1.(a)LL exp(LL a, LL n)

{

LL tmp=1;

while (n>0)

{

if (b&1) tmp=tmp\*a;

a=a\*a;

b=b/2;

}

return tmp;

}

复杂度O(logn)

(b) matrix operator ^(matrix e, llg k)

{

matrix tmp=I;

while (k)

{

if (k&1) tmp=tmp\*e;

k=k/2;

e=e\*e;

}

return tmp;

}

复杂度O(logn)

2.(a)(1)n=1时成立；假设n=k时成立，即f(k+2)=1+∑f(k)， 当n=k+1时，f(k+3)=f(k+2)+f(k+1)=1+∑f(k)+f(k+1)=1+∑f(k+1)。结论成立。

(2)n=1 时成立 ； 假设n<=k时成立即f(k+2)，当n=k+1时， f(k+3)=f(k+2)+f(k+1)=。结论成立。

(b)O(2n) (c)O(n) (d)O(logn)

(e)第三种方法效率较高

3.(1)将平面不断地划分成左右两个半区，分别求解完左右两个半区的答案，然后将左右两边的点按照y坐标升序排序，然后计算跨越左右两个半区的答案，即为左边的点选取可以匹配的右边的点。计算的时候，每找到一对友谊点对，就做如下处理：

每个点记录yh和yl代表可以选取的点的y坐标区间。当一个点找到在它右上角的点时，更新yh，找到右下角的点时更新yl。然后下次再找点的时候，只去寻找y坐标在[yh, yl]区间里的点。并且因为按y坐标排序后每个点的[yh, yl]区间不可能有交叉，所以线性扫描一遍两边的点集即可。

(2)Solve(l, r)

{

L==r return

mid=(l+r)/2

Solve(l, mid)

Solve(mid+1, r)

For p in VL

Find all q in VR that yl<=q.y<=yh

Update yl and yh for p

Merge(VL, VR)

}

(3)F(n)=2\*F(n/2)+n F(n)=O(nlogn)

4.类似最近点对的算法，只不过把每次的d换成当前最小周长的一半。

5.蛮力法枚举每个线段，对于每个线段判断是否所有点都在这个线段的同侧，这样可以保证求解出来的满足条件的线段构成的多边形的内部。

6.将平面不断地划分成左右两个半区，分别求解完左右两个半区的答案。那么当前状态有两种：(1)所有的点均已配对，那么问题求解完毕。(2)左边剩x个相同颜色的点，右边剩y个另一种颜色的点。那么只需将这些点一一匹配即可。

7.对q求凸包，再将p、q凸包的边界看成直线，做一遍半平面交，如果半平面交为和q点的凸包相同，则说明q点全在p点的凸包里面。

7.树分治。做法详见 《分治算法在树的路径问题中的应用》

8.MidNum(lx, rx, ly, ry)

{

If (lx==rx) return min(x[lx], y[ly])

Midx=(lx+rx)/2;

Midy=(ly+ry)/2;

If (x[midx]==y[midy]) return x[midx]

Else if (x[midx]<y[midy] return MidNum(midx+1, rx, ly, midy)

Else return MidNum(lx, midx, midy+1, ry)

}

9.(1)归并排序，同时求出逆序对个数。

(2)

void solve(int l,int r)

{

        if (l>=r) return ;

        int mid=(l+r)>>1;

        solve(l,mid);

        solve(mid+1,r);

        int n1=0, n2=0;

        for (int i=l;i<=mid;i++) p[n1++]=a[i];

        for (int j=mid+1;j<=r;j++) q[n2++]=a[j];

        p[n1]=q[n2]=inf;

        int i=0,j=0;

        for (int k=l;k<=r;k++)

        {

                if (p[i]<=q[j]) a[k]=p[i++];

                else a[k]=q[j++], ans+=(n1-i);

        }

}

10.首先对S集合排序，然后对于S中的每一个数p，二分查找x-p是否存在。复杂度O(nlogn)

11.(1)第k个

(2)二分k，判断是否ak>ak+1

12.先将A、B排序，然后以A为横向量，B为列向量，通过A\*B得到一个矩阵P。我们要查找第k小的元素，首先二分得到一个矩阵中第k小的元素x，然后通过比较，找到矩阵中最大比x小的元素，即为第k小的元素。

13.对于一个子矩阵(I,j,a,b)来说，判断p[i][j]<=x<=p[a][b]是否成立，如果成立，则将矩阵分为(i,j,a/2,b/2), (a/2,j,a, b/2), (i,b/2,a/2,b),(a/2,b/2,a/b)四个子矩阵，递归求解。

14.对白点和黑点分别作凸包，然后判断两个凸包是否有交。