查找

|  |
| --- |
| 顺序查找 |
| 二分查找 |
| 分块查找 |
| 哈希查找 |

排序

|  |
| --- |
| void print\_indent(int indent=0)  {  int i;  for(i=0;i!=indent;++i)  {  printf(" ");  }  }  void print\_array(int\*p,int n,int highlight1=-1,int highlight2=-1)  {  int i;  for(i=0;i!=n;++i)  {  if(i==highlight1||i==highlight2)  {  printf("[%d]",p[i]);  }  else  {  printf(" %d ",p[i]);  }  }  }  void sort(){}  int main()  {  int i,max=20;  const int n=10;  int a[n];  srand(time(NULL));  for(i=0;i!=n;++i)  {  a[i]=rand()%max;  }  print\_array(a,n);printf("\n");  sort();  print\_array(a,n);printf("\n");  } |

|  |  |  |  |
| --- | --- | --- | --- |
| 插入排序   |  | | --- | | 直接插入排序 | | 折半插入排序  int insert\_position(int\*p,int n,int key)  {  int min=0,mid,max=n-1;  while(min<=max)  {  mid=(min+max)/2;  if(p[mid]==key)  {  return mid;  }  if(p[mid]>key)  {  max=min-1;  }  else  {  min=mid+1;  }  }  mid=(min+max)/2;  if(p[mid]<key)  {  return mid+1;  }  else  {  return mid;  }  }  void sort(int\*p,int n)  {  int i,j,pos,temp\_swap;  for(i=1;i!=n;++i)  {  pos=insert\_position(p,i,p[i]);  printf("(i=%d)",i);print\_array(p,n,i,pos);printf("\n");  temp\_swap=p[i];  for(j=i;j!=pos;--j)  {  p[j]=p[j-1];  }  p[pos]=temp\_swap;  printf ("(i=%d)",i);print\_array(p,i+1,pos);printf("\n");  }  } | | 希尔排序 | |
| 交换排序   |  | | --- | | 冒泡排序  void sort(int\*p,int n)  {  int i,flag=n-1,next\_flag;  int temp;  while(flag>0)  {  next\_flag=0;  printf ("flag=%d\n",flag);  for(i=0;i!=flag;++i)  {  if(p[i]>p[i+1])  {  temp=p[i];p[i]=p[i+1];p[i+1]=temp;  next\_flag=i;  printf ("(i=%d)",i);print\_array(p,10,i,i+1);printf ("\n");  }  }  flag=next\_flag;  printf ("flag=%d\n",flag);  printf ("\n");  }  } | | 快速排序  void sort(int\*p,int n,int indent=0)  {  int i,j,temp\_swap;  print\_indent(indent);printf("(n=%d)",n);printf("\n");  print\_indent(indent);print\_array(p,n,-1,n-1);printf("\n");  i=0;j=n-2;  print\_indent(indent);printf("(i=%d j=%d)\n",i,j);  while(1)  {  while(i!=n-1&&p[i]<=p[n-1])  {  ++i;  }  while(j!=-1&&p[j]>=p[n-1])  {  --j;  }  if(i<j)  {  temp\_swap=p[i];p[i]=p[j];p[j]=temp\_swap;  print\_indent(indent);print\_array(p,n,i,j);printf("(i=%d j=%d)",i,j);printf("\n");  }  else  {  break;  }  }  print\_indent(indent);printf("(i=%d j=%d)\n",i,j);  temp\_swap=p[i];p[i]=p[n-1];p[n-1]=temp\_swap;  print\_indent(indent);print\_array(p,n,i,n-1);printf("\n");  printf("\n");  if(i>1)sort(p,i,indent+2);  if(n-i-1>1)sort(p+i+1,n-i-1,indent+2);  } | |
| 选择排序   |  | | --- | | 简单选择排序  void sort(int\*p,int n)  {  int i,j,tempi,temp\_swap;  for(i=0;i!=n-1;++i)  {  tempi=i;  for(j=i+1;j!=n;++j)  {  if(p[j]<p[tempi])  {  tempi=j;  }  }  if(tempi!=i)  {  temp\_swap=p[i];p[i]=p[tempi];p[tempi]=temp\_swap;  printf("(i=%d tempi=%d)",i,tempi);print\_array(p,n,i,tempi);printf("\n");  }  }  } | | 堆排序  void heap\_refresh(int\*p,int n,int current)  {  int child=current\*2+1,temp\_swap;  //print\_indent(2);print\_array(p,n);printf("\n");  for(;child<n;current=child,child=child\*2+1)  {  if(child+1<n&&p[child]<p[child+1])++child;  if(p[current]>p[child])return;  temp\_swap=p[current];p[current]=p[child];p[child]=temp\_swap;  print\_indent(2);print\_array(p,n,current,child);printf("(current=%d child=%d)",current,child);printf("\n");  }  }  void sort(int\*p,int n)  {  int i,temp\_swap;  for(i=n/2-1;i!=0;--i)  {  print\_array(p,n,i,-1);printf("\n");  heap\_refresh(p,n,i);  }  printf("\n");  for(i=n;i!=1;--i)  {  print\_array(p,n,0,i-1);printf("\n");  heap\_refresh(p,i,0);  temp\_swap=p[i-1];p[i-1]=p[0];p[0]=temp\_swap;  print\_array(p,n,0,i-1);printf("\n");  }  } | |
| 归并排序  void sort(int\*p,int n,int indent=0)  {  int i,i1,i2;  int mid=(n-1)/2,n1=mid+1,n2=n-n1;  print\_indent(indent);print\_array(p,n);printf("\n");  if(n1>1)sort(p,n1,indent+2);  if(n2>1)sort(p+n1,n2,indent+2);  print\_indent(indent);print\_array(p,n1,0,n1-1);print\_array(p+n1,n2,0,n2-1);printf("\n");  int\*p\_temp=new int[n]();  i=i1=i2=0;  while(i1!=n1&&i2!=n2)  {  if(p[i1]<(p+n1)[i2])  {  p\_temp[i]=p[i1];  print\_indent(indent);print\_array(p,n1,i1);print\_array(p+n1,n2,i2);printf("(i1=%d i2=%d)\n",i1,i2);  ++i1;  }  else  {  p\_temp[i]=(p+n1)[i2];  print\_indent(indent);print\_array(p,n1,i1);print\_array(p+n1,n2,i2);printf("(i1=%d i2=%d)\n",i1,i2);  ++i2;  }  ++i;  print\_indent(indent);print\_array(p\_temp,i,i-1);printf("\n");  }  for(;i1!=n1;++i1,++i)  {  p\_temp[i]=p[i1];  }  for(;i2!=n2;++i2,++i)  {  p\_temp[i]=(p+n1)[i2];  }  for(i=0;i!=n;++i)  {  p[i]=p\_temp[i];  }  print\_indent(indent);print\_array(p,n);printf("\n");  delete[]p\_temp;  } |
| 分配排序   |  | | --- | | 基数排序 | | 桶排序  void sort(int\*p,int n,int max)//[0,max)  {  int\*p\_temp=new int[max]();  int i,j=0;  for(i=0;i!=n;++i)  {  ++(p\_temp[p[i]]);  }  for(i=0;i!=max;++i)  {  while(p\_temp[i]>0)  {  p[j]=i;  ++j;  --(p\_temp[i]);  printf("(i=%d p2[i]=%d)",i,p\_temp[i]);print\_array(p,j,j-1);printf("\n");  }  }  delete[]p\_temp;  } | |