1. Given a value x, we can know the rank of x in the matrix in $O(n)$ time, by monotone pointers. Using binary search (on n^2 elements), the running time is $O(n \log U)$ or $O(n \log n)$.
2. $O(\min(k, m))$ if it's a $n \times m$ matrix and $n \leq m$.
https://chaoxuprime.com/posts/2014-04-02-selection-in-a-sorted-matrix.html

for small k:
3. put the first element in each row into a heap, whenever we pop, add the next element in the corresponding row. $O(k \log n)$.
4. $O(\sqrt{k} \log k)$. The crucial observation is that there are xy elements smaller than entry (x, y), so the possible region for the solution can be divided into $2\sqrt{k}$ rows and columns (i.e. row/columns $1, \ldots, \sqrt{k}$). use results on selection in union of sorted arrays.
see reference for 4. median of two sorted arrays.

note. is [1] useful to get $O(k)$?

References

[1] Greg N Frederickson. An optimal algorithm for selection in a min-heap. *Information and Computation*, 104(2):197–214, 1993.