External Sort

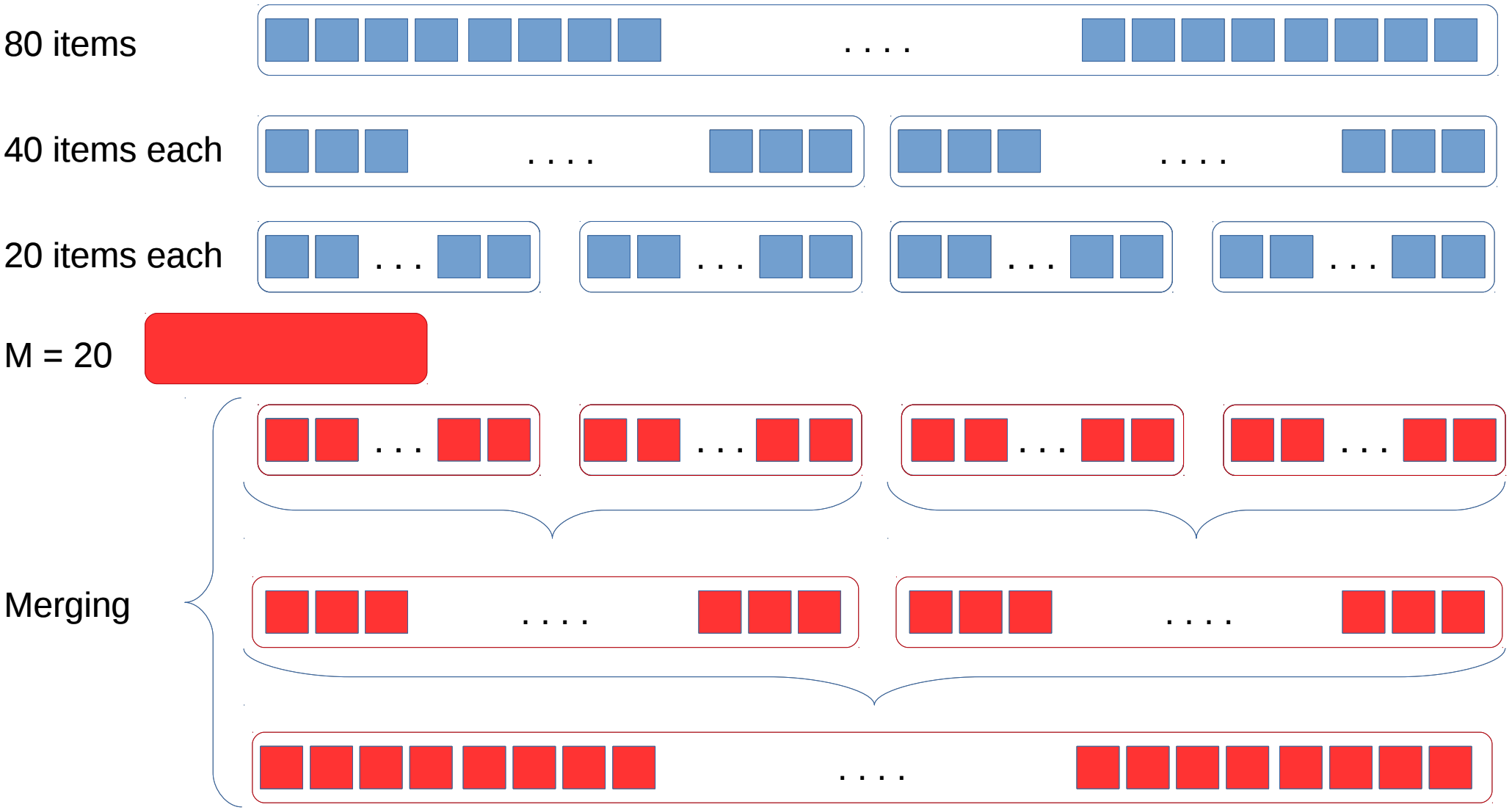
Input: N items

Internal Memory – capacity: M items

- If N <= M : Internal Sort
- Else:
  - Create *k* sorting - subproblems
  - Invoke External Sort for each subproblem
  - Merge *k sorted* sub arrays

N=80
M=20
k=2

Create sorting sub-problems
Internal sort

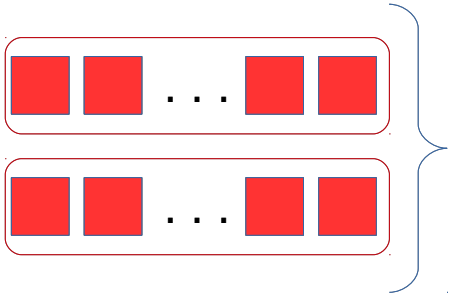


Merging

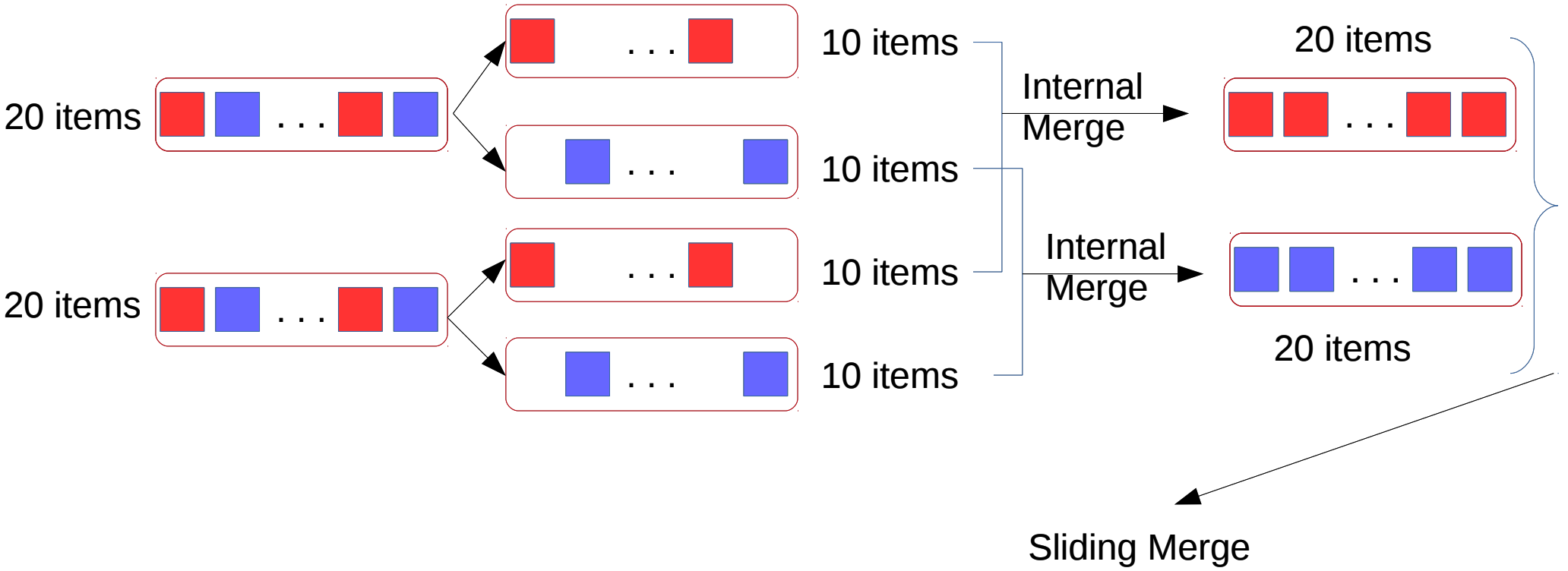
There are k – sorted arrays, each has n elements => merging to 1 array of k.n elements

- If k.n < M: internal merge
- Else:
  - create k merging subproblems, each is to merge k *sorted sub-arrays (after this, we have k*k sub arrays)*
  - Recursively solve these sub-problems
  - Sliding merge to get the final merged array

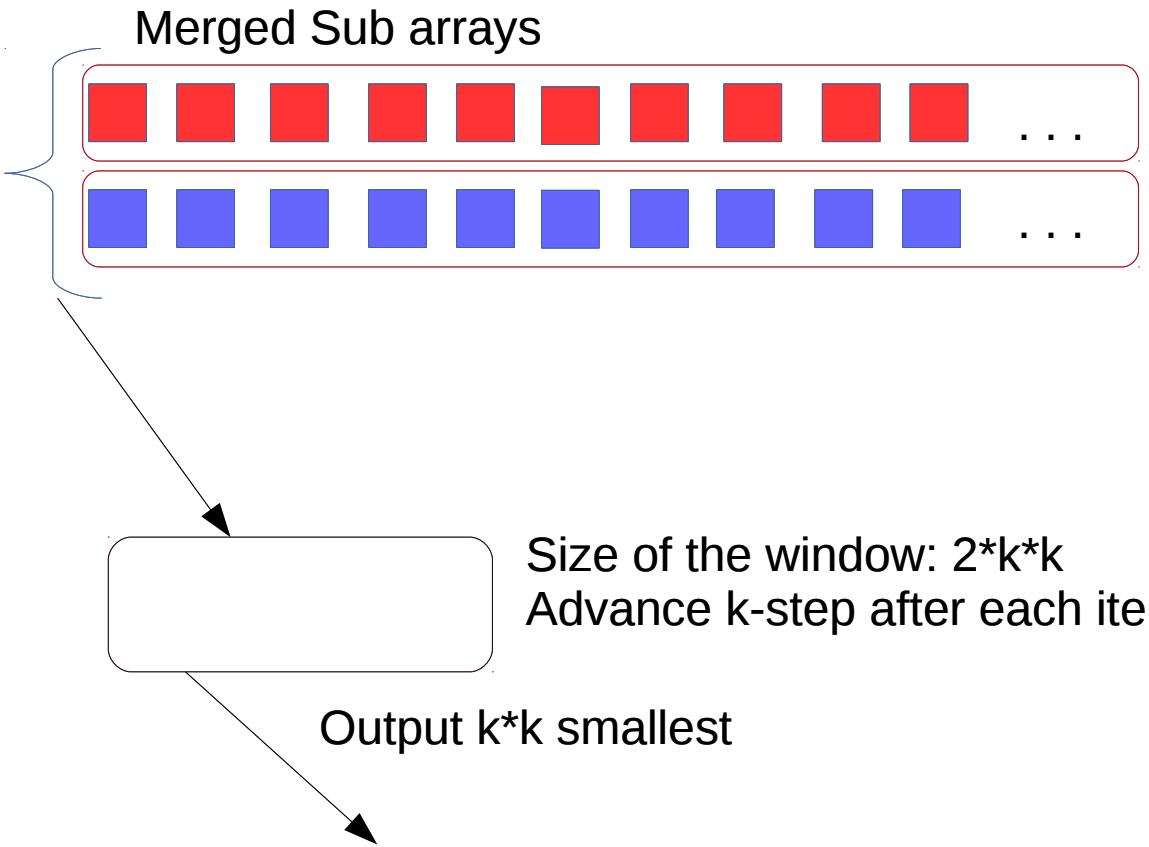
Merging



Total size = 40 → not fit in M
→ create merge sub-problems (k = 2)



Sliding Merging



- First iteration: take in 2*k items from each merged sub-array
=> 2*k*k in total, output k*k, keep k*k

- From 2nd iteration onward, take k Items from each merged sub-array
=> takes in extra k*k each time, output k*k each time