**2.2.2 反向扫描**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | #include <bits/stdc++.h> //hdu 2029  using namespace std;  int main(){  int n; cin >> n; //n是测试用例个数  while(n--){  string s; cin >> s; //读一个字符串  bool ans = true;  int i = 0, j = s.size() - 1; //双指针  while(i < j){  if(s[i] != s[j]) { ans = false; break; }  i++; j--; //移动双指针  }  if(ans) cout << "yes" << endl;  else cout << "no" << endl;  }  return 0;  } |

**2.2.3 同向扫描**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | #include <bits/stdc++.h> //例2-4. 找相同数对  using namespace std;  const int N = 2e5 + 5;  int a[N];  int main (){  int n , c; cin >> n >> c;  for(int i = 1 ; i <= n ; i ++) cin >> a[i];  sort(a + 1 , a + 1 + n);  long long ans = 0; //答案数量超过int，需要用long long  for(int i=1,j=1,k=1; i <= n ; i ++) {  while(j <= n && a[j] - a[i] < c ) j ++; //用j、k查找数字相同的区间  while(k <= n && a[k] - a[i] <= c) k ++; //区间[j,k]内所有数字相同  if(a[j]-a[i]==c && a[k-1]-a[i]==c && k-1>=1) ans += k - j;  }  cout << ans;  return 0;  } |

**2.3.2 整数二分**

bin\_search()和lower\_bound()对比测试

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | | #include<bits/stdc++.h>  using namespace std;  #define MAX 100 //试试10000000  #define MIN -100  int a[MAX]; //大数组a[MAX]应定义为全局  unsigned long ulrand(){ //生成一个大随机数  return (  (((unsigned long)rand()<<24)& 0xFF000000ul)  |(((unsigned long)rand()<<12)& 0x00FFF000ul)  |(((unsigned long)rand()) & 0x00000FFFul));  }  int bin\_search(int \*a, int n, int x){ //a[0]～a[n-1]是有序的  int left = 0, right = n; //不是 n-1  while (left < right) {  int mid = left+(right-left)/2; //int mid = (left+ right)>>1;  if (a[mid] >= x) right = mid;  else left = mid + 1;  }  return left; //特殊情况：如果最后的a[n-1] < key，left = n  }  int main(){  int n = MAX;  srand(time(0));  while(1){  for(int i=0; i< n; i++) //产生[MIN, MAX]内的随机数，有正有负  a[i] = ulrand() % (MAX-MIN + 1) + MIN;  sort(a, a + n ); //排序，a[0]~a[n-1]  int test = ulrand() % (MAX-MIN + 1) + MIN; //产生一个随机的x  int ans = bin\_search(a,n,test);  int pos = lower\_bound(a,a+n,test)-a;  //比较bin\_search()和lower\_bound()的输出是否一致：  if(ans == pos) cout << "!"; //正确  else { cout << "wrong"; break;} //有错，退出  }  } |
| 例2-7. “通往奥格瑞玛的道路” 洛谷P1462 | | | | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | | #include<bits/stdc++.h> //例2-8. 进击的奶牛 洛谷P1824  using namespace std;  int n,c,x[100005]; //牛棚数量，牛数量，牛棚坐标  bool check(int dis){ //当牛之间距离最小为dis时，检查牛棚够不够  int cnt=1, place=0; //第1头牛，放在第1个牛棚  for (int i = 1; i < n; ++i) //检查后面每个牛棚  if (x[i] - x[place] >= dis){ //如果距离dis的位置有牛棚  cnt++; //又放了一头牛  place = i; //更新上一头牛的位置  }  if (cnt >= c) return true; //牛棚够  else return false; //牛棚不够  }  int main(){  scanf("%d%d",&n, &c);  for(int i=0;i<n;i++) scanf("%d",&x[i]);  sort(x,x+n); //对牛棚的坐标排序  int left=0, right=x[n-1]-x[0]; //right=1000000也行，因为是log(n)的  int ans = 0;  while(left < right){  int mid = left + (right - left)/2; //二分  if(check(mid)){ //当牛之间距离最小为mid时，牛棚够不够?  ans = mid; //牛棚够，先记录mid  left = mid + 1; //扩大距离  }  else right = mid; //牛棚不够，缩小距离  }  cout << ans; //打印答案  return 0;  } | |

**2.3.3 实数二分**

|  |
| --- |
| 例2-9. Pie poj 3122 |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | #include<stdio.h>  #include<math.h>  double PI = acos(-1.0); //3.141592653589793  #define eps 1e-5 //试试eps多小会超时，多大会Wrong Answer  double area[10010];  int n,m;  bool check(double mid){  int sum = 0;  for(int i=0;i<n;i++) //把每个圆蛋糕都按大小mid分开。统计总数  sum += (int)(area[i] / mid);  if(sum >= m) return true; //最后看总数够不够m个  else return false;  }  int main(){  int T; scanf("%d",&T);  while(T--){  scanf("%d%d",&n,&m); m++;  double maxx = 0;  for(int i=0;i<n;i++){  int r; scanf("%d",&r);  area[i] = PI\*r\*r;  if(maxx < area[i]) maxx = area[i]; //最大的一块蛋糕  }  double left = 0, right = maxx;  for(int i = 0; i<100; i++){ //for循环的次数，试试多小会Wrong Answer  //while((right-left) > eps) { //用while循环做二分  double mid = left+(right-left)/2;  if(check(mid)) left = mid; //每人能分到mid大小的蛋糕  else right = mid; //不够分到mid大小的蛋糕  }  printf("%.4f\n",left); // 打印right也对，right-left < eps  }  return 0;  } |

**2.4.2 实数三分**

|  |
| --- |
| 例2-10. 模板三分法 洛谷P3382 |

（1）三等分，mid1和mid2为[*l*, *r*]的三等分点。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | #include<bits/stdc++.h>  using namespace std;  const double eps = 1e-6;  int n;  double a[15];  double f(double x){ //计算函数值  double s = 0;  for(int i=n;i>=0;i--) s = s\*x + a[i]; //注意函数求值的写法  return s;  }  int main(){  double L,R; scanf("%d%lf%lf",&n,&L,&R);  for(int i=n;i>=0;i--) scanf("%lf",&a[i]);  while(R-L > eps){ // for(int i = 0; i<100; i++){ //用for也行  double k =(R-L)/3.0;  double mid1 = L+k, mid2 = R-k;  if(f(mid1) > f(mid2)) R = mid2;  else L = mid1;  }  printf("%.5f\n",L);  return 0;  } |

（2）近似三等分，mid1和mid2在[*l*, *r*]的中间点附近。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | #include<bits/stdc++.h>  using namespace std;  const double eps = 1e-6;  int n; double a[15];  double f(double x){  double s=0;  for(int i=n;i>=0;i--) s = s\*x + a[i];  return s;  }  int main(){  double L,R;  scanf("%d%lf%lf",&n,&L,&R);  for(int i=n;i>=0;i--) scanf("%lf",&a[i]);  while(R-L > eps){ // for(int i = 0; i<100; i++){ //用for也行  double mid = L+(R-L)/2;  if(f(mid - eps) > f(mid)) R = mid;  else L = mid;  }  printf("%.5f\n",L);  return 0;  } |

**2.4.3 整数三分**

|  |
| --- |
| 例2-13. 期末考试 洛谷 P3745 |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | #include<bits/stdc++.h>  const int N = 100005;  using namespace std;  typedef long long ll;  int n,m,t[N],b[N];  ll A,B,C,ans;  ll calc1(int p){ //计算通过A，B操作把时间调到p的不愉快度  ll x=0,y=0;  for(int i=1;i<=m;i++){  if(b[i]<p) x += p-b[i];  else y += b[i]-p;  }  if(A<B) return min(x,y)\*A+(y-min(x,y))\*B; //A<B,先用A填补，再用B  else return y\*B; //B<=A,直接全部使用B  }  ll calc2(int p) { //计算学生们的不愉快度总和  ll sum=0;  for(int i=1;i<=n;i++)  if(t[i]<p) sum += (p-t[i])\*C;  return sum;  }  int main(){  cin>>A>>B>>C>>n>>m;  for(int i=1;i<=n;i++) cin >> t[i];  for(int i=1;i<=m;i++) cin >> b[i];  sort(b+1,b+m+1); sort(t+1,t+n+1);  if(C>=1e16) { cout<<calc1(t[1])<<endl; return 0;} //一个特判  ans=1e16;  int l=1,r=N; //left, right  while(r-l > 2){ //把2改成其他数字也行，后面的for再找最小值  int mid1 = l+(r-l)/3; int mid2 = r-(r-l)/3;  ll c1 = calc1(mid1) + calc2(mid1); //总不愉快度  ll c2 = calc1(mid2) + calc2(mid2);  if (c1<=c2) r = mid2;  else l = mid1;  }  for(int i=l;i<=r;i++){ //在上面求出的区间内再枚举时间求出最小值  ll x = calc1(i) + calc2(i);  ans = min(ans,x);  }  cout<<ans<<endl;  return 0;  } |

**2.5.1 倍增**

|  |  |  |
| --- | --- | --- |
| 例2-14. 国旗计划 洛谷 P4155 | | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | #include<bits/stdc++.h>  using namespace std;  const int N = 4e5+1;  int n, m;  struct warrior{  int id, L, R; //id:战士的编号；L、R，战士的左右区间  bool operator < (const warrior b) const{return L < b.L;}  }w[N\*2];  int n2;  int go[N][20];  void init(){ //贪心 + 预计算倍增  int nxt = 1;  for(int i=1;i<=n2;i++){ //用贪心求每个区间的下一个区间  while(nxt<=n2 && w[nxt].L<=w[i].R)  nxt++; //每个区间的下一个是右端点最大的那个区间  go[i][0] = nxt-1; //区间i的下一个区间  }  for(int i=1;(1<<i)<=n;++i) //倍增：i=1,2,4,8,... 共log(n)次  for(int s=1;s<=n2;s++) //每个区间后的第2^i个区间  go[s][i] = go[go[s][i-1]][i-1];  }  int res[N];  void getans(int x){ //从第x个战士出发  int len=w[x].L+m, cur=x, ans=1;  for(int i=log2(N);i>=0;i--){ //从最大的i开始找：2^i = N  int pos = go[cur][i];  if(pos && w[pos].R < len){  ans += 1<<i; //累加跳过的区  cur = pos; //从新位置继续开始  }  }  res[w[x].id] = ans+1;  }  int main(){  scanf("%d%d",&n,&m);  for(int i=1;i<=n;i++){  w[i].id = i; //记录战士的顺序  scanf("%d%d",&w[i].L, &w[i].R);  if(w[i].R < w[i].L) w[i].R += m; //把环变成链  }  sort(w+1, w+n+1); //按左端点排序  n2 = n;  for(int i=1;i<=n;i++) //拆环加倍成一条链  { n2++; w[n2]=w[i]; w[n2].L=w[i].L+m; w[n2].R=w[i].R+m; }  init();  for(int i=1;i<=n;i++) getans(i); //逐个计算每个战士  for(int i=1;i<=n;i++) printf("%d ",res[i]);  return 0;  } |

**2.5.2 ST算法**

|  |
| --- |
| 例2-15. Balanced Lineup 洛谷P2880 (poj 3264) |

代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | #include <bits/stdc++.h>  using namespace std;  const int N = 50005;  int n,m;  int a[N],dp\_max[N][22],dp\_min[N][21];  int LOG2[N]; //自己计算以2为底的对数，向下取整  void st\_init(){  LOG2[0] = -1;  for(int i = 1;i<=N;i++) LOG2[i]=LOG2[i>>1]+1; //不用系统log()函数，自己算  for(int i=1;i<=n;i++){ //初始化区间长度为1时的值  dp\_min[i][0] = a[i];  dp\_max[i][0] = a[i];  }  //int p=log2(n); //可倍增区间的最大次方: 2^p <= n  int p= (int)(log(double(n)) / log(2.0)); //两者写法都行  for(int k=1;k<=p;k++) //倍增计算小区间。先算小区间，再算大区间，逐步递推  for(int s=1;s+(1<<k)<=n+1;s++){  dp\_max[s][k]=max(dp\_max[s][k-1], dp\_max[s+(1<<(k-1))][k-1]);  dp\_min[s][k]=min(dp\_min[s][k-1], dp\_min[s+(1<<(k-1))][k-1]);  }  }  int st\_query(int L,int R){  // int k = log2(R-L+1); //3种方法求k  int k = (int)(log(double(R-L+1)) / log(2.0));  // int k = LOG2[R-L+1]; //自己算LOG2  int x = max(dp\_max[L][k],dp\_max[R-(1<<k)+1][k]); //区间最大  int y = min(dp\_min[L][k],dp\_min[R-(1<<k)+1][k]); //区间最小  return x-y; //返回差值  }  int main(){  scanf("%d%d",&n,&m); //输入  for(int i=1;i<=n;i++) scanf("%d",&a[i]);  st\_init();  for(int i=1;i<=m;i++){  int L,R; scanf("%d%d",&L,&R);  printf("%d\n",st\_query(L,R));  }  return 0;  } |

**2.6.1 一维差分**

|  |
| --- |
| 例2-16. Color the ball hdu 1556 |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | //hdu 1556用差分数组求解  #include<bits/stdc++.h>  using namespace std;  const int N = 100010;  int a[N],D[N]; //a是气球，D是差分数组  int main(){  int n;  while(~scanf("%d",&n)) {  memset(a,0,sizeof(a)); memset(D,0,sizeof(D));  for(int i=1;i<=n;i++){  int L,R; scanf("%d%d",&L,&R);  D[L]++; D[R+1]--; //区间修改  }  for(int i=1;i<=n;i++){ //求原数组  a[i] = a[i-1] + D[i]; //差分。求前缀和a[]，a[i]就是气球i的值  if(i!=n) printf("%d ", a[i]); //逐个打印结果  else printf("%d\n",a[i]);  } //小技巧：14行到17行，把a[]改成D[]也行  }  return 0;  } |

|  |
| --- |
| 例2-17. 地毯 洛谷P3397 |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | #include<bits/stdc++.h>  using namespace std;  int D[5000][5000]; //差分数组  //int a[5000][5000]; //原数组，不定义也行  int main(){  int n,m; scanf("%d%d",&n,&m);  while(m--){  int x1,y1,x2,y2; scanf("%d%d%d%d",&x1,&y1,&x2,&y2);  D[x1][y1] += 1; D[x2+1][y1] -= 1;  D[x1][y2+1] -= 1; D[x2+1][y2+1] += 1; //计算差分数组  }  for(int i=1;i<=n;++i){//根据差分数组计算原矩阵的值（想象成求小格子的面积和）  for(int j=1;j<=n;++j){ //把用过的D[][]看成a[][]，就不用再定义a[][]了  //a[i][j] = D[i][j] + a[i-1][j] + a[i][j-1] - a[i-1][j-1];  //printf("%d ",a[i][j]); //这两行和下面两行的效果一样  D[i][j] += D[i-1][j] + D[i][j-1] - D[i-1][j-1];  printf("%d ",D[i][j]);  }  printf("\n");//换行  }  return 0;  } |

**2.6.3 三维差分**

|  |  |  |
| --- | --- | --- |
| 例2-18. 三体攻击 https://www.lanqiao.cn/problems/180/learning/ | | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58 | #include<stdio.h>  int A,B,C,n,m;  const int N = 1000005;  int s[N]; //存储舰队生命值  int D[N]; //三维差分数组（压维）；同时也用来计算每个点的攻击值  int x2[N], y2[N], z2[N]; //存储区间修改的范围，即攻击的范围  int x1[N], y1[N], z1[N];  int d[N]; //记录伤害，就是区间修改  int num(int x,int y,int z) {  //小技巧：压维，把三维坐标[(x,y,z)转为一维的((x-1)\*B+(y-1))\*C+(z-1)+1  if (x>A || y>B || z>C) return 0;  return ((x-1)\*B+(y-1))\*C+(z-1)+1;  }  bool check(int x){ //做x次区间修改。即检查经过x次攻击后是否有战舰爆炸  for (int i=1; i<=n; i++) D[i]=0; //差分数组的初值，本题是0  for (int i=1; i<=x; i++) { //用三维差分数组记录区间修改：有8个区间端点  D[num(x1[i], y1[i], z1[i])] += d[i];  D[num(x2[i]+1,y1[i], z1[i])] -= d[i];  D[num(x1[i], y1[i], z2[i]+1)] -= d[i];  D[num(x2[i]+1,y1[i], z2[i]+1)] += d[i];  D[num(x1[i], y2[i]+1,z1[i])] -= d[i];  D[num(x2[i]+1,y2[i]+1,z1[i])] += d[i];  D[num(x1[i], y2[i]+1,z2[i]+1)] += d[i];  D[num(x2[i]+1,y2[i]+1,z2[i]+1)] -= d[i];  }  //下面从x、y、z三个方向计算前缀和  for (int i=1; i<=A; i++)  for (int j=1; j<=B; j++)  for (int k=1; k<C; k++) //把x、y看成定值，累加z方向  D[num(i,j,k+1)] += D[num(i,j,k)];  for (int i=1; i<=A; i++)  for (int k=1; k<=C; k++)  for (int j=1; j<B; j++) //把x、z看成定值，累加y方向  D[num(i,j+1,k)] += D[num(i,j,k)];  for (int j=1; j<=B; j++)  for (int k=1; k<=C; k++)  for (int i=1; i<A; i++) //把y、z看成定值，累加x方向  D[num(i+1,j,k)] += D[num(i,j,k)];  for (int i=1; i<=n; i++) //最后判断是否攻击值大于生命值  if (D[i]>s[i])  return true;  return false;  }  int main() {  scanf("%d%d%d%d", &A, &B, &C, &m);  n = A\*B\*C;  for (int i=1; i<=n; i++) scanf("%d", &s[i]); //读生命值  for (int i=1; i<=m; i++) //读每次攻击的范围，用坐标表示  scanf("%d%d%d%d%d%d%d",&x1[i],&x2[i],&y1[i],&y2[i],&z1[i],&z2[i],&d[i]);  int L = 1,R = m; //经典的二分写法  while (L<R) { //对m进行二分，找到临界值。总共只循环了log(m)次  int mid = (L+R)>>1;  if (check(mid)) R = mid;  else L = mid+1;  }  printf("%d\n", R); //打印临界值  return 0;  } |

**2.7.1 离散化概念**

**2.7.2 离散化手工编码**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | #include<bits/stdc++.h>  const int N = 500010; //自己定义一个范围  struct data{  int val; //元素的值  int id; //元素的位置  }olda[N]; //离散化之前的原始数据  int newa[N]; //离散化后的结果  bool cmp(data x,data y){ return x.val < y.val; } //用于sort()  int main(){  int n; scanf("%d",&n); //读元素个数  for(int i=1;i<=n;i++) {  scanf("%d",&olda[i].val); //读元素的值  olda[i].id = i; //记录元素的位置  }  std::sort(olda+1,olda+1+n,cmp); //对元素的值排序  for(int i=1;i<=n;i++){ //生成 newa[]  newa[olda[i].id]=i; //这个元素原来的位置在olda[i].id，把它的值赋为i，i是离散化后的新值  if(olda[i].val == olda[i-1].val) //若两个元素的原值相同，把新值赋为相同  newa[olda[i].id] = newa[olda[i-1].id];  }  for(int i=1;i<=n;i++) printf("%d ",newa[i]); //打印出来看看  return 0;  } |

**2.7.3 用STL函数实现离散化**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | #include<bits/stdc++.h>  using namespace std;  const int N = 500010; // 自己定义一个范围  int olda[N]; // 离散化前  int newa[N]; // 离散化后  int main(){  int n; scanf("%d",&n);  for(int i=1;i<=n;i++) {  scanf("%d",&olda[i]); //读元素的值  newa[i] = olda[i];  }  sort(olda+1,olda+1+n); //排序  int cnt = n;  //cnt = unique(olda+1,olda+1+n)-(olda+1); //去重，cnt是去重后的数量  for(int i=1;i<=cnt;i++) //生成 newa[]  newa[i]=lower\_bound(olda+1,olda+1+n,newa[i])-olda;  //查找相等的元素的位置，这个位置就是离散化后的新值  for(int i=1;i<=cnt;i++) printf("%d ",newa[i]); //打印出来看看  printf("\n cnt=%d",cnt);  return 0;  } |

**2.8.1 排序函数**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #include<bits/stdc++.h>  using namespace std;  bool my\_less(int i, int j) {return (i < j);} //自定义小于  bool my\_greater(int i, int j)  {return (i > j);} //自定义大于  int main (){  vector<int> a = {3,7,2,5,6,8,5,4};  sort(a.begin(),a.begin()+4); //对前4个排序，输出2 3 5 7 6 8 5 4  //sort(a.begin(),a.end()); //从小到大排序， 输出2 3 4 5 5 6 7 8  //sort(a.begin(),a.end(),less<int>()); //输出2 3 4 5 5 6 7 8  //sort(a.begin(),a.end(),my\_less); //自定义排序，输出 2 3 4 5 5 6 7 8  //sort(a.begin(),a.end(),greater<int>()); //从大到小排序，输出 8 7 6 5 5 4 3 2  //sort(a.begin(),a.end(),my\_greater); // 输出 8 7 6 5 5 4 3 2  for(int i=0; i<a.size(); i++) cout<<a[i]<< " "; //输出  return 0;  } |

|  |
| --- |
| 例2-19. 奖学金 洛谷P1093。 |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | #include<bits/stdc++.h>  using namespace std;  struct stu{  int id; //学号  int c,m,e; //语、数、外  int sum;  }st[305];  bool cmp(stu a,stu b){  if(a.sum > b.sum) return true;  else if(a.sum < b.sum) return false;  else{ //a.sum == b.sum  if(a.c > b.c) return true;  else if(a.c < b.c) return false;  else{ //a.c == b.c  if(a.id > b.id) return false;  else return true;  }  }  }  int main(){  int n; cin>>n;  for(int i=1;i<=n;i++){  st[i].id = i; //学号  cin >> st[i].c >> st[i].m >> st[i].e;  st[i].sum = st[i].c + st[i].m + st[i].e; //总分  }  sort(st+1,st+1+n,cmp); //用cmp()排序  for(int i=1;i<=5;i++) cout << st[i].id << ' ' << st[i].sum << endl;  return 0;  } |

**2.8.2 排列**

**2．自写排列函数**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include<bits/stdc++.h>  using namespace std;  int a[20] = {1,2,3,4,5,6,7,8,9,10,11,12,13};  bool vis[20]; //记录第i个数是否用过  int b[20]; //生成的一个全排列  void dfs(int s,int t){  if(s == t) { //递归结束，产生一个全排列  for(int i = 0; i < t; ++i) cout << b[i] << " "; //输出一个排列  cout<<endl;  return;  }  for(int i=0;i<t;i++)  if(!vis[i]){  vis[i] = true;  b[s] = a[i];  dfs(s+1,t);  vis[i] = false;  }  }  int main(){  int n = 3;  dfs(0,n); //前n个数的全排列  return 0;  } |

|  |
| --- |
| 寒假作业 （蓝桥杯2016年省赛C++A组第6题） |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #include <bits/stdc++.h>  using namespace std;  int a[20] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};  int main() {  int ans=0;  do{ if( a[0]+a[1]==a[2] && a[3]-a[4]==a[5]  &&a[6]\*a[7]==a[8] && a[11]\*a[10]==a[9])  ans++;  }while(next\_permutation(a,a+13));  cout<<ans<<endl;  } |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | #include<bits/stdc++.h>  using namespace std;  int a[20]={1,2,3,4,5,6,7,8,9,10,11,12,13};  bool vis[20];  int b[20];  int ans=0;  void dfs(int s,int t){  if(s==12) {  if(b[9]\*b[10] == b[11]) ans++;  return;  }  if(s==3 && b[0]+b[1]!=b[2]) return; //剪枝  if(s==6 && b[3]-b[4]!=b[5]) return; //剪枝  if(s==9 && b[6]\*b[7]!=b[8]) return; //剪枝  for(int i=0;i<t;i++)  if(!vis[i]){  vis[i]=true;  b[s]=a[i]; //本题不用a[]，改成b[s]=i+1也行  dfs(s+1,t);  vis[i]=false;  }  }  int main(){  int n=13;  dfs(0,n); //前n个数的全排列  cout<<ans;  return 0;  } |

**2.9.1 汉诺塔和快速幂**

|  |
| --- |
| 例2-20. 汉诺塔 https://www.lanqiao.cn/problems/1512/learning/ |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | #include<bits/stdc++.h>  using namespace std;  int sum = 0, m;  void hanoi(char x,char y,char z,int n){ //三个柱子x、y、z  if(n==1) { //最小问题  sum++;  if(sum==m) cout<<"#"<<n<<": "<<x<<"->"<<z<<endl;  }  else { //分治  hanoi(x,z,y,n-1); //(1)先把x的n-1个小盘移到y，然后把第n个大盘移到z  sum++;  if(sum==m) cout<<"#"<<n<<": "<<x<<"->"<<z<<endl;  hanoi(y,x,z,n-1); //(2)把y的n-1个小盘移到z  }  }  int main(){  int n; cin>>n>>m;  hanoi('A','B','C',n);  cout<<sum<<endl;  return 0;  } |

**2. 快速幂**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #include<bits/stdc++.h>  using namespace std;  typedef long long ll; //注意要用long long，用int会出错  ll fastPow(ll a, ll n,ll m){ //an mod m  if(n == 0) return 1; //特判 a0 = 1  if(n == 1) return a;  ll temp = fastPow(a, n/2,m); //分治  if(n%2 == 1) return temp \* temp \* a % m; //奇数个a。也可以这样写： if(n &1)  else return temp \* temp % m ; //偶数个a  }  int main(){  ll a,n,m; cin >> a >> n>> m;  cout << fastPow(a,n,m);  return 0;  } |

**2.9.2 归并排序**

|  |  |  |
| --- | --- | --- |
| 例2-21. Inversion hdu 4911 | | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | #include<bits/stdc++.h>  const int N = 100005;  typedef long long ll;  ll a[N], b[N], cnt;  void Merge(ll l, ll mid, ll r){  ll i=l, j = mid+1, t=0;  while(i <= mid && j <= r){  if(a[i] > a[j]){  b[t++] = a[j++];  cnt += mid-i+1; //记录逆序对数量  }  else b[t++]=a[i++];  }  //一个子序列中的数都处理完了，另一个还没有，把剩下的直接复制过来：  while(i <= mid) b[t++]=a[i++];  while(j <= r) b[t++]=a[j++];  for(i=0; i<t; i++) a[l+i] = b[i]; //把排好序的b[]复制回a[]  }  void Mergesort(ll l, ll r){  if(l<r){  ll mid = (l+r)/2; //平分成两个子序列  Mergesort(l, mid);  Mergesort(mid+1, r);  Merge(l, mid, r); //合并  }  }  int main(){  ll n,k;  while(scanf("%lld%lld", &n, &k) != EOF){  cnt = 0;  for(ll i=0;i<n;i++) scanf("%lld", &a[i]);  Mergesort(0,n-1); //归并排序  if(cnt <= k) printf("0\n");  else printf("%I64d\n", cnt - k);  }  return 0;  } |

**2.9.3 快速排序**

|  |
| --- |
| 例2-22. Who's in the Middle Poj 2388 |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | #include <cstdio>  #include <algorithm>  using namespace std;  const int N = 10010;  int a[N]; //存数据  int quicksort(int left, int right, int k){  int mid = a[left + (right - left) / 2];  int i = left, j = right - 1;  while(i <= j){  while(a[i] < mid) ++i;  while(a[j] > mid) --j;  if(i <= j){  swap(a[i], a[j]);  ++i;--j;  }  }  if(left <= j && k <= j) return quicksort(left, j + 1, k);  if(i < right && k >= i) return quicksort(i, right, k);  return a[k]; //返回第k大数  }  int main(){  int n; scanf("%d", &n);  for(int i = 0; i < n; i++) scanf("%d", &a[i]);  int k = n/2;  printf("%d\n", quicksort(0, n, k));  return 0;  } |

2.10.1 贪心法

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | #include<bits/stdc++.h>  using namespace std;  const int n = 3;  int coin[] = {1,2,5};  int main(){  int i, money;  int ans[n] = {0}; //记录每种硬币的数量  cin >> money; //输入钱数  for(i= n-1; i>=0; i--){ //求每种硬币的数量  ans[i] = money/coin[i];  money = money - ans[i]\*coin[i];  }  for(i= n-1; i>=0; i--)  cout << coin[i]<<": "<<ans[i]<<endl; //输出每种面值硬币的数量  return 0;  } |