模板

滑动数组

```
void huadongmin(int a[],int n,int k){
    deque<int> q;
    for(int i=1;i<=n;i++){
        while(!q.empty()&&a[i]<q.back()) q.pop_back();
        q.push_back(a[i]);
        if(i>=k+1) if(q.front()==a[i-k]) q.pop_front();
        if(i>=k) cout<<q.front()<<" ";
    }
    q.clear();
}
void huadongmax(int a[],int n,int k){
    deque<int> q;
    for(int i=1;i<=n;i++){
        while(!q.empty()&&a[i]>q.back()) q.pop_back();
        q.push_back(a[i]);
        if(i>=k+1) if(q.front()==a[i-k]) q.pop_front();
        if(i>=k) cout<<q.front()<<" ";
    }
    q.clear();
}
```

并查集

```


void inti_set(int size)
{
    for (int i = 1; i <= size; i++){
        uset[i] = i;
        rank[i] = 0;
    }
}
int find_set(int i){
    if (i == uset[i]){
        return i;
    }
    return uset[i] = find(uset[i]);
}

void unite(int x, int y){
    int x = find(x);
    int y = find(y);
    if (x == y){
        return;
    }
```

```
        if (rank[x] < rank[y]){
            uset[x] = y;
        }
        else{
            uset[y] = x;
            if (rank[x] == rank[y]){
                rank[x]++;
            }
        }
    }
```

二分查找 返回第一个大于等于X的下标

```
lower_bound(a.begin(),a.end(),x)-a;
//二分查找 返回第一个大于X的下标
upper_bound(a.begin(),a.end(),x)-a;


double pi=acos(-1.0);//pai的大小
double asin (double); //结果介于[-PI/2, PI/2]
double acos (double); //结果介于[0, PI]
double atan (double); //反正切(主值), 结果介于[-PI/2, PI/2]
double atan2 (double, double); //反正切(整圆值), 结果介于[-PI/2, PI/2]
double ceil (double);  //取上整
double floor (double); //取下整

//全排函数
while(next_permutation(str.begin(),str.end()))
    cout<<str<<endl;

//素数筛
int primes[N],cnt;
bool book[N];
void Primes(int n){
    memset(book,false,sizeof(book));

    for(int i=2;i<=n;i++){
        if(!book[i])
            primes[cnt++]=i;

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

滚动前缀和 某一节点的加减

```
int lowbit(int x) { return x & -x; }

void init() {
    for (int i = 1; i <= n; i++) {
        c[i] = pre[i] - pre[i - lowbit(i)];
    }
}
void add(int x, int y) {
    for (; x <= N; x += lowbit(x)) {
        c[x] += y;
    }
}
int ask(int x) { // 查询a序列的 [1,x] 中所有数的和
    int ans = 0;
    for (; x; x -= lowbit(x)) {
        ans += c[x];
    }
    return ans;
}
```

快速幂

```
ull quick_pow(ull a,ull b)
{
    if(b==0) return 1;
    ull res=quick_pow(a,b>>1);
    if(b&1) return res*res*a;
    return res*res;
}
```

二分查找

```
bool check(int x){
    //判断X是否满足某种性质
}
int search (int l,int r){
    //浮点数
    /*
        const double eps=1e-6;
        while(r-l>eps)
    */
    while(l<r){
        int mid=l+r>>1;
        if(check(mid)) r=mid;
        else l=mid+1;
    }
    return l;
}
```

进制转换（n进制转m进制）

```
#include<cstdio>
const int N=1e5+3;
int n,m,x,tmp,ansl;
char a[N],ans[N];
inline int n_to_ten() {
    int x=0;
    for(int i=0; a[i]; ++i) {
        x*=n;
        if (a[i]>'A'&&a[i] <='F')
            x+=(a[i]-'A'+10);
        else
            x+=(a[i]-'0');
    }
    return 0;
}
inline void ten_to_m(){
    do{
        tmp=x%m;
        if(tmp<10)ans[+ansl]='0'+tmp;
        else ans[+ansl]='A'+tmp-10;
        x/=m;
    }while(x);
}
int main(){
    scanf("%d%s%d"&n,a &m);
    x=n_to_ten();
    ten_to_m();
    for(int i=ansl;i;--i)
    printf("%c",ans[i]);
    puts("");
    return 0;
}
```

向量

点积αβ=|α||β|cosθ

```
struct Point{
      double x,y;
      Point(double x = 0, double y = 0):x(x), y(y){ }//构造函数
}Vector[100];
double Dot(Vector A, Vector B){return A.x * B.x + A.y * B.y;}
/*
α×β
若β在α的逆时针方向，则为正值、顺时针则为负值、两向量共线则为0
*/
double Cross(Vector A, Vector B){return A.x * B.y - B.x * A.y;}
//取模（求长度）(Length)
double Length(Vector A){return sqrt(Dot(A, A));}
//计算两向量夹角(Angle)
double Angle(Vector A, Vector B){return acos(Dot(A, B) / Length(A) / Length(B));}
//计算两向量构成的平行四边形有向面积(Area2)
double Area2(Point A, Point B, Point C){return Cross(B - A, C - A);}
//计算向量逆时针旋转后的向量(Rotate)
Vector Rotate(Vector A, double rad){
return Vector(A.x * cos(rad) - A.y * sin(rad), A.x * sin(rad) + A.y * cos(rad));
}
```

海伦公式

```
p = (a + b + c)/2;
s = sqrt(p*(p-a)*(p-b)*(p-c));
```