#define INF 999999999999999

//2D Sparse Table

//0 based indexes for everything

//2d matrix 0 based row columns

inline int max(int a,int b)

{

if(a>b)

return a;

return b;

}

inline int max(int a,int b,int c,int d)

{

if(a>=b && a>=c && a>=d)

return a;

else if(b>=a && b>=c && b>=d)

return b;

else if(c>=a && c>=b && c>=d)

return c;

return d;

}

inline int min(int a,int b)

{

if(a>b)

return b;

return a;

}

int n,m; //dimension of the original matrix

int M[1002][1002][11][11]; //sparse table

int matrix[1005][1005]; // contains the original 2d matrix

int cum\_matrix[1005][1005];

void sparse\_table\_init()

{

for (int i = 0 ; (1<<i) <= n; i += 1)

{

for(int j = 0 ; (1<<j) <= m ; j += 1)

{

for (int x = 0 ; x + (1<<i) -1 < n; x+= 1)

{

for (int y = 0 ; y + (1<<j) -1 < m; y+= 1)

{

if (i == 0 and j == 0)

M[x][y][i][j] = matrix[x][y]; // store x, y

else if (i == 0)

M[x][y][i][j] = max(M[x][y][i][j-1], M[x][y+(1<<(j-1))][i][j-1]);

else if (j == 0)

M[x][y][i][j] = max(M[x][y][i-1][j], M[x+ (1<<(i-1))][y][i-1][j]);

else

M[x][y][i][j] = max(M[x][y][i-1][j-1], M[x + (1<<(i-1))][y][i-1][j-1], M[x][y+(1<<(j-1))][i-1][j-1], M[x + (1<<(i-1))][y+(1<<(j-1))][i-1][j-1]);

}

}

}

}

}

inline int clz(int N) {

return N ? 32 - \_\_builtin\_clz(N) : -INF;

}

inline int max\_rn(int x, int y, int x1, int y1)

{

int k = clz(x1 - x + 1) - 1; //O(log2(N))

int l = clz(y1 - y + 1) - 1; //O(log2(N))

int ans = max(M[x][y][k][l], M[x1 - (1<<k) + 1][y][k][l], M[x][y1 - (1<<l) + 1][k][l], M[x1 - (1<<k) + 1][y1 - (1<<l) + 1][k][l]);

return ans;

}