算法笔记@蔚蓝星辰mic

1.位运算

快速幂（取模）a^b mod k

```
#include <iostream>
#define IO ios::sync_with_stdio(false);cin.tie(0);cout.tie(0)
#define ll long long
using namespace std;
ll fastpow(ll base,ll pow,ll m){
    ll ans =1;
    while(pow){
        if(pow & 1)   //如果pow是奇数
            ans = ans * base %m;
        base = base * base %m;
        pow >>= 1;   //相当于pow/=2
    }
    return (ans %m);
}
int main(){
    IO;
    int base,pow,m;
    cin>>base>>pow>>m;
    cout<<fastpow(base,pow,m);
    return 0;
}
```

64位整数乘法 a*b mod k

```
#include <iostream>
#define IO ios::sync_with_stdio(false);cin.tie(0);cout.tie(0)
#define ll long long
#define ull unsigned long long
using namespace std;
ll fastpow(ull base,ull pow,ull m){
    ull ans =0;
    while(pow){
        if(pow & 1)
            ans = (ans + base) %m;
        base = base * 2 %m;
        pow >>= 1;
    }
    return ans;
}
int main(){
```

```
    IO;
    ull base,pow,m;
    cin>>base>>pow>>m;
    cout<<fastpow(base,pow,m);
    return 0;
}
```

2.STL

sort

> #include <algorithm>
> sort(a,a+n,cmp) :对容器或普通数组中 [first, last) 范围内的元素进行排序，默认进行升序排序。
> （序的数组的起始地址，结束的地址，[排序的方法]）
> 比较函数： less<int>() greater<int>()
> int cmp(int a,int b){return a<b;}

```
#include <iostream>
#include <algorithm>
#define IO ios::sync_with_stdio(false);cin.tie(0);cout.tie(0)
using namespace std;

int a[100005];

int main(){
    IO;
    int n;
    cin>>n;
    for(int i=0;i<n;i++){
        cin>>a[i];
    }
    sort(a,a+n,less<int>());
    for(int i=0;i<n;i++){
        cout<<a[i]<<" ";
    }
    return 0;
}
```

stable_sort

> stable_sort() 函数完全可以看作是 sort() 函数在功能方面的升级版。换句话说，stable_sort() 和 sort() 具有相同的使用场景，就连语法格式也是相同的，只不过前者在功能上除了可以实现排序，还可以保证不改变相等元素的相对位置。

lower_bound

uppor_bound

Vector

> 动态数组，从末尾能够快速插入与删除，直接访问如何元素

C++ Vectors

Vectors 包含着一系列连续存储的元素，其行为和数组类似。访问Vector中的任意元素或从末尾添加元素都可以在常量级时间复杂度内完成，而查找特定值的元素所处的位置或是在Vector中插入元素则是线性时间复杂度。

Constructors	构造函数
Operators	对vector进行赋值或比较
assign()	对Vector中的元素赋值
at()	返回指定位置的元素
back()	返回最末一个元素
begin()	返回第一个元素的迭代器
capacity()	返回vector所能容纳的元素数量(在不重新分配内存的情况下)
clear()	清空所有元素
empty()	判断Vector是否为空（返回true时为空）
end()	返回最末元素的迭代器(译注:实指向最末元素的下一个位置)
erase()	删除指定元素
front()	返回第一个元素
get_allocator()	返回vector的内存分配器
insert()	插入元素到Vector中
max_size()	返回Vector所能容纳元素的最大数量（上限）
pop_back()	移除最后一个元素
push_back()	在Vector最后添加一个元素
rbegin()	返回Vector尾部的逆迭代器
rend()	返回Vector起始的逆迭代器
reserve()	设置Vector最小的元素容纳数量
resize()	改变Vector元素数量的大小
size()	返回Vector元素数量的大小
swap()	交换两个Vector

queue

> 队列先进先出