Mid-2021Spr-两航-C期中考试

题目列表

- A 2283 求模
- B 4079 统计
- C 4076 编码
- D 2284 分解
- E 4085 座位
- F 2287 搜索
- G 2275 扫雷
- H 4082 进制
- I 4087 羊圈
- J 4144 RSA 解密初步 II

A 求模

题目分析

签到题。

只需要完成所有数的平方和即可

示例代码

```
#include<stdio.h>
typedef long long ll;
ll a[114514];
int n, i;
ll ans, x;
int main() {
    scanf("%d", &n);
    for (i = 0; i < n; ++i) scanf("%lld", &x), ans += x * x;
    printf("%lld\n", ans);
}
```

B 统计

题目分析

签到题

只需要统计平均数即可

示例代码

```c
#include<stdio.h>
typedef long long ll;
ll a[114514];
int n, i;
double sum, x, ave;
int main() {
    scanf("%d", &n);
    for (i = 0; i < n; ++i) scanf("%lf", &x), sum += x;
    printf("%.2f\n", sum / n);
}
```

C 编码

题目分析

示例代码

```c
#include<stdio.h>
#define abs(x) (x < 0 ? -x : x)
char s[12] = "!@#$%^&*()";
char trans(int a) {
    if (a >= 10 && a <= 35)  return a - 10 + 'A';
    else if (a >= 0 && a <= 9) return a + '0';
    else if (a >= -10 && a <= -1) return s[abs(a) - 1];
    else return abs(a) - 11 + 'a';
}
int n,i,x;
int main() {
    scanf("%d", &n);
    for (i = 0; i < n; ++i)scanf("%d", &x),putchar(trans(x));

}
```

D 分解

题目分析

示例代码

```c
#include<stdio.h>
#include<stdbool.h>
#define maxn 200010
int n, i;
unsigned prime[maxn], cnt;
bool vis[maxn];
bool isprime[maxn];
void get_prime(unsigned n) {
    vis[1] = 1;
    for (i = 2; i <= n; ++i) {
        if (!vis[i])prime[++cnt] = i, isprime[i] = 1;
        for (int j = 1; j <= cnt && prime[j] * i <= n; j++) {
            vis[i * prime[j]] = 1;
            if (i % prime[j] == 0)break;
        }
    }
```

```
}
int main() {
    get_prime(200000);
    scanf("%d", &n);
    for (i = 2; i <= 200000; ++i) {
        if (!(n % i) && isprime[i]) {
            printf("%d", n / i);
            break;
        }
    }
}
```

E 座位

题目分析

优先队列可以出具体占用顺序，在不要求的情况下，直接递归也可以过。

示例代码

```
#include<stdio.h>
#include<stdbool.h>
#include<string.h>
#include<stdlib.h>
#define maxn 100010
bool occupied[maxn];
typedef struct temp {
    int l, r;
}temp;
bool bigger(temp a, temp b) {
    if (a.r - a.l != b.r - b.l) return a.r - a.l < b.r - b.l;
    else return a.l > b.l;
}
temp heap[maxn];
int heap_size;
int n;
int op, x;
void perlocate_up(int lo, int hi) {
    //printf("lo is %d, hi is %d\n", lo, hi);
    //上滤操作,堆区间为[lo,hi] hi为刚刚插入的末尾
    int son = hi;
    int dad = (hi - 1) >> 1;
    while (son > 0) {
        if (!bigger(heap[dad], heap[son]))break;
        else {
            temp tmp = heap[dad];
            heap[dad] = heap[son], heap[son] = tmp;
            //swap(heap[dad], heap[son]);
            son = dad, dad = (son - 1) >> 1;
        }
    }
}
void perlocate_down(int lo, int hi) {
    //下滤操作,堆区间为[lo,hi] lo为堆的根
    int dad = lo;
    int son = (dad << 1) + 1;
    while (son <= hi) {
```

```
        if (son + 1 <= hi && bigger(heap[son], heap[son + 1]))
            son++;//如果存在双分支,选择其中较小者进行下滤
        if (!bigger(heap[dad], heap[son]))break;//此时如果堆序性正确,则整个堆都正确,直
接停机
        else {
            temp tmp = heap[dad];
            heap[dad] = heap[son], heap[son] = tmp;
            //swap(heap[dad], heap[son]);
            dad = son, son = (dad << 1) + 1;
            //调整,并且根据下滤的节点继续往下看
        }
    }
}
void insert(temp x) {
    //printf("l is %d, r is %d\n", x.l, x.r);
    heap[heap_size++] = x;
    if (heap_size > 1)perlocate_up(0, heap_size - 1);
}
void pop() {
    heap[0] = heap[--heap_size];
    heap[heap_size].l = heap[heap_size].r = 0;
    if (heap_size > 1)perlocate_down(0, heap_size - 1);
}
temp top() {
    temp ret;
    ret.l = 0, ret.r = 0;
    return heap_size > 0 ? heap[0] : ret;
}
bool unfinished() {
    temp a = top();
    //printf("l is %d r is %d\n", a.l, a.r);
    return a.r - a.l + 1 >= 3;
}
int n, i;
temp tmp;
int main() {
    scanf("%d", &n);
    tmp.l = 1, tmp.r = n;
    insert(tmp);
    while (unfinished()) {
        tmp = top();
        pop();
        int l = tmp.l, r = tmp.r;
        int length = r - l + 1, target = 0;
        if (length & 1) target = (tmp.r + tmp.l) >> 1;
        else target = ((tmp.r + tmp.l) >> 1) + 1;
        occupied[target] = 1;
        tmp.l = l, tmp.r = target - 1; insert(tmp);
        tmp.l = target + 1, tmp.r = r; insert(tmp);
    }
    for (i = 1; i <= n; ++i) if (occupied[i]) printf("%d ", i);
}
```

F 搜索

题目分析

`strstr` 函数可过

示例代码

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
char key[1010], str[1010];
char* gt;
int i;
int main() {
    scanf("%s%s", key, str);
    for (i = 0; i < strlen(key); ++i)if (isalpha(key[i]))key[i] =
tolower(key[i]);
    for (i = 0; i < strlen(str); ++i)if (isalpha(str[i]))str[i] =
tolower(str[i]);
    gt = strstr(str, key);
    if (!gt) puts("Emmm");
    else printf("Yes:%d\n", gt - str + 1);
}
```

G 扫雷

题目分析

二维数组

示例代码

```
#include<stdio.h>
#include<string.h>
char s[110][110];
int n, m, i, j;
int detect(int i, int j) {
    int ret = 0, I, J;
    for (I = -1; I <= 1; ++I)
        for (J = -1; J <= 1; ++J)
            ret += s[i + I][j + J] == 'Y';
    return ret;
}
int main() {
    scanf("%d%d", &n, &m);
    for (i = 1; i <= n; ++i) scanf("%s", s[i] + 1);
    for (i = 1; i <= n; ++i, putchar('\n'))
        for (j = 1; j <= m; ++j)
            printf("%d ", detect(i, j));
}
```

H 进制

题目分析

模拟短除法

示例代码

```
#include<stdio.h>
#include<string.h>
#define maxn 114514
const int src_radix = 10, dst_radix = 2;
char src[maxn], dst[maxn];
int _index, i, j;
int main() {
    while (scanf("%s", src) != EOF) {
        if (!strcmp(src, "0")) { puts("0.0"); continue; }
        memset(dst, 0, sizeof(dst));
        _index = 0, i = strlen(src) - 1;
        for (; i != 1; --i) {
            int num = src[i] - 48;
            j = 0;
            for (; (j < _index) || num; ++j) {
                int lala = num * dst_radix + (j < _index ? dst[j] - 48 : 0);
                dst[j] = lala / src_radix + 48;
                num = lala % src_radix;
            }
            _index = j;
        }
        dst[_index] = '\0';
        printf("0.%s", dst);
        puts(strlen(dst) ? "" : "0");
    }
}
```

I 羊圈

题目分析

判断点是否在多边形内（**不是凸包**）

示例代码

```
#include<stdio.h>
#include<math.h>
#include<stdbool.h>
#include<stdio.h>
#define maxn 100010
#define max(a, b) (a > b ? a : b)
#define min(a, b) (a < b ? a : b)
const double eps = 1e-10;

bool lessEqual(double a, double b) {// <=
    return a < b + eps;
}
```

```
bool largerEqual(double a, double b) {// >=
    return a + eps > b;
}

bool equal(double a, double b) {
    return fabs(a - b) < eps;
}
//二维点坐标
typedef struct point {
    double x;
    double y;
}point;

point ps[maxn];
int n, m;

//叉乘
//如果b在c的右手边，为正；
//b在c的左手边，为负
//a,b,c共线，0
double x_multi(point* a, point* b, point* c) {
    return (b->x - a->x) * (c->y - a->y) - (c->x - a->x) * (b->y - a->y);
}

//点乘
//夹角小于90，正
//大于90，负
//等于90，0
double dot_multi(point* a, point* b, point* c) {
    return (b->x - a->x) * (c->x - a->x) + (b->y - a->y) * (c->y - a->y);
}

//c点是否在线段ab之间
bool onSegment(point* a, point* b, point* c) {
    double max_x = max(a->x, b->x);
    double max_y = max(a->y, b->y);
    double min_x = min(a->x, b->x);
    double min_y = min(a->y, b->y);



    if (equal(x_multi(a, b, c), 0.0) && lessEqual(c->x, max_x) && largerEqual(c-
>x, min_x) && lessEqual(c->y, max_y) && largerEqual(c->y, min_y))
        return 1;
    return 0;
}
//已知a,b,c共线 看c是否在线段ab上
bool onLine(point* a, point* b, point* c) {
    double max_x = max(a->x, b->x);
    double max_y = max(a->y, b->y);
    double min_x = min(a->x, b->x);
    double min_y = min(a->y, b->y);

    if (lessEqual(c->x, max_x) && largerEqual(c->x, min_x) && lessEqual(c->y,
max_y) && largerEqual(c->y, min_y))
        return true;
    return false;
}
```

```
bool segmentIntersect(point* a, point* b, point* c, point* d) {//两条线段是否相交
    double d1 = x_multi(a, b, d);
    double d2 = x_multi(a, b, c);
    double d3 = x_multi(c, d, a);
    double d4 = x_multi(c, d, b);

    if (d1 * d2 < 0 && d3 * d4 < 0)
        return 1;

    if (equal(d1, 0.0) && onLine(a, b, d))
        return 1;
    if (equal(d2, 0.0) && onLine(a, b, c))
        return 1;
    if (equal(d3, 0.0) && onLine(c, d, a))
        return 1;
    if (equal(d4, 0.0) && onLine(c, d, b))
        return 1;
    return 0;
}
//判断点是否在多边形内
bool isInside(point* pt) {
    int count = 0;
    point end;
    end.x = 1e9, end.y = pt->y;//射线的假定端点
    //point end(1e9, pt.y);
    for (int i = 0; i < n; ++i) {
        int j = (i + 1) % n;
        if (onSegment(&ps[i], &ps[j], pt))
            return 1;//确认在多边形内部
        if (!equal(ps[i].y, ps[j].y)) {
            point low = ps[i];
            if (low.y > ps[j].y)
                low = ps[j];
            if (onSegment(pt, &end, &low))//较低端点不计算
                continue;
            if (segmentIntersect(pt, &end, &ps[i], &ps[j]))
                ++count;
            /*int tmp = -1;
            if (onSegment(pt,end,ps[i]))
                tmp = i;
            else if (onSegment(pt,end,ps[j]))
                tmp = j;
            if(tmp != -1 && equal(ps[tmp].y, max(ps[i].y,ps[j].y)))
                ++count;
            else if (tmp == -1 && segmentIntersect(ps[i],ps[j],pt,end))
                ++count;*/
        }
    }

    return count % 2 == 1;
}

int main() {
    scanf("%d", &n);
    //输入多边形的n个点
    for (int i = 0; i < n; ++i)
        scanf("%lf%lf", &ps[i].x, &ps[i].y);
```

```
    scanf("%d", &m);
    for (int i = 0; i < m; ++i) {
        point pt;
        scanf("%lf%lf", &pt.x, &pt.y);
        puts(isInside(&pt) ? "True" : "False");
    }
}
```

J RSA 解密初步 II

题目分析

大质因数分解 `Pollard-Rho` 与质数判定的 `Miller-Rabin` 模板

```
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
#include<stdbool.h>
#define maxn 100010
typedef long long ll;
typedef unsigned long long ull;
const int RM_iter = 5;//Rabin-Miller测试一般迭代5次
const int PR_iter = 107;//Pollard-Rho分解的迭代次数
//龟速乘&快速幂
int cmp(const void* p1, const void* p2) {
    ll* a = (ll*)p1, * b = (ll*)p2;
    if (*a < *b)return -1;
    else if (*a > *b) return 1;
    else return 0;
}
ll AmulBmodP(ll a, ll b, ll p) {
    ull c = (long double)a / p * b;
    ll ret = (ull)a * b - (ull)c * p;
    ret %= p;

    if (ret < 0)
        ret += p;
    return ret % p;
}
ll ApowBmodP(ll a, ll b, ll p) {
    ll ret = 1;

    while (b) {
        if (b & 1)
            ret = AmulBmodP(ret, a, p);
        a = AmulBmodP(a, a, p);
        b >>= 1;
    }
    return ret % p;
}
//质数表&埃式筛法
bool not_prime[maxn];
int prime[maxn], prime_size;
inline void get_prime() {
    for (int i = 2; i < maxn; ++i) {
        if (!not_prime[i]) {
            prime[++prime_size] = i;
```

```
            for (int j = i + i; j < maxn; j += i)
                not_prime[j] = true;
        }
    }
}
int Miller_Rabin(ll x) {
    if (x == 2)
        return 1;
    if (x <= 1 || ~x & 1)
        return 0;
    if (x < maxn)
        return !not_prime[x];
    ll s = 0, t = x - 1;
    while (~t & 1)
        s++, t >>= 1;
    srand(NULL);
    for (int i = 1; i <= RM_iter; ++i) {
        ll b = prime[rand() % prime_size + 1], k;
        b = ApowBmodP(b, t, x);
        for (int j = 1; j <= s; ++j) {
            k = AmulBmodP(b, b, x);
            if (k == 1 && b != 1 && b != x - 1)
                return 0;
            b = k;
        }
        if (b != 1)
            return 0;
    }
    return 1;
}
ll fac[1010], fac_size;//质因数的分解结果 刚得到结果的时候无序
void add_factor(ll a) {
    fac[++fac_size] = a;
}
ll gcd(ll a, ll b) {
    //特判
    if (a < 0) a = -a; if (b < 0) b = -a;
    if (a == 0) return b;
    if (b == 0) return a;
    int r = 0;//a和b的2^r形式的公因子
    while (!((a & 1) || (b & 1))) {
        //a和b都是偶数的时候
        a >>= 1; b >>= 1; r++;
    }
    ll ret = 0;
    while (1) {//首次到这里时，至少一奇
        while (!(a & 1))a >>= 1;//剔除a中的因子2
        while (!(b & 1))b >>= 1;//剔除b中的因子2
        if (a > b)a = a - b;
        else b = b - a;//简化为gcd(max(a,b)-min(a,b),min(a,b)) 可以证明这步的正确性
        if (0 == a) { ret = b << r; break; }//最后这步倒是和欧几里得做法类似
        if (0 == b) { ret = a << r; break; }
    }
    return ret;
}
//找出其中一个因子
ll Pollard_Rho(ll x, ll c) {
```

```
    ll i = 1, k = 2;
    ll x0 = rand() % (x - 1) + 1;
    ll y = x0;
    while (1) {
        ++i;
        x0 = (AmulBmodP(x0, x0, x) + c) % x;
        ll d = gcd(y - x0, x);
        if (d != 1 && d != x) return d;
        if (y == x0) return x;
        if (i == k) y = x0, k += k;
    }
}
void find_fac(ll n) {
    if (n == 1) return;
    if (Miller_Rabin(n)) { add_factor(n); return; }
    ll p = n;
    int c = PR_iter;
    while (p >= n) p = Pollard_Rho(p, c--);
    find_fac(p), find_fac(n / p);
}
inline ll read() {
    ll k = 0;// int f = 1;
    char c = getchar();
    while (c < '0' || c>'9') {
        //if (c == '-')f = -1;
        c = getchar();
    }
    while (c >= '0' && c <= '9') {
        k = (k << 1) + (k << 3) + c - 48;
        c = getchar();
    }
    return k;//*f
}
inline void write(ll x) {
    //if(x < 0)putchar('-'),x=-x;
    if (x > 9)write(x / 10);
    putchar(x % 10 + 48);
}
ll T;
ll n;
ll min_fac, max_fac;
int main() {
    //printf("gcd(-6, 12) is %lld\n", gcd(-6, 12)); ans is 6
    srand(NULL);
    get_prime();
    scanf("%lld", &T);
    while (T--) {
        scanf("%lld", &n);
        if (n == 1) { puts("NO"); continue; }
        fac_size = 0;
        find_fac(n);
        qsort(fac + 1, fac_size, sizeof(fac[0]), cmp);
        //std::sort(fac + 1, fac + fac_size + 1);
        if (fac_size == 2) printf("%lld %lld\n", fac[1], fac[2]);
        else puts("NO");
        //for (int i = 1; i <= fac_size; ++i) write(fac[i]), putchar(i ==
fac_size ? '\n' : '*');
    }
```

```
}
```