思想方法

二分

```
//small to big
#include<cstdio>
#include<iostream>
#include<cstring>
#include<algorithm>
using namespace std;
int search(int *arr,int n,int key){
        int left=0,right=n-1;
        while(left<=right){
                int mid=(left+right)>>1;
                if(arr[mid]==key) return mid;
                else if(arr[mid]>key) right=mid-1;
                else left=mid+1;
        }
        return -1;
}
int searchFirstEqual(int *arr,int n,int key){
        int left=0,right=n-1;
        while(left<=right){
                int mid=(left+right)>>1;
                if(arr[mid]>=key) right=mid-1;
                else left=mid+1;
        }
        if(left<n&&arr[left]==key){
                return left;
        }
        return -1;
}
int searchLastEqual(int *arr,int n,int key){
        int left=0,right=n-1;
        while(left<=right){
                int mid=(left+right)>>1;
                if(arr[mid]>key) right=mid-1;
                else left=mid+1;
        }
        if(right>=0&&arr[right]==key){
                return right;
        }
        return -1;
}
int searchLastEqualOrSmaller(int *arr,int n,int key){
        int left=0,right=n-1;
        while(left<=right){
                int mid=(left+right)>>1;
                if(arr[mid]>key) right=mid-1;
                else left=mid+1;
        }
        return right;
}
int searchLastSmaller(int *arr,int n,int key){
        int left=0,right=n-1;
        while(left<=right){
                int mid=(left+right)>>1;
                if(arr[mid]>=key) right=mid-1;
                else left=mid+1;
        }
        return right;
}
int searchFirstEqualOrLarger(int *arr,int n,int key){
        int left=0,right=n-1;
        while(left<=right){
                int mid=(left+right)>>1;
                if(arr[mid]>=key) right=mid-1;
                else left=mid+1;
        }
        return left;
}
int searchFirstLarger(int *arr,int n,int key){
        int left=0,right=n-1;
        while(left<=right){
                int mid=(left+right)>>1;
                if(arr[mid]>key) right=mid-1;
                else left=mid+1;
        }
        return left;
}

int main(){
        int arr[17] = {1,
                           2, 2, 5, 5, 5,
                           5, 5, 5, 5, 5,
                           5, 5, 6, 6, 7};
    printf("First Equal           : - \n", searchFirstEqual(arr, 16, 5));
    printf("Last Equal            : - \n", searchLastEqual(arr, 16, 5));
    printf("First Equal or Larger : - \n", searchFirstEqualOrLarger(arr, 16, 5))
    printf("First Larger          : - \n", searchFirstLarger(arr, 16, 5));
    printf("Last Equal or Smaller : - \n", searchLastEqualOrSmaller(arr, 16, 5))
    printf("Last Smaller          : - \n", searchLastSmaller(arr, 16, 5));
    return 0;
}
```

三分

```
#include<iostream>
#include<cstdio>
using namespace std;
int n;
double a[200];
double l,r;
double speed(double a,int b){
        double cur=a,ans=1;
        while(b){
                if(b&1) ans*=cur;
                cur*=cur;
                b>>=1;
        }
        return ans;
}
double num(double x){
        double sum=0;
        for(int i=0;i<=n;i++){
                sum=sum+speed(x,i)*a[i];
        }
        return sum;
}
double abs(double c){
        if(c<0) return -c;
        return c;
}
int main(){
        scanf("%d%lf%lf",&n,&l,&r);
        for(int i=1;i<=n+1;i++){
                scanf("%lf",&a[n-i+1]);
        }
        while(abs(r-l)>0.000001){
                double x=(2*l+r)/3.0,y=(2*r+l)/3.0;
                if(num(x)>num(y)){
                        r=y;
                }
                else{
                        l=x;
                }

        }
        printf("%.5lf",l);
        return 0;
}
```

离散化

```
//unique,lower_bound,sort,离散化
#include<iostream>
#include<cstdio>
#include<algorithm>
using namespace std;
const int MAXN=10020;
int a[MAXN],b[MAXN];
int n,m;
int main(){
        scanf("%d",&n);
        for(int i=1;i<=n;i++){
                scanf("%d",&a[i]);
                b[i]=a[i];
        }
        sort(b+1,b+1+n);
        m=unique(b+1,b+1+n)-(b+1);
        for(int i=1;i<=n;i++){
                a[i]=lower_bound(b+1,b+1+m,a[i])-b;
        }
        for(int i=1;i<=n;i++){
                cout<<a[i]<<' ';
        }
        return 0;
}
```