目录

tactic

在编译时加入命令`-std=c++11`并✓

排序：贪心、减少时间复杂度

比较有用的函数或小技巧

1、printf的换行搭配

printf是不带换行的

`puts（“”）；`

puts相当于一个字符串+换行

2、数组操作

数组复制

```
memcpy(backup, st, sizeof st);
```

数组整体重置为0

```
memcpy(backup, 0, sizeof backup);
```

获得字符数组长度

```
int length = strlen(ch);
```

3、单独获取一个十进制数的每一位数and将单独的数字变成十进制数

```
//单独获取一个十进制数的每一位数
while(b)
 {
     int x = b % 10;
     b /= 10;
 }

//将单独的数字变成十进制数
sum= sum * 10 + i;
```

4、异或1表示二进制的最后一位取反

```
backup[a][b] ^= 1;      //'0' = 48 = 110000B, '1' = 49 = 110001B, 即异或1表示翻转
```

5、取二进制的第i位数

```
op >> i //表示取二进制的第i位的数
```

6、用二进制数枚举开关问题的方案

```
//枚举所有方案：2 ^ 16种方案：0 ~ 2 ^ 16 - 1
for(int op = 0; op < 1 << 16; op++)
{
    //备份方案
```

```
    memcpy(backup, g, sizeof g);
    int temp = 0;

    //进行操作
    for(int i = 0; i < 4; i++)
        for(int j = 0; j < 4; j++)
            if(op >> get(i, j) & 1)
            {
                temp++;
                turn_all(i, j);
            }

    //判断是否全亮
    bool closed = false;
    for(int i = 0; i < 4; i++)
        for(int j = 0; j < 4; j++)
            if(g[i][j] == '+')
            {
                closed = true;
                break;
            }
    //该方案有效的操作
    if(!closed)
    {
        and = min(temp, ans);
    }

    //还原方案
    memcpy(g, backup, sizeof g);
}
```

7、键值对pair的一般简化操作和vector常用函数

```
//pair组一般存在vector中
#include<vector>

//将first，second简化成x，y
#define x first
#define y second

//给pair<int, int>取一个别名PII
typedef pair<int, int> PII;

int main()
{
    //定义
    vector<PII> temp;

    //插入
    temp.push_back({1, 2});

    //求数量
    temp.size();

    //判空
```

```
    temp.empty();

    //遍历and输出键值对
    for(auto op : temp) cout << op.x << op.y << endl;

    return 0;
}
```

8、输出保留几位小数

```
 while(r - l > 1e-8) //为了保证精度，一般比精度多两位
    {
        double mid = (l + r) / 2;
        if(mid * mid * mid <= x) l = mid;
        else r = mid;
    }

printf("%lf\n", r);      //保留六位小数（默认）的输出方法
printf("%.3lf\n", r);    //保留三位小数的输出方法
printf("%.4lf", r);      //保留四位小数的输出方法
```

9、sort（）用法

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

/*
sort()函数可以对给定区间所有元素进行排序。它有三个参数sort(begin, end, cmp)
其中begin为指向待sort()的数组的第一个元素的指针
end为指向待sort()的数组的最后一个元素的下一个位置的指针(加个数)
cmp参数为排序准则，cmp参数可以不写，如果不写的话，默认从小到大进行排序。如果我们想从大到小排序可以将cmp参数写为greater<int>()就是对int数组进行排序，当然<>中我们也可以写double、long、float等等。如果我们需要按照其他的排序准则，那么就需要我们自己定义一个bool类型的函数来传入。比如我们对一个整型数组进行从大到小排序：
*/
int main(){
    int num[10] = {6,5,9,1,2,8,7,3,4,0};
    sort(num,num+10,greater<int>());
    for(int i=0;i<10;i++){
        cout<<num[i]<<" ";
    }//输出结果:9 8 7 6 5 4 3 2 1 0

    return 0;
}

/*
上面我们说到sort()函数可以自定义排序准则（或重载<），以便满足不同的排序情况。使用sort()我们不仅仅可以从大到小排或者从小到大排，还可以按照一定的准则进行排序。比如说我们按照每个数的个位进行从大到小排序，我们就可以根据自己的需求来写一个函数作为排序的准则传入到sort()中。
```

```
我们可以将这个函数定义为：
*/
bool cmp(int x,int y){
    return x % 10 > y % 10;
}

/*
sort()也可以对结构体进行排序，比如我们定义一个结构体含有学生的姓名和成绩的结构体Student，然后我们按照每个学生的成绩从高到底进行排序。首先我们将结构体定义为：
*/
bool cmp_score(Student x,Student y){
    return x.score > y.score;
}

//sort对vector对象 p 排序
sort(p.begin(), p.end(), cmp);
```

10、绝对值

abs()

11、上取整

stl库里面的函数ceil（），返回的是double类型的数

或者 $\lceil \frac{a}{b} \rceil = \lfloor \frac{a+b-1}{b} \rfloor$

12、求正余数

```
int get_mod(int a, int b)   //求a除以b的正余数
{
    return (a % b + b) % b;
}
```

13、未指定读入数据个数，用sstream读入

```
#include<iostream>
#include<cstring>

using namespace std;

const int N = 10010;
```

```
int n;
int a[N];

int main()
{
    int cnt;    //行数
    cin >> cnt;

    string line;

    getline(cin, line); //忽略掉将第一行的回车
    while(cnt--)
    {
        //1、通过getline()将cin中的内容读入到字符串line中
        getline(cin, line);

        //2、通过字符串line创建一个stringstream对象ssin
        stringstream ssin(line);

        //3、ssin右移计入到a[]中，n计数
        while(ssin >> a[n]) n++;
    }
}
```

14、判断日期是否合法（含平闰年判断）

```
//闰年的2月有29天，  平年的2月有28天

//闰年判断的两种条件
//1、是4的倍数且不是100的倍数
year % 4 == 0 && year % 100 != 0
//2、是400的倍数
year % 400 == 0
```

```
//日期问题的重要函数
//1, 3, 5, 7, 8, 10, 腊
int months[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int is_leap(int y)
{
    if((y % 4 == 0 && y % 100 != 0) || y % 400 == 0) return 1;

    return 0;
}

int get_month_days(int y, int m)
{
    if(m == 2) return 28 + is_leap(y);
    return months[m];
}
```

```cpp
#include<iostream>
#include<cstring>
#include<algorithm>

using namespace std;

int days[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}; //每个月的天数

bool check(int date)
{
    int year = date / 10000;
    int month = date % 10000 / 100;
    int day = date % 100;

    //默认年份合法，先判断月，再判断日（找出不合法的情况）
    if(month == 0 || month > 12) return false;  //月等于0 或 大于12 则不合法

    if(day == 0 || month != 2 && day > days[month]) return false;   //日等于0 或在不是二月的情况下，日大于该月的天数 则不合法

    if(month == 2)  //如果是2月，天数是否合法，主要是判断平闰年
    {
        int leap = year % 4 == 0 && year % 100 || year % 400 == 0;  //两种判断方法，有一个满足就是闰年，即leap=1；不是闰年则leap=0
        if(day > 28 + leap) return false;
    }

    return true;
}

int main()
{
    int date;
    cout << "请输入一个8位数，前四位数表示年份，后四位表示日期" << endl;
    cin >> date;
    if(check(date)) cout << "该日期合法" << endl;

    return 0;
}
```

15、格式化输入输出

```cpp
//scanf和printf在格式化输入输出时具有强大功能
//例如:要求输入的格式为A/B/C，输出的格式为A-B-C（B和C不足两位时前面用0补齐）

#include<iostream>
#include<cstdio>
#include<cstring>

using namespace std;

int main()
{
    int a, b, c;
    scanf("%d/%d/%d", &a, &b, &c);
```

```
    printf("%d%02d%02d", a, b, c);

    return 0;
}

//一直读直到结束
while(~scanf("%04d%02d%02d\n%04d%02d%02d", &y1, &m1, &d1, &y2, &m2, &d2))
```

16、强大的字符串解析工具sscanf(const char *str, const char *format, ...)

```
//第一个参数是字符数组，因为c中没有string，所以要通过c_str()将string转换成字符数组
//第二个参数是形式
//其余参数是对应变量

//该函数本身会返回一个int，表示成功转换了几个参数
```

```
//输入：17:48:19 21:57:04
//输出：17-48-19 21-57-04 0
#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>

using namespace std;

int main()
{
    string line;
    getline(cin, line);

    //统一格式
    if(line.back() != ')') line += " (+0)";

    //解析字符串里面的数据
    int h1, m1, s1, h2, m2, s2, d;
    sscanf(line.c_str(), "%d:%d:%d %d:%d:%d (+%d)", &h1, &m1, &s1, &h2, &m2,
&s2, &d);

    //使用
    printf("%02d-%02d-%02d %02d-%02d-%02d %d", h1, m1, s1, h2, m2, s2, d);

}
```

16、int的无穷大

```
//在最值问题中常常要将数组值初始化为无穷大
memset(f, -0x3f, sizeof f);
```

17、余数取正

```
(x % k + k) % k
```

18、平方，开方，绝对值

```
#include<cmath>

int a = pow(4, 2);       //4的平方
int b = pow(4, 0.5);     //根号4
int c = sqrt(4);         //根号4
int d = abs(b - c);      //整数的绝对值
int e = fabs(b - c);     //浮点数的绝对值
```

二分

整数二分步骤

二分问题看似很复杂，其实背好两种模板就会简化很多，重点是判断条件

1、找一个区间**[L ,R]**，使得答案一定在该区间中（找到**二段性**）

2、找一个**判断条件**，使得该判断条件具有二段性，并且答案一定是该二段性的分界点(答案满足判断条件)

3、分析中点M在该判断条件下是否成立，如果成立，考虑答案在哪个区间；如果不成立，考虑答案在哪个区间（**问自己mid在目标值的左边还是右边，左边先左再加一，右边先右不处理**）

4、如果更新方式为R=M，L=M+1则不用做任何处理；如果更新方式为L = M，R=M-1则需要在计算M时加上1

整数二分模板题---789.数的范围

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<algorithm>

using namespace std;

const int N = 100010;

int n, q, k;
int a[N];

bool check1(int x)
{
    if(x >= k) return true;
    return false;
}

bool check2(int x)
{
```

```
    if(x <= k) return true;
    return false;
}

int main()
{
    cin >> n >> q;
    for(int i = 0; i < n; i++) scanf("%d", &a[i]);

    while(q--)
    {
        scanf("%d", &k);
        int l = 0, r = n - 1;
        while(l < r)
        {
            int mid = (l + r) >> 1;
            if(check1(a[mid])) r = mid;
            else l = mid + 1;
        }
        if(a[l] != k) printf("-1 -1\n");
        else
        {
            printf("%d ", l);
            r = n - 1;
            while(l < r)
            {
                int mid = (l + r + 1) >> 1;
                if(check2(a[mid])) l = mid;
                else r = mid - 1;
            }
            printf("%d\n", l);
        }
    }

    return 0;
}
```

浮点数二分模板题---790.数的三次方根

与整数区别：

1、`while(r - 1 > eps)` 判断条件是大于精度小两个数量级

2、`double mid = (l + r) / 2` 不用管+1，必须用double、/2

3、else 后面不用+1或-1

```
#include<iostream>
#include<algorithm>

using namespace std;

const double eps = 1e-8;

double n;
bool fa;
```

```
int main()
{
    cin >> n;

    if(n < 0) fa = true;
    else fa = false;
    n = abs(n);

    double l = 0, r = 22;
    while(r - l > eps)
    {
        double mid = (l + r) / 2;
        if(mid * mid * mid >= n) r = mid;
        else l = mid;
    }
    if(fa) cout << "-";
    printf("%.6f", l);

    return 0;
}
```

1227.分巧克力

```
#include<iostream>
#include<algorithm>

#define x first
#define y second

using namespace std;

const int N = 100010;

typedef pair<int, int> PII;
typedef long long LL;

int n, k;
PII p[N];

bool check(int mid)
{
    LL res = 0;
    for(int i = 0; i < n; i++)
    {
        res += (LL)(p[i].x / mid) * (p[i].y / mid);
        if(res >= k) return true;
    }
    return false;
}

int main()
{
    cin >> n >> k;
    int x, y;
    for(int i = 0; i < n; i++)
    {
        scanf("%d%d", &x, &y);
```

```
        p[i] = {x, y};
    }

    int l = 1, r = N;
    while(l < r)
    {
        int mid = (l + r + 1) >> 1;
        if(check(mid)) l = mid;
        else r = mid - 1;
    }

    cout << l;

    return 0;

}
```

总结

背好模板，注意细节，能反应过来能用二分做就没问题

前缀和

一维前缀和模板题---795.前缀和

```
s[i] = s[i - 1] + a[i];

a[l] + ... + a[r] = S[r] - S[l - 1]
```

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>

using namespace std;

const int N = 100010;

int n, m;
int a[N], s[N];

int main()
{
    cin >> n >> m;

    for(int i = 1; i <= n; i++)
    {
        scanf("%d",&a[i]);
        s[i] = s[i - 1] + a[i];
    }

    while(m--)
    {
```

```
        int l, r, res;
        scanf("%d%d", &l, &r);
        res = s[r] - s[l - 1];
        printf("%d\n", res);
    }

    return 0;
}
```

二维前缀和模板题---796.子矩阵的和

```
s[i][j] = s[i - 1][j] + s[i][j - 1] - s[i - 1][j - 1] + a[i][j];

//以(x1, y1)为左上角，(x2, y2)为右下角的子矩阵的和为：
S[x2, y2] - S[x1 - 1, y2] - S[x2, y1 - 1] + S[x1 - 1, y1 - 1];
```

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>

using namespace std;

const int N = 1010;

int n, m, q;
int a[N][N], s[N][N];

int main()
{
    cin >> n >> m >> q;

    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= m; j++)
        {
            scanf("%d", &a[i][j]);
            s[i][j] = s[i][j-1] + s[i-1][j] - s[i-1][j-1] + a[i][j];
        }

    while(q--)
    {
        int x1, x2, y1, y2;
        scanf("%d%d%d%d", &x1, &y1, &x2, &y2);
        printf("%d\n", s[x2][y2] - s[x2][y1-1] - s[x1-1][y2] + s[x1-1][y1-1]);
    }

    return 0;
}
```

1230.K倍区间

```
#include<iostream>

using namespace std;
```

```
typedef long long LL;

const int N = 100010;

int n, k;
LL cnt[N], s[N];  //前缀和余数相同的，相减即可满足K倍区间

int main()
{
    cin >> n >> k;
    for(int i = 1; i <= n; i++)
    {
        scanf("%lld", &s[i]);
        s[i] += s[i - 1];
    }

    LL res = 0;
    cnt[0] = 1;
    for(int i = 1; i <= n; i++)
    {
        res += cnt[s[i] % k];
        cnt[s[i] % k]++;
    }

    cout << res;

    return 0;

}
```

差分

一维差分模板题---797.差分

给区间[l, r]中的每个数加上c：b[l] += c, b[r + 1] -= c;

初始化b[]：insert(i, i, a[i]);

```
#include<iostream>

using namespace std;

const int N = 100010;

int n, m;
int a[N], b[N];

void insert(int l, int r, int c)
{
    b[l] += c;
    b[r + 1] -= c;
}

int main()
{
```

```
    cin >> n >> m;
    for(int i = 1; i <= n; i++)
    {
        scanf("%d", &a[i]);
        insert(i, i, a[i]);
    }

    while(m--)
    {
        int l, r, c;
        scanf("%d%d%d", &l, &r, &c);
        insert(l, r, c);
    }

    for(int i = 1; i <= n; i++)
    {
        a[i] = a[i - 1] + b[i];
        printf("%d ", a[i]);
    }

    return 0;
}
```

二维差分模板题---798.差分矩阵

给以(x1, y1)为左上角，(x2, y2)为右下角的子矩阵中的所有元素加上c：
b[x1, y1] += c; b[x1][y2 + 1] -= c; b[x2 + 1][y1] -= c; b[x2 + 1][y2 + 1] += c;

初始化b[]: insert(i, j, i, j, a[i]);

```
#include<iostream>

using namespace std;

const int N = 1010;

int n, m, q;
int a[N][N], b[N][N];

void insert(int x1, int y1, int x2, int y2, int c)
{
    b[x1][y1] += c;
    b[x1][y2 + 1] -= c;
    b[x2 + 1][y1] -= c;
    b[x2 + 1][y2 + 1] += c;
}

int main()
{
    cin >> n >> m >> q;
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= m; j++)
        {
            scanf("%d", &a[i][j]);
            insert(i, j, i, j, a[i][j]);
        }
```

```
    while(q--)
    {
        int x1, y1, x2, y2, c;
        scanf("%d%d%d%d%d", &x1, &y1, &x2, &y2, &c);
        insert(x1, y1, x2, y2, c);
    }

    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j <= m; j++)
        {
            a[i][j] = a[i - 1][j] + a[i][j - 1] - a[i - 1][j - 1] + b[i][j];
            printf("%d ", a[i][j]);
        }
        puts("");
    }

    return 0;
}
```

5396.棋盘

```
#include<iostream>

using namespace std;

const int N = 2010;

int n, m;
int a[N][N], b[N][N];

void insert(int x1, int y1, int x2, int y2, int c)
{
    b[x1][y1] += c;
    b[x1][y2 + 1] -= c;
    b[x2 + 1][y1] -= c;
    b[x2 + 1][y2 + 1] += c;
}

int main()
{
    cin >> n >> m;
    while(m--)
    {
        int x1, y1, x2, y2;
        scanf("%d%d%d%d", &x1, &y1, &x2, &y2);
        insert(x1, y1, x2, y2, 1);
    }

    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j <= n; j++)
        {
            a[i][j] = a[i - 1][j] + a[i][j - 1] - a[i - 1][j - 1] + b[i][j];
            if(a[i][j] % 2 == 0) cout << "0";
            else cout << "1";
```

```
        }
        puts("");
    }

    return 0;
}
```

4655.重新排序

```
#include<iostream>
#include<algorithm>

using namespace std;

typedef long long LL;

const int N = 100010;

int n, m;
int a[N], b[N];

int main()
{
    cin >> n;
    for(int i = 1; i <= n; i++) scanf("%d", &a[i]);

    cin >> m;
    while(m--)
    {
        int l, r;
        scanf("%d%d", &l, &r);
        b[l]++, b[r + 1]--;
    }

    for(int i = 1; i <= n; i++) b[i] += b[i - 1];

    LL sum1 = 0;
    for(int i = 1; i <= n; i++) sum1 += (LL)b[i] * a[i];

    sort(a + 1, a + n + 1);
    sort(b + 1, b + n + 1);
    LL sum2 = 0;
    for(int i = 1; i <= n; i++) sum2 += (LL)b[i] * a[i];

    cout << sum2 - sum1;

    return 0;
}
```

日期问题

```
//闰年判断的两种条件
```

```
//1、是4的倍数且不是100的倍数
year % 4 == 0 && year % 100 != 0
//2、是400的倍数
year % 400 == 0

//日期问题的重要函数
//1, 3, 5, 7, 8, 10, 腊
int months[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int is_leap(int y)
{
    if((y % 4 == 0 && y % 100 != 0) || y % 400 == 0) return 1;

    return 0;
}

int get_month_days(int y, int m)
{
    if(m == 2) return 28 + is_leap(y);
    return months[m];
}
```

1229.日期问题

```
#include<iostream>
#include<cstring>
#include<algorithm>

using namespace std;

int months[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int is_leap(int y)
{
    if(y % 4 == 0 && y % 100 != 0 || y % 400 == 0) return 1;
    return 0;
}

bool is_valid(int y, int m, int d)
{
    if(m <= 0 || m > 12) return false;

    if(m != 2)
    {
        if(d <= 0 || d > months[m]) return false;
    }
    else
    {
        if(d <= 0 || d > months[m] + is_leap(y)) return false;
    }
    return true;
}

int main()
{
    int a, b, c;
    scanf("%d/%d/%d", &a, &b, &c);
```

```
    for(int date = 19600101; date <= 20591231; date++)
    {
        int year = date / 10000, month = date % 10000 / 100, day = date % 100;
        if(is_valid(year, month, day))
        {
            if(((year % 100) == a && month == b && day == c)
            || (month == a && day == b && (year % 100) == c)
            || (day == a && month == b && (year % 100) == c))
                printf("%d-%02d-%02d\n", year, month, day);
        }
    }
    return 0;
}
```

466.回文日期

```
#include<iostream>

using namespace std;

int months[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int is_leap(int year)
{
    if(year % 4 == 0 && year % 100 != 0 || year % 400 == 0) return 1;
    return 0;
}

bool is_valid(int date)
{
    int y = date / 10000, m = date % 10000 / 100, d = date % 100;

    if(m <= 0 || m > 13) return false;
    if(d <= 0 || m != 2 && d > months[m]) return false;

    if(d > months[m] + is_leap(y)) return false;

    return true;
}

int main()
{
    int date1, date2;
    cin >> date1 >> date2;

    int res = 0;
    for(int i = 1000; i < 10000; i++)
    {
        int date = i, x = i;
        for(int j = 0; j < 4; j++) date = date * 10 + x % 10, x /= 10;
        if(date >= date1 && date <= date2 && is_valid(date)) res++;
    }
    cout << res;

    return 0;
}
```

3218.日期计算

```
#include<iostream>

using namespace std;

int months[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int is_leap(int year)
{
    if(year % 4 == 0 && year % 100 != 0 || year % 400 == 0) return 1;
    return 0;
}

int get_days(int year, int month)
{
    if(month != 2) return months[month];
    else return months[month] + is_leap(year);
}

int main()
{
    int year, d;
    cin >> year >> d;

    for(int i = 1; i <= 12; i++)
        for(int j = 1; j <= get_days(year, i); j++)
        {
            d--;
            if(d == 0) printf("%d\n%d", i, j);
        }

    return 0;
}
```

dfs

树与图的存储---邻接表

```
// 对于每个点k，开一个单链表，存储k所有可以走到的点。h[k]存储这个单链表的头结点
int e[N], ne[N], h[N], idx;
//e[i]表示编号为i的节点的数值，ne[i]表示编号i的节点能走到的节点编号，h[i]表示数值为i的节点的编号，idx表示节点编号

// 添加一条边a->b
void add(int a, int b)
{
    e[idx] = b, ne[idx] = h[a], h[a] = idx++;
}

// 初始化
idx = 0;
memset(h, -1, sizeof h);
```

dfs模板

```
int dfs(int u)
{
    st[u] = true; // st[u] 表示点u已经被遍历过

    for (int i = h[u]; i != -1; i = ne[i])  //遍历与u相连的所有点，u是节点数值，i是节点编号
    {
        int j = e[i];
        if (!st[j]) dfs(j);
    }
}
```

1207.大臣的旅费

```
#include<iostream>
#include<vector>

using namespace std;

const int N = 100010;

struct edge
{
    int id, w;
};
vector<edge> h[N];
int dist[N];
int n;

void dfs(int u, int father, int distance)
{
    dist[u] = distance;
    for(auto node : h[u])
    {
        if(node.id != father) dfs(node.id, u, distance + node.w);
    }
}

int main()
{
    cin >> n;
    int a, b, c;
    for(int i = 0; i < n - 1; i++)
    {
        scanf("%d%d%d", &a, &b, &c);
        h[a].push_back({b, c});
        h[b].push_back({a, c});
    }

    dfs(1, -1, 0);

    int u = 1;
```

```
    for(int i = 1; i <= n; i++)
        if(dist[i] > dist[u]) u = i;

    dfs(u, -1, 0);
    for(int i = 1; i <= n; i++)
        if(dist[i] > dist[u]) u = i;

    int s = dist[u];
    cout << (long long)s * 10 + (long long)(1 + s) * s / 2;

    return 0;
}
```

bfs

bfs模板

```
queue<int> q;
st[1] = true; // 表示1号点已经被遍历过
q.push(1);

while (q.size())
{
    int t = q.front();
    q.pop();

    for (int i = h[t]; i != -1; i = ne[i])
    {
        int j = e[i];
        if (!st[j])
        {
            st[j] = true; // 表示点j已经被遍历过
            q.push(j);
        }
    }
}
```

stl

```
vector，变长数组，倍增的思想
    size()  //返回元素的个数
    empty() //返回是否为空
    clear() //清空
    front()
    back()
    push_back()
    pop_back()
    begin()
    end()
    []
    支持比较运算，按字典序
```

```
pair<int, int>
    first   //第一个元素
    second  //第二个元素
    支持比较运算，以first为第一关键字，以second为第二关键字（字典序）

string，字符串
    size()  /   length()    //返回字符串的长度
    empty()
    clear()
    substr(起始下标，（字串长度）)      //返回子串
    c_str() //返回字符串所在的字符数组的起始地址

queue，队列
    size()
    empty()
    push()
    pop()
    front()
    back()

stack()，栈
    size()
    empty()
    push()
    pop()
    top()   //返回栈顶元素
```

数学问题

欧几里得算法---求最大公约数

```
int gcd(int a, int b)
{
    return b ? gcd(b, a % b) : a;
}
```

筛法求素数

```
#include<iostream>

using namespace std;

const int N = 100010;

int primes[N], cnt;
bool st[N];

void get_primes(int n)
```

```
{
    for(int i = 2; i <= n; i++)
    {
        if(!st[i]) primes[cnt++] = i;
        for(int j = 0; primes[j] * i <= n; j++)
        {
            st[primes[j] * i] = true;
            if(i % primes[j] == 0) break;
        }
    }
}

int main()
{
    int n;
    cin >> n;
    get_primes(n);

    for(int i = 0; i < cnt; i++) cout << primes[i] << endl;

    return 0;
}
```

排序问题

归并排序

```
#include<iostream>

using namespace std;

const int N = 100010;

int n;
int q[N], temp[N];

void merge_sort(int q[], int l, int r)
{
    if(l >= r) return;
    int mid = (l + r) >> 1;
    merge_sort(q, l, mid);
    merge_sort(q, mid + 1, r);

    int i = l, j = mid + 1, k = 0;
    while(i <= mid && j <= r)
        if(q[i] < q[j]) temp[k++] = q[i++];
        else temp[k++] = q[j++];
    while(i <= mid) temp[k++] = q[i++];
    while(j <= r) temp[k++] = q[j++];

    for(i = l, j = 0; i <= r; i++, j++) q[i] = temp[j];

    return;
}

int main()
```

```
{
    cin >> n;
    for(int i = 0; i < n; i++) scanf("%d", &q[i]);

    merge_sort(q, 0, n - 1);

    for(int i = 0; i < n; i++) printf("%d ", q[i]);

    return 0;
}
```

按从大到小的顺序排序

```
sort(a, a + n, greater<int>());
```