基础算法模板

1. 归并排序求逆序对数量
2. 高精度加减乘除法
3. 离散化
4. 二分
5. 最长连续不重复子序列（双指针算法）
6. 数组模拟链表——头插法
7. kmp字符串
8. 滑动窗口
9. 单调栈
10. 模拟堆
11. 模拟散列表

1.归并排序求逆序对数量

```
#include<stdio.h>
int a[100005],n;
int temp[100005];
int sum;
long long int f(int q[],int l,int r)
{
    if(l>=r) return 0;

    int mid=(l+r)/2;

    long long int sum=f(q,l,mid)+f(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++];
        sum+=mid-i+1;
    }

    while(i<=mid)
    temp[k++]=q[i++];

    while(j<=r)
    temp[k++]=q[j++];

    for(k=0,i=l;i<=r;i++,k++)
    a[i]=temp[k];

    return sum;
}
int main()
{
    int i;
    scanf("%d",&n);

    for(i=0;i<n;i++)
    scanf("%d",&a[i]);


    printf("%lld",f(a,0,n-1));

 }
```

2.高精度加减乘除法

```
//高精度加法
#include<stdio.h>
#include<string.h>
char a[100005];
char b[100005];
int c[100005];
int len1,len2,i,i1;
int main()
{
  scanf("%s%s",a,b);
  len1=strlen(a);
  len2=strlen(b);

  for(i=0;len1-i-1>=0||len2-i-1>=0;i++)
  {
    if(len1-i-1>=0&&len2-i-1>=0)
    {
      c[i+1]=(c[i]+a[len1-i-1]-'0'+b[len2-i-1]-'0')/10;
      c[i]=(c[i]+a[len1-i-1]-'0'+b[len2-i-1]-'0')%10;
    }
```

```
    else if(len1-i-1>=0&&len2-i-1<0)
    {
      c[i+1]=(c[i]+a[len1-i-1]-'0')/10;
      c[i]=(c[i]+a[len1-i-1]-'0')%10;
    }

    else if(len1-i-1<0&&len2-i-1>=0)
    {
      c[i+1]=(c[i]+b[len2-i-1]-'0')/10;
      c[i]=c[i]+(b[len2-i-1]-'0')%10;
    }
  }

  if(c[i]==1)
  {
    for(i1=i;i1>=0;i1--)
    {
      printf("%d",c[i1]);
    }
  }
  if(c[i]==0)
  {
    for(i1=i-1;i1>=0;i1--)
    {
      printf("%d",c[i1]);
    }
  }

}

//高精度减法
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;
bool cmp(vector<int> A,vector<int> B)
{
  if(A.size()!=B.size())
  return A.size()>B.size();
  else
  {
    for(int i=A.size()-1;i>=0;i--)
    {
      if(A[i]!=B[i])
      return A[i]>B[i];
    }
  }
  return true;
}
vector<int> sub(vector<int> A,vector<int> B)
{
  vector<int> C;
  int t=0;
  for(int i=0;i<A.size();i++)
  {
    t=A[i]-t;
    if(i<B.size())
```

```
    t-=B[i];

    C.push_back((t+10)%10);
    if(t<0)
    t=1;
    else
    t=0;
  }

  while(C.size()>1&&C.back()==0)
  C.pop_back();

  return C;
}
int main()
{
  string a,b;
  vector<int> A,B;
  cin>>a>>b;//123456
  for(int i=a.size()-1;i>=0;i--)
  A.push_back(a[i]-'0');//A=[6,5,4,...]
  for(int i=b.size()-1;i>=0;i--)
  B.push_back(b[i]-'0');

  if(cmp(A,B))//判断是不是A>=B
  {
    auto C=sub(A,B);
    for(int i=C.size()-1;i>=0;i--)
    printf("%d",C[i]);
  }
  else
  {
    auto C=sub(B,A);

    printf("-");
    for(int i=C.size()-1;i>=0;i--)
    printf("%d",C[i]);
  }
}

//高精度乘法
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;
vector<int> mul(vector<int> a,int b)
{
  vector<int> c;
  int t=0;
  for(int i=0;i<a.size();i++)
  {
    t+=a[i]*b;
    c.push_back(t%10);
    t/=10;
  }
  while(t)
  {
    c.push_back(t%10);
```

```
    t/=10;
  }
  while(c.size()>1&&c.back()==0)
  c.pop_back();
  return c;
}

int main()
{
  string x;
  int b;
  cin>>x>>b;
  vector<int> a;
  for(int i=x.size()-1;i>=0;i--)
  {
    a.push_back(x[i]-'0');
  }
  vector<int> c;
  c=mul(a,b);

  for(int i=c.size()-1;i>=0;i--)
  {
    printf("%d",c[i]);
  }
}

//高精度除法
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;
vector<int> div(vector<int> A,int b,int &r)
{
    vector<int> C;
    r=0;

    for(int i=A.size()-1;i>=0;i--)
    {
        r=r*10+A[i];
        C.push_back(r/b);
        r%=b;
    }
    reverse(C.begin(),C.end());

    while(C.size()>1&&C.back()==0)
    C.pop_back();

    return C;
}
int main()
{
    string a;
    int b;
    vector<int> A;
    cin>>a>>b; //a=123456

    for(int i=a.size()-1;i>=0;i--)
        A.push_back(a[i]-'0');//A=[6,5,4,3,2,1]
```

```
    int r;
    auto C=div(A,b,r);

    for(int i=C.size()-1;i>=0;i--)
    {
        printf("%d",C[i]);
    }
    printf("\n");
    printf("%d",r);
}
```

3.离散化

```
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;
const int N=300005;
int a[N],s[N];
int n,m;
typedef pair<int,int> PII;
vector<int> all;
vector<PII> add,query;
int find(int x)
{
  int l=0,r=all.size()-1;

  while(l<r)
  {
    int mid=l+r>>1;

    if(all[mid]>=x)
    r=mid;
    else
    l=mid+1;
  }
  return r+1;
}
int main()
{
  scanf("%d%d",&n,&m);

  for(int i=0;i<n;i++)
  {
    int x,c;
    scanf("%d%d",&x,&c);

    add.push_back({x,c});
    all.push_back(x);
  }

  for(int i=0;i<m;i++)
```

```
  {
    int l,r;
    scanf("%d%d",&l,&r);

    query.push_back({l,r});
    all.push_back(l);
    all.push_back(r);
  }

  sort(all.begin(),all.end());
  all.erase(unique(all.begin(), all.end()), all.end());
```

4.二分

```
int findl()//找第一个大于等于x的数
{
  int l=0,r=n-1;
  int mid;
  while(l<r)
  {
    mid=(l+r)/2;
    if(a[mid]<x)
    l=mid+1;
    else
    r=mid;
  }
  return l;
}
int findr()//找第一个小于等于x的数
{
  int l=0,r=n-1;
  int mid;
  while(l<r)
  {
    mid=(l+r+1)/2;
    if(a[mid]<=x)
    l=mid;
    else
    r=mid-1;

  }
  return l;
}
```

5.最长连续不重复子序列（双指针算法）

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

```
using namespace std;
const int N=100005;
int n;
int a[N];
int s[N];
int main()
{
  int res=0;
  scanf("%d",&n);

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

  for(int l=1,r=1;r<=n;r++)
  {
    s[a[r]]++;
    while(s[a[r]]>1)
    {
      s[a[l]]--;
      l++;
    }
    res=max(res,r-l+1);
  }
  printf("%d",res);
}
```

6.数组模拟链表——头插法

```
void insert(int a)
{
    e[idx] = a, ne[idx] = head, head = idx ++ ;
}
```

7.kmp字符串

```
//lwy版本
#include<stdio.h>
#include<algorithm>
#include<iostream>
#include<string.h>
using namespace std;
const int N=100005;
const int M=1000005;
char p[N],s[M];//p为模板链，s为文本链
int ne[N];
int lenp,lens;
```

```
void getne()
{
  ne[0]=-1;
  int k=-1;
  int j=0;
  while(j<lenp)
  {
    if(k==-1||p[j]==p[k])
    {
      k++;
      j++;
      ne[j]=k;
    }
    else
    k=ne[k];
  }
}
void kmpfind()
{
  int i=0;
  int j=0;
  while(i<lens)
  {

    if(s[i]==p[j]||j==-1)
    {
      i++;
      j++;
    }
    else
    {
      j=ne[j];
    }

    if(j==lenp)
    {
    printf("%d ",i-j);
    j=ne[j];
    }

  }
}
int main()
{
  scanf("%d",&lenp);
  scanf("%s",p);
  scanf("%d",&lens);
  scanf("%s",s);

  getne();

//  for(int i=0;i<lenp;i++)
//  {
//    printf("%d ",ne[i]);
//  }
  kmpfind();
}
```

Copy of 模板

8.滑动窗口

```
#include<stdio.h>
#include<iostream>
using namespace std;
const int N=1000005;
int q[N];
int hh=0;//队头
int tt=-1;//队尾
int a[N];
int n,k;
int main()
{
  scanf("%d%d",&n,&k);

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

  //求滑动窗口里面的最小数
  for(int i=0;i<n;i++)
  {
    while(tt>=hh&&a[i]<=a[q[tt]])//优先队列的入队
    tt--;

    q[++tt]=i;

    if(tt>=hh&&q[hh]<i-k+1)//判断队头是否在滑动窗口里面，不在则队头往前移。
    hh++;


    if(i>=k-1)
    printf("%d ",a[q[hh]]);

  }
  puts("");
  hh=0;tt=-1;
    for(int i=0;i<n;i++)
    {
      while(tt>=hh&&a[i]>=a[q[tt]])//优先队列的入队
      tt--;

      q[++tt]=i;

      if(tt>=hh&&q[hh]<i-k+1)//判断队头是否在滑动窗口里面，不在则队头往前移。
      hh++;


      if(i>=k-1)
      printf("%d ",a[q[hh]]);
```

```
    }
}
```

9.单调栈

```
#include<iostream>
using namespace std;
const int N=100005;
int n;
int h[N];
int tt;
int main()
{
    scanf("%d",&n);

    for(int i=1;i<=n;i++)
    {
        int x;
        scanf("%d",&x);
       while(tt!=0&&h[tt]>=x)
       tt--;

       if(tt==0) printf("-1 ");
       else printf("%d ",h[tt]);

       h[++tt]=x;
    }
}
```

10.模拟堆

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
using namespace std;
//a[]——小根堆里面存的为数值
//p[]——下标索引到第几个插入的数
//fuck[]——第几个插入的数，索引每个下标
const int N=100005;
int a[N],p[N],fuck[N];
int cnt;//到第几个下标了
int st;//到第几个插入的数了
void up(int u)
{
  int t=u;
  if(u/2!=0&&a[u/2]>a[u])
  {
    t=u/2;
```

```
    swap(fuck[p[t]],fuck[p[u]]);
    swap(p[t],p[u]);
    swap(a[t],a[u]);
  }

  if(t!=u)
  up(t);
}
void down(int u)
{
  int t=u;
  if(u*2<=cnt&&a[u*2]<a[t])
  t=u*2;
  if(u*2+1<=cnt&&a[u*2+1]<a[t])
  t=u*2+1;

  if(t!=u)
  {
    swap(fuck[p[t]],fuck[p[u]]);
    swap(p[u],p[t]);
    swap(a[u],a[t]);
    down(t);
  }
}
int main()
{
  int n;
  string op;
  int k,x;
  cin>>n;
  while(n--)
  {
    /*for(int i=1;i<=cnt;i++)
    printf("%d ",a[i]);
    puts("");*/
    cin>>op;
    if(op=="I")
    {
      cin>>x;
      a[++cnt]=x;
      p[cnt]=++st;
      fuck[st]=cnt;
      up(cnt);
    }
    else if(op=="PM")
    {
      printf("%d\n",a[1]);
    }
    else if(op=="DM")
    {
      swap(fuck[p[1]],fuck[p[cnt]]);
      swap(p[1],p[cnt]);
      a[1]=a[cnt];
      cnt--;
      down(1);
    }
    else if(op=="D")
    {
```

```
      cin>>k;
      a[fuck[k]]=a[cnt];
      p[fuck[k]]=p[cnt];
      fuck[p[cnt]]=fuck[k];
      cnt--;
      up(fuck[k]);
      down(fuck[k]);
    }
    else if(op=="C")
    {
      cin>>k>>x;
      //printf("fuck[k]=%d\n",fuck[k]);
      a[fuck[k]]=x;
      up(fuck[k]);
      down(fuck[k]);
    }
  }
}
```

11.模拟散列表

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<string.h>
using namespace std;
const int N=100003;
int a[N];
int idx,e[N],ne[N];
void inset(int x)
{
  int k=(x%N+N)%N;
  //a[k]同时作为头结点
  e[idx]=x;
  ne[idx]=a[k];
  a[k]=idx++;
}

bool query(int x)
{
  int k=(x%N+N)%N;
  for(int r=a[k];r!=-1;r=ne[r])
  {
    if(e[r]==x)
    return true;
  }
  return false;
}
int main()
{
  int n;
  scanf("%d",&n);
  memset(a,-1,sizeof(a));
```

```
  while(n--)
  {
    char op[5];
    int x;
    scanf("%s%d",op,&x);

    if(*op=='I')
    {
      inset(x);
    }
    else
    {
      if(query(x))
      puts("Yes");
      else
      puts("No");
    }
  }
}
```