到文件output.txt 。  
  
输入文件示例输出文件示例  
input.txt output.txt   
6 50 5  
2 3 13 8 80 20

#### 算法思路

因为要最大化存储的数量，所以优先考虑存储空间小的。

排序后取前面占用空间和刚好小于等于最大空间的个数。

#### 实验程序

#include <bits/stdc++.h>  
using namespace std;  
const int maxn = 1e5;  
int a[maxn];  
void solve()  
{  
 int n, l;  
 scanf("%d%d", &n, &l);  
 for(int i = 1; i <= n; ++i)scanf("%d", &a[i]);  
 sort(a + 1, a + 1 + n);  
 int ans = 0;  
 int sum = 0;  
 for(int i = 1; i <= n; ++i)  
 {  
 sum += a[i];  
 if(sum <= l)++ans;  
 else break;  
 }  
 printf("%d\n", ans);  
 return;  
}  
int main()  
{  
// freopen("input.txt", "r", stdin);  
// freopen("output.txt", "w", stdout);  
  
 solve();  
 return 0;  
}

#### 测试结果

input:  
6 50   
2 3 13 8 80 20  
  
output:  
5

### 汽车加油问题

#### 题目

算法实现题 汽车加油问题  
  
问题描述：  
  
　　一辆汽车加满油后可行驶n 公里。旅途中有若干个加油站。设计一个有效算法，指出应在哪些加油站停靠加油，使沿途加油次数最少。并证明算法能产生一个最优解。  
  
编程任务：  
  
　　对于给定的n 和k 个加油站位置，编程计算最少加油次数。  
  
数据输入：  
  
　　由文件input.txt 给出输入数据。第一行有2 个正整数n 和k，表示汽车加满油后可行驶n 公里，且旅途中有k 个加油站。接下来的1 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，汽车已加满油。第k+1 个加油站表示目的地。  
  
结果输出:   
  
　　将编程计算出的最少加油次数输出到文件output.txt 。如果无法到达目的地，则输出”No Solution”。  
  
输入文件示例输出文件示例  
input.txt output.txt   
7 7 4  
1 2 3 4 5 1 6 6

#### 算法思路

贪心考虑加油，当当前剩余油量不足以到达下一站是，在这一个站加油，否则一直走下去。

#### 实验程序

#include <bits/stdc++.h>  
using namespace std;  
const int maxn = 1e5;  
int n, k, a[maxn];  
int main()  
{  
 // freopen("input.txt", "r", stdin);  
 // freopen("output.txt", "w", stdout);  
 cin >> n >> k;  
 for(int i = 0; i <= k; ++i)cin >> a[i];  
 int ans = 0;  
 for(int i = 1; i <= k; ++i)  
 {  
 int sum = 0;  
 int nxt = i;  
 for(nxt = i; nxt <= k; ++nxt)  
 {  
 sum += a[nxt];  
 if(sum > n)break;  
 }  
 if(i == nxt)  
 {  
 cout << "No Solution" << endl;  
 return 0;  
 }  
 i = nxt - 1;  
 ++ans;  
 if(nxt == k + 1)break;  
 }  
 cout << ans - 1 << endl;  
 return 0;  
}

#### 测试结果

input:  
7 7  
1 2 3 4 5 1 6 6  
  
output;  
4

### 最优分解问题

#### 题目

算法实现 最优分解问题  
  
问题描述：  
  
　　设n 是一个正整数。现在要求将n 分解为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。  
  
编程任务：  
  
　　对于给定的正整数n，编程计算最优分解方案。  
  
数据输入：  
  
　　由文件input.txt 提供输入数据。文件的第1 行是正整数n。  
  
结果输出:   
  
　　程序运行结束时，将计算出的最大乘积输出到文件output.txt 中。  
  
输入文件示例 输出文件示例  
input.txt output.txt   
10 30

#### 算法思路

若 ， 当 很接近时，此时 最大，所以先将n分解为 且 ，对于多出来的一部分数，倒着为每一位加一，保证最后的乘积的最大。

#### 实验程序

#include <bits/stdc++.h>  
using namespace std;  
const int maxn = 1e5 + 5;  
int a[maxn];  
int n;  
void solve()  
{  
 for(int i = 1; i \* i / 4 <= n; ++i)  
 a[i] = i;  
 int sum = 0;  
 int last;  
 for(int i = 2; i \* i / 4 <= n; ++i)  
 {  
 sum += a[i];  
 if(sum >= n)  
 {  
 sum -= a[i];  
 sum = n - sum;  
 last = i - 1;  
 break;  
 }  
 }  
 while(sum)  
 {  
 ++a[last];  
 --sum;  
 if(!sum)break;  
 for(int i = last; i >= 2; --i)  
 {  
 --sum;  
 ++a[i];  
 if(!sum)break;  
 }  
 }  
 int ans = 1;  
 for(int i = 2; i <= last; ++i)ans \*= a[i];  
 cout << ans << endl;  
 return;  
}  
int main()  
{  
 // freopen("input.txt", "r", stdin);  
 // freopen("output.txt", "w", stdout);  
 while(~scanf("%d", &n))  
 solve();  
 return 0;  
}

#### 测试结果

intut:  
10  
  
output:  
30

## 递归与分治策略

### 众数问题

#### 题目

算法实现题 众数问题  
  
问题描述：  
  
　　给定含有n 个元素的多重集合S，每个元素在S 中出现的次数称为该元素的重数。多重集S 中重数最大的元素称为众数。  
　　  
例如，S={1，2，2，2，3，5}。  
多重集S 的众数是2，其重数为3。  
  
编程任务：  
  
　　对于给定的由n 个自然数组成的多重集S，编程计算S 的众数及其重数。  
  
数据输入：  
  
　　输入数据由文件名为input.txt 的文本文件提供。文件的第1 行多重集S 中元素个数n；接下来的n 行中，每行有一个自然数。  
  
结果输出:   
  
　　程序运行结束时，将计算结果输出到文件output.txt 中。输出文件有2 行，第1 行给出众数，第2 行是重数。  
  
输入文件示例 输出文件示例  
input.txt output.txt   
6 2   
1 3   
2   
2   
2   
3   
5

#### 算法思路

排序后二分查找到每一个数的出现次数，记录下来取最大值即可。

#### 实验程序

#include <bits/stdc++.h>  
using namespace std;  
const int maxn = 1e5;  
int a[maxn];  
int ans, ansn;  
int n;  
void solve()  
{  
 scanf("%d", &n);  
 for(int i = 1; i <= n; ++i)scanf("%d", &a[i]);  
 int l, r;  
 l = r = 1;  
 int t;  
 sort(a + 1, a + 1 + n);  
 ans = ansn = 0;  
 for(int i = 1; i <= n; ++i)  
 {  
 l = lower\_bound(a + 1, a + 1 + n, a[i]) - a - 1;  
 r = upper\_bound(a + 1, a + 1 + n, a[i]) - a - 1;  
 t = r - l;  
 if(t >= ansn)  
 {  
 ans = a[i];  
 ansn = t;  
 i = r;  
 }  
 }  
 printf("%d\n%d\n", ans, ansn);  
 return;  
}  
int main()  
{  
// freopen("input.txt", "r", stdin);  
// freopen("output.txt", "w", stdout);  
  
 solve();  
 return 0;  
}

#### 测试结果

input:   
6   
1   
2   
2   
2   
3   
5   
  
output:  
2  
3

### 整数因子分解问题

#### 题目

算法实现题 整数因子分解问题  
  
问题描述：  
  
　大于1 的正整数n 可以分解为：n=x1\*x2\*…\*xm。  
例如，当n=12 时，共有8 种不同的分解式：  
12=12；  
12=6\*2；  
12=4\*3；  
12=3\*4；  
12=3\*2\*2；  
12=2\*6；  
12=2\*3\*2；  
12=2\*2\*3 。  
  
编程任务：  
  
　　对于给定的正整数n，编程计算n 共有多少种不同的分解式。  
  
数据输入：  
  
　　由文件input.txt 给出输入数据。第一行有1 个正整数n (1≤n≤2000000000)。  
  
结果输出:   
  
将计算出的不同的分解式数输出到文件output.txt 。  
  
输入文件示例 输出文件示例  
input.txt output.txt   
　12 8

#### 算法思路

暴力递归寻找出每一个可能分解出的因式，计数即可。（对于大于1e7的某些整数这样的做法可能超过1s）

#### 实验程序

#include <bits/stdc++.h>  
using namespace std;  
typedef long long ll;  
ll ans, n;  
void solve(ll x)  
{  
 if(x == 1)  
 ++ans;  
 else  
 for(int i = 2; i <= x; ++i)  
 if(!(x % i))  
 solve(x / i);  
}  
int main()  
{  
 freopen("input.txt", "r", stdin);  
 freopen("output.txt", "w", stdout);  
  
 while(cin >> n)  
 {  
 ans = 0;  
 solve(n);  
 cout << ans << endl;  
 }  
 return 0;  
}

#### 测试结果

intput:  
12  
100  
233  
123456  
10000000  
  
output:  
8  
26  
1  
2496  
3112896