2019年

1 相隔天数

题目：输入日期格式：YYYYMMDD，求与20190205相隔的天数。

样例：

- 输入：20190208
- 输出：3

```
/*思路：把输入的日期和20190208都转换到20190101，天数分别记为count1，count2，然后再计算YYYY到2019之间相隔的天数count，最后相隔的天数就等于abs（count+count1-count2）（注意：count可能为负数）
*/
#include <iostream>
#include <cstdio>
#include <cmath>
using namespace std;

bool isLeapYear(int y){
 if ((y % 100 != 0 && y % 4 == 0) || y % 400 == 0){
  return true;
 }
 return false;
}

int main(){
 int yea[2] = {365, 366};
 int mon[2][13] = {
  {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
  {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
 };
 //year, month, day表示输入的年月日
 //count记录从year0101到20190101之间的天数
 //count1记录从yearmonthday到year0101的之间的天数
 //count2记录从20190205到20190101的之间的天数
 int year, month, day, count = 0, count1 = 0, count2 = 35;
 scanf("%4d%2d%2d", &year, &month, &day);
 //将输入的年月日转换到year0101
 while (month > 1){
  month--;
  count1 += mon[isLeapYear(year)][month];
 }
 if (day > 1){
  count1 += day - 1;
 }
 //将year转换到2019
 while (year > 2019){
  year--;
  count += yea[isLeapYear(year)];
 }

 while (year < 2019){
  count += yea[isLeapYear(year)];
  year++;
 }
 printf("%d", abs(count + count1 - count2));
 return 0;
}
```

2 最大连续子序列

题目：给定一个数字序列A1,A2…An，求i,j(1<=i<=j<=n)，使得Ai+…+Aj最大，输出这个最大和。

样例：

- 输入：6 -2 11 -4 13 -5 -2
- 输出：20

```
/*思路：动态规划求解
*/
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
const int MAXN = 10010;
int dp[MAXN], A[MAXN];

int main(){
 int n;
 cin >> n;
 for (int i = 0; i < n; i++){
  cin >> A[i];
 }
 //边界
 dp[0] = A[0];
 //状态转移方程
 for (int i = 1; i < n; i++){
  dp[i] = max(dp[i-1] + A[i], A[i]);
 }
 int k = 0;
 for (int i = 1; i < n; i++){
  if (dp[i] > dp[k]){
   k = i;
  }
 }
 cout << dp[k];
 return 0;
}
```

3 有向树形态

题目：求N个结点能够组成的二叉树的个数。

样例：

- 输入：3
- 输出：5

分析：求N个结点能够组成的二叉树的个数考察的是卡特兰数原理（如下图），所以我们只要求出下面式子的结果即可；但由于输出的数会很大，long long类型会溢出，所以只能用字符串实现大数的加减乘除。（Java类库中内置了BigInter类，所以不需要自己写）

$$Catalan_n = \frac{1}{n+1} C_{2n}^{n}$$

```
#include <cstdio>
#include <iostream>
#include <cstring>
#include <string>
using namespace std;
const int MAXN = 10000;

struct bignum{
 int d[MAXN];
 int len;
 bignum(){
  memset(d, 0, sizeof(d));
  len = 0;
 }
};

bignum change(string str){
 bignum a;
 a.len = str.size();
 for (int i = 0; i < a.len; i++){
  a.d[i] = str[a.len - i - 1] - '0';
 }
 return a;
}

bignum multi(bignum a, int b){
 bignum c;
 int carry = 0;
 for (int i = 0; i < a.len; i++){
  int temp = a.d[i] * b + carry;
  c.d[c.len++] = temp % 10;
  carry = temp / 10;
 }
 while (carry != 0){
  c.d[c.len++] = carry % 10;
  carry /= 10;
 }
 return c;
}

bignum divide(bignum a, int b, int &r){
 bignum c;
 c.len = a.len;
 for (int i = a.len - 1; i >= 0; i--){
  r = r * 10 + a.d[i];
  if (r < b) c.d[i] = 0;
  if (r > b){
   c.d[i] = r / b;
   r = r % b;
  }
 }
 while (c.len > 1 && c.d[c.len - 1] == 0){
  c.len--;
 }
 return c;
}

void print(bignum a){
 for (int i = a.len - 1; i >= 0; i--){
  printf("%d", a.d[i]);
 }
}


int main(){
 int n, r = 0;
 cin >> n;
 string str = "1";
 bignum a = change(str);
 for (int i = 2 * n; i > n; i--){
  a = multi(a, i);
 }
 for (int i = n; i >= 1; i--){
  a = divide(a, i, r);
 }
 a = divide(a, n + 1, r);
 print(a);
 return 0;
}
```

2018年

1 求众数

题目：众数就是一个序列中出现次数最多的数字。如果不唯一，则输出小的那个值。（每个数字都在int范围内）

样例：

- 输入：8

  10 3 8 8 3 2 2 2

- 输出：2

```
//用map实现
#include <iostream>
#include <cstdio>
#include <map>
using namespace std;

int main(){
 int n, num;
 cin >> n;
 map<int, int> mp;
 for (int i = 0; i < n; i++){
  cin >> num;
  mp[num]++;
 }
 int ans, max = 0;
 for (map<int, int>::iterator it = mp.begin(); it != mp.end(); it++){
```

```
  if (it->second > max){
   max = it->second;
   ans = it->first;
  }
 }
 cout << ans;
 return 0;
}
```

2 解方程

题目：给定一个字符串，代表一个 一元一次 方程。 如果有解求解，输出格式“x=数字”；如果解的个数无穷，输出 “infinite solutions”；如果没有解输出“no solution”。字符串长度不超过 256 。

样例：

- 输入：10x-2x-8=4x+7+x
- 输出：x=5

```
#include <cstdio>
#include <string>
#include <iostream>
using namespace std;
string str;
int main()
{
 cin >> str;
 int a = 0, b = 0, sym = 1, i = 0, flag = 1;//sym表示符号，flag=1表示等式左边，flag=-1表示等式右边
 while (i < str.size()){
  if (str[i] == '=') { //遇到等号时，sym置1规避之前的影响
   flag = -1;
   sym = 1;
  }
  else if(str[i]=='+') sym = 1;
  else if(str[i]=='-') sym = -1;
  else {
   int t = 0;
   while(i < str.size() && str[i] >= '0' && str[i] <= '9'){
    t = 10 * t + (str[i] - '0');
    i++;
   }
   if (i >= str.size()) {//当表达式最后是数字时单独判断，如4x-1=2x+3，否则会越界
    b -= t * sym;
    break;
   }
   if (str[i] == 'x' && t == 0) a += sym * flag; //x前系数为正负1时单独判断处理
   else if(str[i] == 'x') a += t * sym * flag; //处理系数
   else { //处理常数
    b += t * sym * flag;
    continue;
   }
  }
  i++;
 }
 if (a == 0 && b == 0) printf("infinite solutions\n");
 else if(a != 0 && b % a == 0) printf("x=%d\n", -b/a);
 else printf("no solution\n");
 return 0;
}
```

3 骨牌

题目：有 2*n 的地板，用 1*2 和 2*1 的骨牌进行铺地板。问共有多少种情况。 结果对 999983 取余。(1<=n<=10000)

样例：

- 输入：6
- 输出：12

```
//动态规划
#include <iostream>
#include <cstdio>
using namespace std;
const int MAXN = 10010;
int dp[MAXN];

int main(){
 int n;
 cin >> n;
 //边界
 dp[1] = 1;
 dp[2] = 2;
 //状态转移方程
 for (int i = 3; i <= n; i++){
  dp[i] = dp[i-1] + dp[i-2];
 }
 cout << dp[n];
 return 0;
}
```

2017年

1 中位数

题目：给定一个整数序列，求中位数。

样例：

- 输入：5

  3 2 5 3 7

- 输出：

  3

```
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
const int MAXN = 10010;
int A[MAXN];
```

```
int main(){
 int n;
 cin >> n;
 for (int i = 0; i < n; i++){
  cin >> A[i];
 }
 sort(A, A + n - 1);
 if (n % 2 == 1){
  cout << A[n/2];
 } else {
  double ans = (double)(A[n/2-1] + A[n/2]) / 2;
  cout << ans;
 }
 return 0;
}
```

2 求校验位

题目：给定一个9位数字的ISBN，求其校验位。ISBN格式为2-02-033598。校验位的计算方法为：从左到右依次将各位数字乘10,9,8,7,6,5,4,3,2；求出其和为S，做模运算得M=S mod 11。若11-M在1和9之间，校验和即为该数字；若11-M等于10，校验位为X；11-M等于11，校验位为0。输出添加校验位的ISBN，如2-02-033598-0。

样例：

- 输入：

  2-02-033598

- 输出：

  2-02-033598-0

```
#include <iostream>
#include <cstdio>
#include <string>
using namespace std;

int main(){
 string str;
 cin >> str;
 int k = 10, s = 0;
 for (int i = 0; i < str.size(); i++){
  if (str[i] >= '0' && str[i] <= '9'){
   s += (str[i] - '0') * k;
   k--;
  }
 }
 int m = s % 11;
  int M = 11 - m;
 if (M >= 1 && M <= 9){
  str += '-';
  str += ('0' + M);
 } else if (M == 10){
  str += "-X";
 } else if (M == 11){
  str += "-0";
 }
 cout << str;
 return 0;
}
```

3 最小生成树

题目：一个无向图，定点为N个，其中M条边已经给定，现在要从K条备选边中选出若干条，使得整个图连通，且选出的边权值和最小。

示例：

- 输入：

  4 4
  1 2 2
  1 4 1
  2 3 3
  3 4 4

- 输出：6

Prim算法

```
#include <cstdio>
#include <iostream>
#include <algorithm>
using namespace std;
const int MAXV = 1000; //最大顶点数
const int INF = 99999999;

int w[MAXV][MAXV];
int d[MAXV];
int vis[MAXV] = {false};

int prim(int n){
 fill(d, d + MAXV, INF);
 d[1] = 0; //顶点下标从1开始
 int ans = 0;
 for (int i = 1; i <= n; i++){
  int u = -1, MIN = INF;
  for (int j = 1; j <= n; j++){
   if (vis[j] == false && d[j] < MIN){
    u = j;
    MIN = d[j];
   }
  }
  if (u == -1) return -1;
  vis[u] = true;
  ans += d[u];
  for (int v = 1; v <= n; v++){
   if (vis[v] == false && w[u][v] != INF && w[u][v] < d[v]){
    d[v] = w[u][v];
   }
  }
 }
 return ans;
}


int main(){
```

```
 int n, m; //n个顶点，m条边
 cin >> n >> m;
 fill(w[0], w[0] + MAXV * MAXV, INF);
 for (int i = 0; i < m; i++){
  int v1, v2, weight;
  cin >> v1 >> v2 >> weight;
  w[v1][v2] = w[v2][v1] = weight;
 }
 cout << prim(n);
 return 0;
}
```

2016

1 最大公共子串长度

题目：给定两个字符串，求最大公共子串的长度。

样例：

- 输入：fdfdfd42543 232fdfdfdjlkj
- 输出：6

暴力解法，时间复杂度$O(n^3)$（可能会超时）：

```
#include <iostream>
#include <string>
using namespace std;

int main() {
 string str1, str2;
 cin >> str1 >> str2;
 int max = -1;
 for (int i = 0; i < str1.length(); i++) {
  for (int j = 1; j < str1.length() - i - 1; j++) {
   int pos = str2.find(str1.substr(i, j));
   if ((pos >= 0)&& j > max) {
    max = j;
   }
  }
 }
 cout << max << endl;
 return 0;
}
```

动态规划：

```
#include <iostream>
#include <string>
#include <cstdio>
using namespace std;
const int MAXN = 10010;
int dp[MAXN][MAXN];

int main() {
 string str1, str2;
 cin >> str1 >> str2;
 //边界
 for (int i = 0; i < str1.size(); i++){
  dp[i][0] = 0;
 }
 for (int j = 0; j < str2.size(); j++){
  dp[0][j] = 0;
 }
 //状态转移方程
 int max = -1;
 for (int i = 1; i <= str1.size(); i++){
  for (int j = 1; j <= str2.size(); j++){
   if (str1[i - 1] != str2[j - 1]){
    dp[i][j] = 0;
   } else {
    dp[i][j] = dp[i-1][j-1] + 1;
   }
   if (dp[i][j] > max){
    max = dp[i][j];
   }
  }
 }
 cout << max;
 return 0;
}
```

2 后缀序列

题目：给定一个后缀序列，要求求值，只有加减（根据题目描述，自己编的用例）

示例：

- 输入：

  23+1+

- 输出：

  6

```
#include <cstdio>
#include <iostream>
#include <string>
#include <stack>
using namespace std;

int main(){
 int temp1, temp2;
 char cur;
 string str;
 getline(cin, str);
 int i = 0;
 stack<int> st;
 while (i < str.size()){
  cur = str[i];
  if (cur >= '0' && cur <= '9') st.push(cur - '0');
  else {
```

```
    temp2 = st.top();
    st.pop();
    temp1 = st.top();
    st.pop();
    if (cur == '+') st.push(temp1 + temp2);
    else if (cur == '-') st.push(temp1 - temp2);
   }
   i++;
  }
  cout << st.top();
  return 0;
}
```

3 哈夫曼编码

题目：

给定一个字符串，求哈夫曼编码的最短长度。

样例：

- 输入：

  aaaaabbbbcccdde

- 输出：

  33

```
#include <cstdio>
#include <iostream>
#include <queue>
using namespace std;
const int MAXN = 130;
int ch[MAXN];

int main(){
 priority_queue<int, vector<int>, greater<int>> q;
 string str;
 cin >> str;
 for (int i = 0; i < str.size(); i++){
  ch[str[i]]++;
 }
 for (int i = 0; i < MAXN; i++){
  if (ch[i] != 0){
   q.push(ch[i]);
  }
 }
 int count = 0;
 while(q.size() > 1){
  int temp1 = q.top();
  q.pop();
  int temp2 = q.top();
  q.pop();
  q.push(temp1 + temp2);
  count += (temp1 + temp2);
 }
 cout << count;
 return 0;
}
```

2015

1 长方形中的正方形

题目：

给出长方形的长和宽，每次从长方形里撕去最大的正方形，输出最后能得到多少正方形？

```
#include <iostream>
using namespace std;

int squre(int X, int Y){
 int count = 0;
 while (X != Y){
  if (X > Y){
   X = X - Y;
  } else {
   Y = Y - X;
  }
  count++;
 }
 count++;
 return count;
}


int main(){
 int X, Y;
 cin >> X >> Y;
 cout << squre(X, Y);
 return 0;
}
```

2 a与b得到c

题目：

给出 a,b,c（3 个整数），判断 a,b 能否通过±*/得到c，ab可以交换位置，可以输出YES，不行输出NO

```
#include <cstdio>
#include <iostream>
using namespace std;

bool judge(long long a, long long b, long long c){
 if (a + b == c) return true;
 if (a * b == c) return true;
 if (a - b == c || b - a == c) return true;
 if (b != 0 && a % b == 0 && a / b == c) return true;
 if (a != 0 && b % a == 0 && b / a == c) return true;
 return false;
}
```

```
int main(){
 long long a, b, c;
 cin >> a >> b >> c;
 if(judge(a, b, c)) cout << "YES";
 else cout << "NO";
 return 0;
}
```

3 实现一个优先队列

题目：给出优先队列的实现，实现 4 个操作

- ADD N P：往队列里加入id为N的优先级为P的任务
- NEXT:输出下一个最高优先级的任务的id，如果优先级相同输出id小的任务，若队列中没有任务输出-1
- REMOVE N：移除id为N的任务
- COUNT：输出队列中的任务数量

示例：

- 输入：

```
10
ADD 1 1
ADD 2 3
ADD 3 2
ADD 4 3
NEXT
REMOVE 3
COUNT
NEXT
NEXT
NEXT
```

- 输出：

```
2
2
4
1
-1
```

```
#include<iostream>
#include<set>
#include<string>
using namespace std;
struct node {
 int N, P;
 bool operator<(const node& A)const {return P != A.P ? P > A.P : N < A.N;}
};
 int main() {
  int n;
  set<node>S;
  scanf("%d", &n);
  for (int i = 0; i < n; i++) {
   string op;
   cin >> op;
   if (op[0] == 'A') {
    node tmp;
    scanf("%d%d", &tmp.N, &tmp.P);
    S.insert(tmp);
   }
   else if (op[0] == 'N') {
    if (S.size() == 0)
     printf("-1\n");
    else {
     printf("%d\n", S.begin()->N);
     S.erase(S.begin());
    }
   }
   else if (op[0] == 'R') {
    int ID;
    scanf("%d", &ID);
    for (auto it = S.begin(); it != S.end(); it++) {
     if (it->N == ID) {
      S.erase(it);
      break;
     }
    }
   }
   else if (op[0] == 'C')
    printf("%d\n", S.size());
  }
  return 0;
}
```