算法学习笔记（可能会记）

一、库函数

1.algorithm库

（1）lower_bound(Iter_1, Iter_2, val) :

二分搜索函数，前提是有序的情况下，lower_bound 返回第一个大于等于val值的位置。使用的范围是
(Iter**1**,**Iter**2),包括Iter_1*到Iter_2*位置中的所有元素。其返回值是一个**迭代器**，指向第一个大于等于val的位置。

upper_bound 返回第一个大于val值的位置（少了等于）。

此外，与sort()一样，可以自定义cmp比较函数...还没学...

（2）sort(begin, end, bool cmp):

一个排序函数，大多情况下比自己写的排序函数快。

其中begin为指向待排的数组的第一个元素的指针，end为指向待排的数组的最后一个元素的**下一个位置**的指针，cmp参数为排序准则，如果不写的话，默认从小到大进行排序。

cmp参数写为`greater<int>()`就是对int数组进行从大到小排序。

（3）unique(begin, end, bool cmp):

一个伪去重函数，遍历容器，将重复的元素移动至容器尾，所以一般搭配.resize()函数去除尾部`a.resize(distance(a.began(),it))`。也可用
`a.erase(unique(a.begin(),a.end()),a.end())`

返回值为去重后的尾地址。

（4）reverse(begin,end):

翻转数组、字符串和向量。

（5）count(begin，end，‘a’):

统计区间内出现的字符的个数。

（6）gcd(x,y)、lcm(x, y):

求最大公约数、求最小公倍数。

lcm(x,y) = x*y/gcd(x,y)

2.容器函数

（1）priority_queue（优先队列）:

优先队列是一种特殊的队列，它允许我们快速访问队列中具有最高（或最低）优先级的元素。此队列顶部元素总是具有最大的值。

empty(): 检查队列是否为空；size(): 返回队列中的元素数量。top(): 返回队列顶部的元素（不删除它）；push(): 向队列添加一个元素；pop(): 移除队列顶部的元素。

头文件`<functional>`中，包含less和greater，

大顶堆（默认值）：`priority_queue<int,vector<int>,less<int>>`

小顶堆：`priority_queue<int,vector<int>,greater<int> >`

或使用构造法：

```
struct compare {
    bool operator()(int a, int b) {
        return a > b; // 定义最小堆
    }
};
priority_queue<int, vector<int>, compare> pq_min;
```

3.memory.h库

（1）memset(结构体/数组名s , 用于替换的字符ch , 前n个字符):

char型初始化函数，将s中的前n个字节用ch替换并且返回s，在一段内存块中填充某一个给定的值，常用于较大的对结构体和数组的清零操作。

例如，将数组初始化为-1: `memset(a,-1,sizeof(a));`

4.string库

（1）atoi(string s):

将字符串转换为对应的整数

（2）substr(pos, len):

复制字符串，从指定位置开始，复制指定长度

（3）find(target,pos);

寻找目标返回出现的第一个索引，若没找到，则返回`string::npos`。

4.math.h/cmath库

（1）ceil(int n):

向上取整函数，返回不小于自变量的最小整数

二、基本概念

1.迭代器

用来遍历容器的，底层是指针，通过解地址“*iter”，来访问一个迭代器指向的元素。

2.前缀和、差分

前缀和求差分可获得原数组，差分求前缀和可获得原数组
（1）前缀和可用于快速查询区间和
（2）差分可用于快速区间修改(离线)，先修改、再求前缀和（变为原数组）、再查询，
d[l]+=x,d[r+1]-=x;

三、常用算法模版（如果有）

1.二分查找

注意：二分答案的判断条件(check()函数)，必须检测某一区间符合并且另一区间不符合，如分界点左边符合条件，右边不符合条件，不能出现中间一段单独符合。

（1）版本1

```
int bsearch_1(int l, int r)
{
    while (l < r)
    {
        int mid = l + r >> 1;
        if (check(mid)) r = mid;
        else l = mid + 1;
    }
    return l;
}
```

（2）版本2

```
int bsearch_2(int l, int r)
{
    while (l < r)
    {
        int mid = l + r + 1 >> 1;
        if (check(mid)) l = mid;
        else r = mid - 1;
    }
    return l;
}
```

2.判断质数

（1）直观法

```
bool isprime(int a) {
if (a == 1)
    return false;
for (int i = 2; i * i <= a; i++)
    if (a % i == 0)
    return false;
return true;
```

```
}
```

3.对角线的表示

从左上到右下的对角线满足 x-y 都相等。

从左下到右上的对角线满足 x+y 都相等。

4.判断是否为 2 的 N 次方

（1）位运算：(a > 0) && (a & (a - 1)) == 0

```
if ((a > 0) && (a & (a - 1)) == 0)
        printf("a 是 2 的幂次");
    else
        printf("a 不是 2 的幂次");
    printf("\n");
```

（2）循环除2

```
void judge(int b)
{
    while (!(b % 2))
    {
        b = b / 2;
        if (b == 1)
        {
            printf("b 是 2 的幂次");
            return;
        }
    }
    printf("b 不是 2 的幂次");
    return;
}
```

5.快速幂

时间复杂度O(logn)

```
long long binpow(long long a, long long b) {  //底数 指数
    long long res = 1;
    while (b > 0) {
        if (b & 1) res = res * a;
        a = a * a;
        b >>= 1;
    }
    return res;
}
```

运算中取模:

```
long long binpow(long long a, long long b, long long m) {  //底数 指数 模数
    a %= m;
    long long res = 1;
    while (b > 0) {
        if (b & 1) res = res * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return res;
}
```

四、错误分析

1.bfs

（1）大数据遇到MLE时，可能是标记数组逻辑有问题，出现了同一个点多次入队的情况，注意在每个点入队之前就将其标记，而不是在出队后再标记。