//2d Seg Tree max + sum

int a[1003][1003];

int t[3\*1005][3\*1005];

int tmn[3\*1005][3\*1005];

int r,c; // x y

inline int max(int a,int b)

{

if(a>b)

return a;

return b;

}

inline int min(int a,int b)

{

if(a>b)

return b;

return a;

}

void build\_y (int vx, int lx, int rx, int vy, int ly, int ry)

{

if (ly == ry)

if (lx == rx)

t[vx][vy] = a[lx][ly];

else

t[vx][vy] = t[vx\*2][vy] + t[vx\*2+1][vy];

else

{

int my = (ly + ry) / 2;

build\_y (vx, lx, rx, vy\*2, ly, my);

build\_y (vx, lx, rx, vy\*2+1, my+1, ry);

t[vx][vy] = t[vx][vy\*2] + t[vx][vy\*2+1];

}

}

void build\_x (int vx, int lx, int rx)

{

if (lx != rx)

{

int mx = (lx + rx) / 2;

build\_x (vx\*2, lx, mx);

build\_x (vx\*2+1, mx+1, rx);

}

build\_y (vx, lx, rx, 1, 1, c);

}

int sum\_y (int vx, int vy, int tly, int try\_, int ly, int ry)

{

if (ly > ry)

return 0;

if (ly == tly && try\_ == ry)

return t[vx][vy];

int tmy = (tly + try\_) / 2;

return sum\_y (vx, vy\*2, tly, tmy, ly, min(ry,tmy))

+ sum\_y (vx, vy\*2+1, tmy+1, try\_, max(ly,tmy+1), ry);

}

int sum\_x (int vx, int tlx, int trx, int lx, int rx, int ly, int ry)

{

if (lx > rx)

return 0;

if (lx == tlx && trx == rx)

return sum\_y (vx, 1, 1, c, ly, ry);

int tmx = (tlx + trx) / 2;

return sum\_x (vx\*2, tlx, tmx, lx, min(rx,tmx), ly, ry)

+ sum\_x (vx\*2+1, tmx+1, trx, max(lx,tmx+1), rx, ly, ry);

}

//For RMQ

void build\_y\_mn (int vx, int lx, int rx, int vy, int ly, int ry)

{

if (ly == ry)

if (lx == rx)

tmn[vx][vy] = a[lx][ly];

else

tmn[vx][vy] = max(tmn[vx\*2][vy] , tmn[vx\*2+1][vy]);

else

{

int my = (ly + ry) / 2;

build\_y\_mn (vx, lx, rx, vy\*2, ly, my);

build\_y\_mn (vx, lx, rx, vy\*2+1, my+1, ry);

tmn[vx][vy] = max(tmn[vx][vy\*2] , tmn[vx][vy\*2+1]);

}

}

void build\_x\_mn (int vx, int lx, int rx)

{

if (lx != rx)

{

int mx = (lx + rx) / 2;

build\_x\_mn (vx\*2, lx, mx);

build\_x\_mn (vx\*2+1, mx+1, rx);

}

build\_y\_mn (vx, lx, rx, 1, 1, c);

}

int min\_y (int vx, int vy, int tly, int try\_, int ly, int ry)

{

if (ly > ry)

return -INF;

if (ly == tly && try\_ == ry)

return tmn[vx][vy];

int tmy = (tly + try\_) / 2;

return max(min\_y (vx, vy\*2, tly, tmy, ly, min(ry,tmy))

, min\_y (vx, vy\*2+1, tmy+1, try\_, max(ly,tmy+1), ry));

}

int min\_x (int vx, int tlx, int trx, int lx, int rx, int ly, int ry)

{

if (lx > rx)

return -INF;

if (lx == tlx && trx == rx)

return min\_y (vx, 1, 1, c, ly, ry);

int tmx = (tlx + trx) / 2;

return max(min\_x (vx\*2, tlx, tmx, lx, min(rx,tmx), ly, ry)

, min\_x (vx\*2+1, tmx+1, trx, max(lx,tmx+1), rx, ly, ry));

}

//Usage

build\_x(1,1,r);

build\_x\_mn(1,1,r);

cur\_sum = sum\_x(1,1,r,i,i+a-1,j,j+b-1);

cur\_min = min\_x(1,1,r,i,i+a-1,j,j+b-1);