CS 186 Discussion #1

External Sorting & Hashing

Original Slides + Diagrams by Michelle Nguyen

Topics for Today

- Introductions
- Logistics
- Rendezvous
- External Sorting + Hashing

Pete Yeh

EECS 2016

peteyeh@berkeley.edu

Discussions

Tue 5-6 pm	3107 Etcheverry
Wed 1-2pm	3113 Etcheverry

Office Hours

MF 10-11 am	651 Soda

Meet New Friends!

(or project partners)

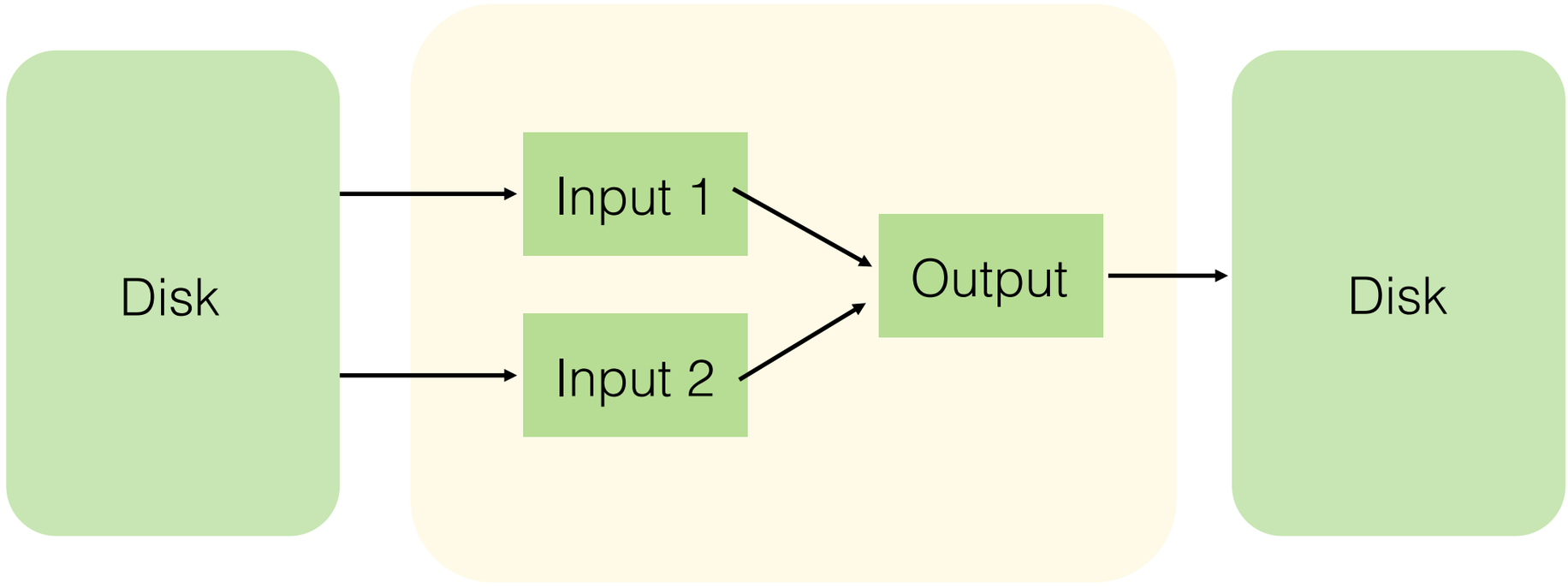
Logistics

- Enrollment
- Vitamins
  - Weekly, Online
  - Released Thursday, Due Monday
- Projects
  - Five Projects, Github
  - Four slip days, -25% per day after
  - Partners (Optional)

Time-Space Rendezvous

External Sorting

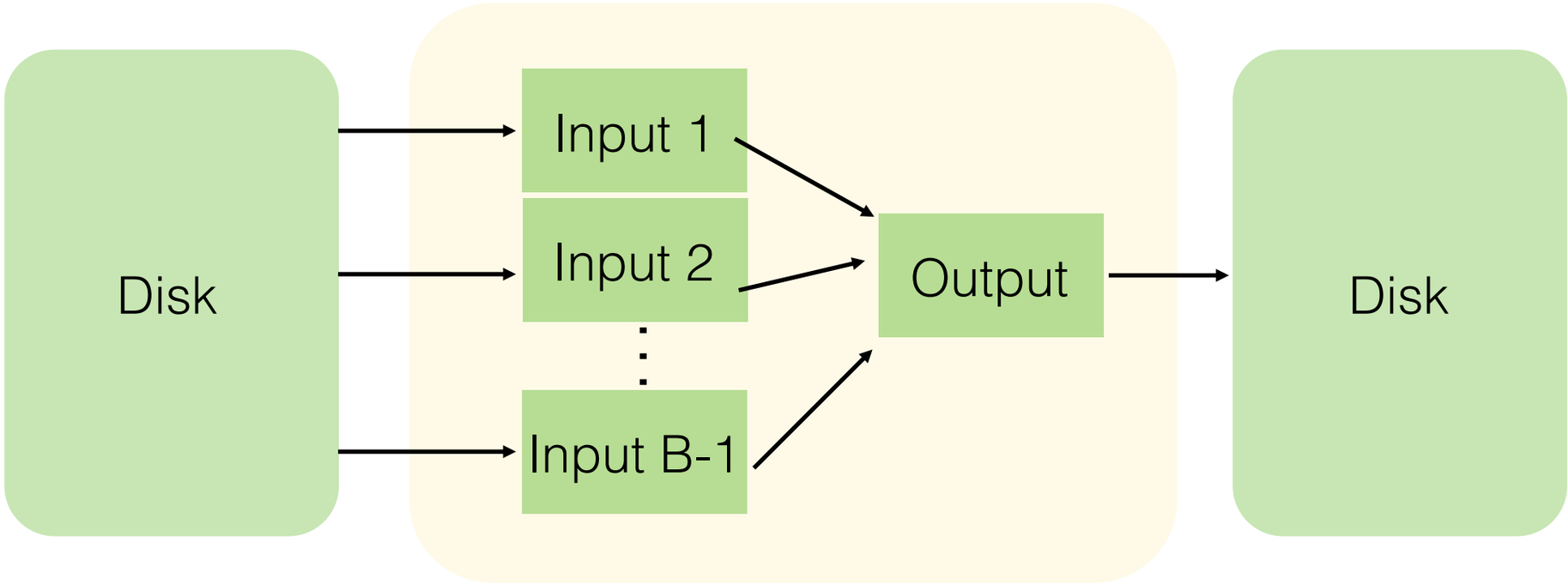
- Two-way Merge Sort (Merge Step)



- Buffer size of 3 pages

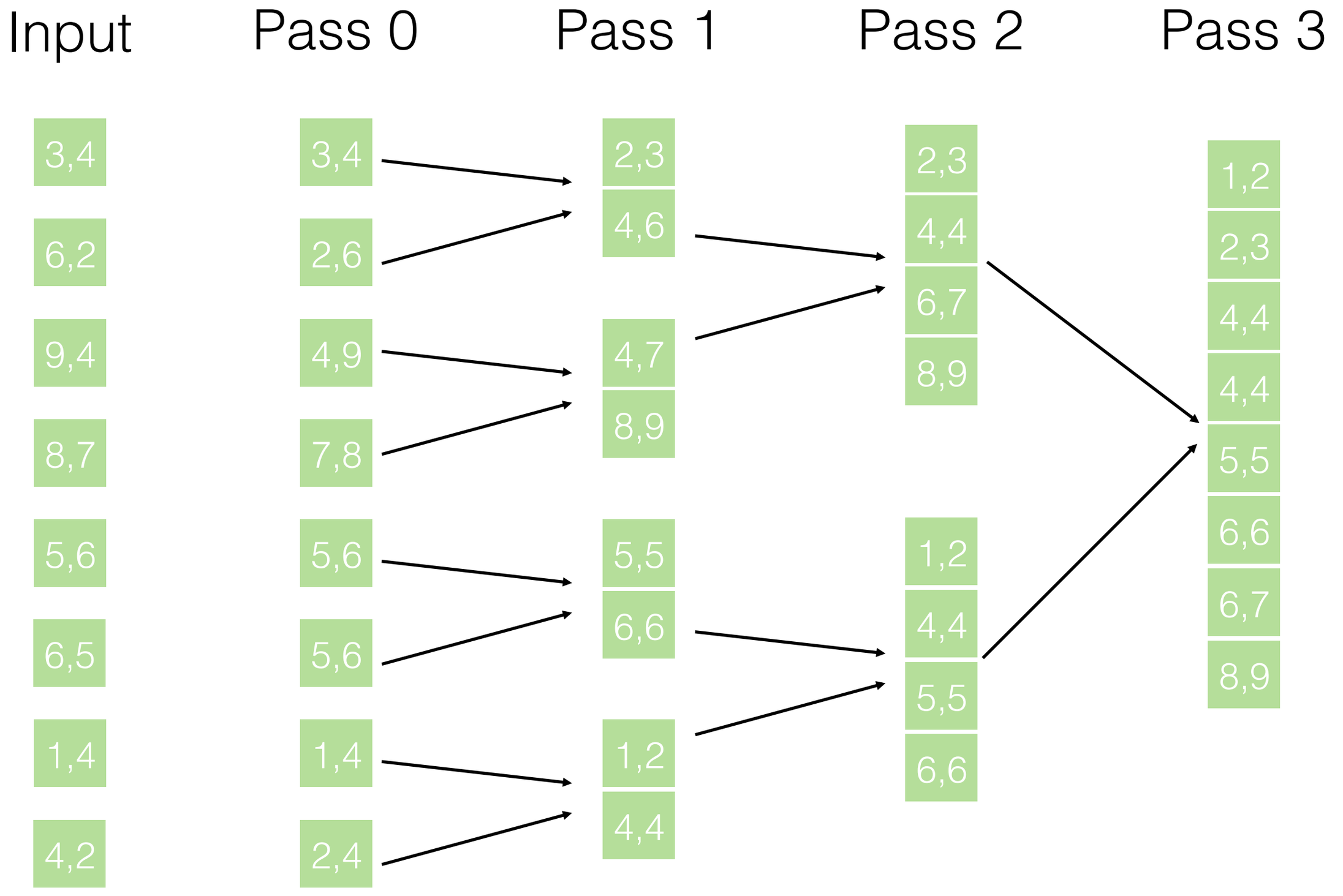
External Sorting

- General Merge Sort (Merge Step)



- Buffer size of *B* pages

External Sorting



External Sorting

- *N* blocks in file, *B* blocks in memory
- Number of Passes
  - Two-way $\lceil \log_2 N \rceil + 1$
  - Generalized $\left\lceil \log_{B-1} \left\lceil \frac{N}{B} \right\rceil \right\rceil + 1$
- Total Cost (I/Os)

$2N$ * [# of passes]

External Sorting

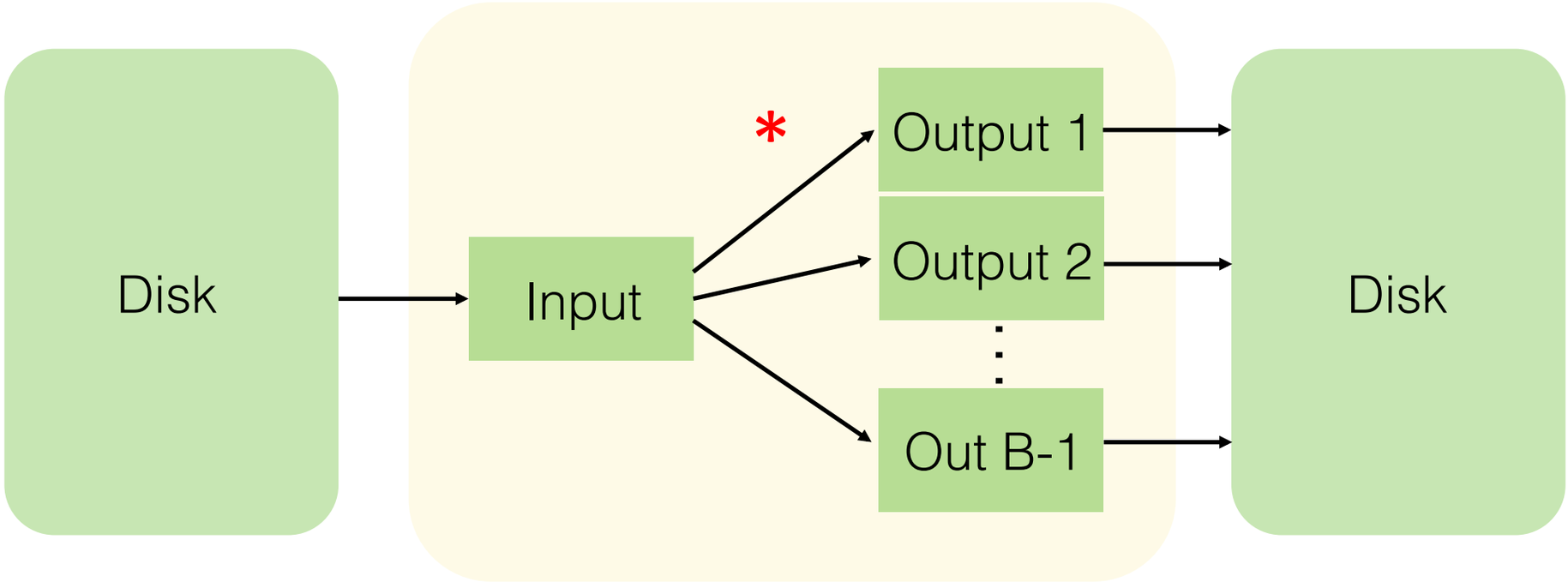
- How big of a file can we sort in two passes?

$$B(B-1)$$

- Why?

External Hashing

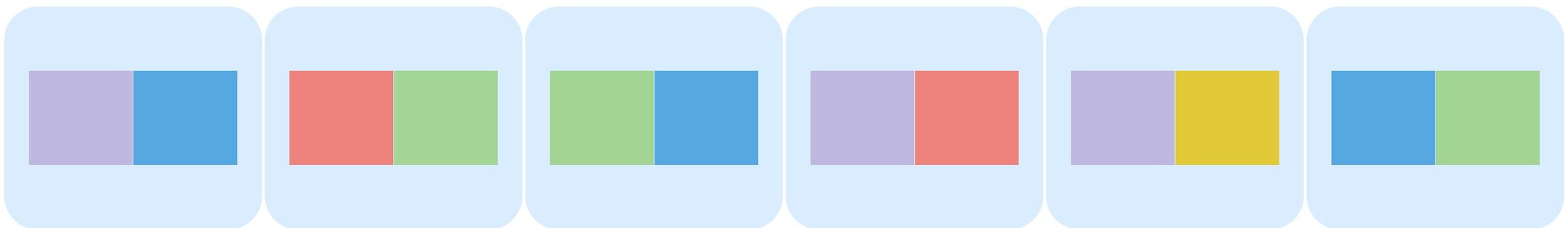
- Partition (Divide) Step



- Buffer size of *B* pages
- * = hash function!

Aggregating Colors

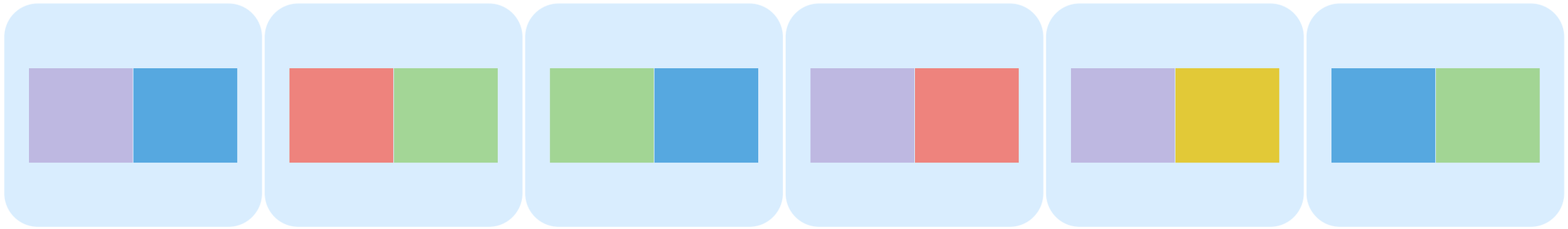
- Goal: Group squares by color
- Setup: 12 squares, each page fits 2 squares. We can hold 4 pages in memory.
- $N = 6$, $B = 4$



Pass 1: Divide

- Read all pages in, hash to B-1 partitions/buckets so that each group guaranteed to be in same partition.
- May not be a whole partition for each group.
- # I/O's = 2N

Pass 1: Divide



N=6, B=4

Assign colors to 3 partitions using hash function.

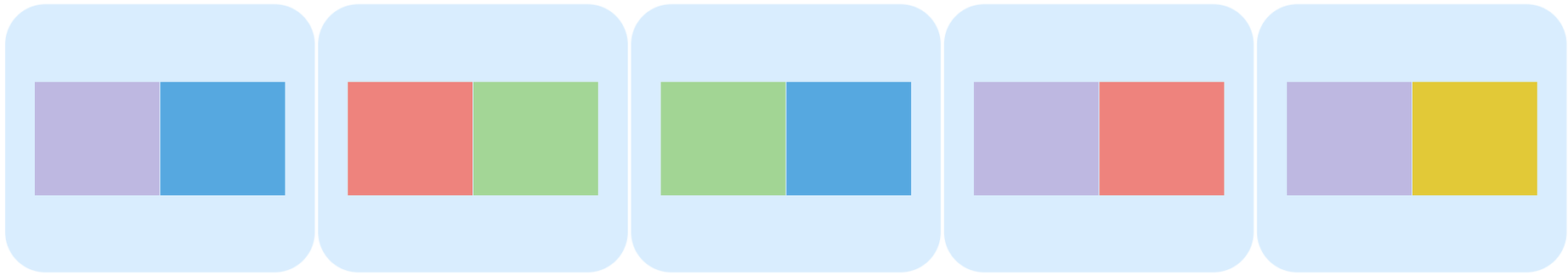
Our hash function:

{G,P} -> 1

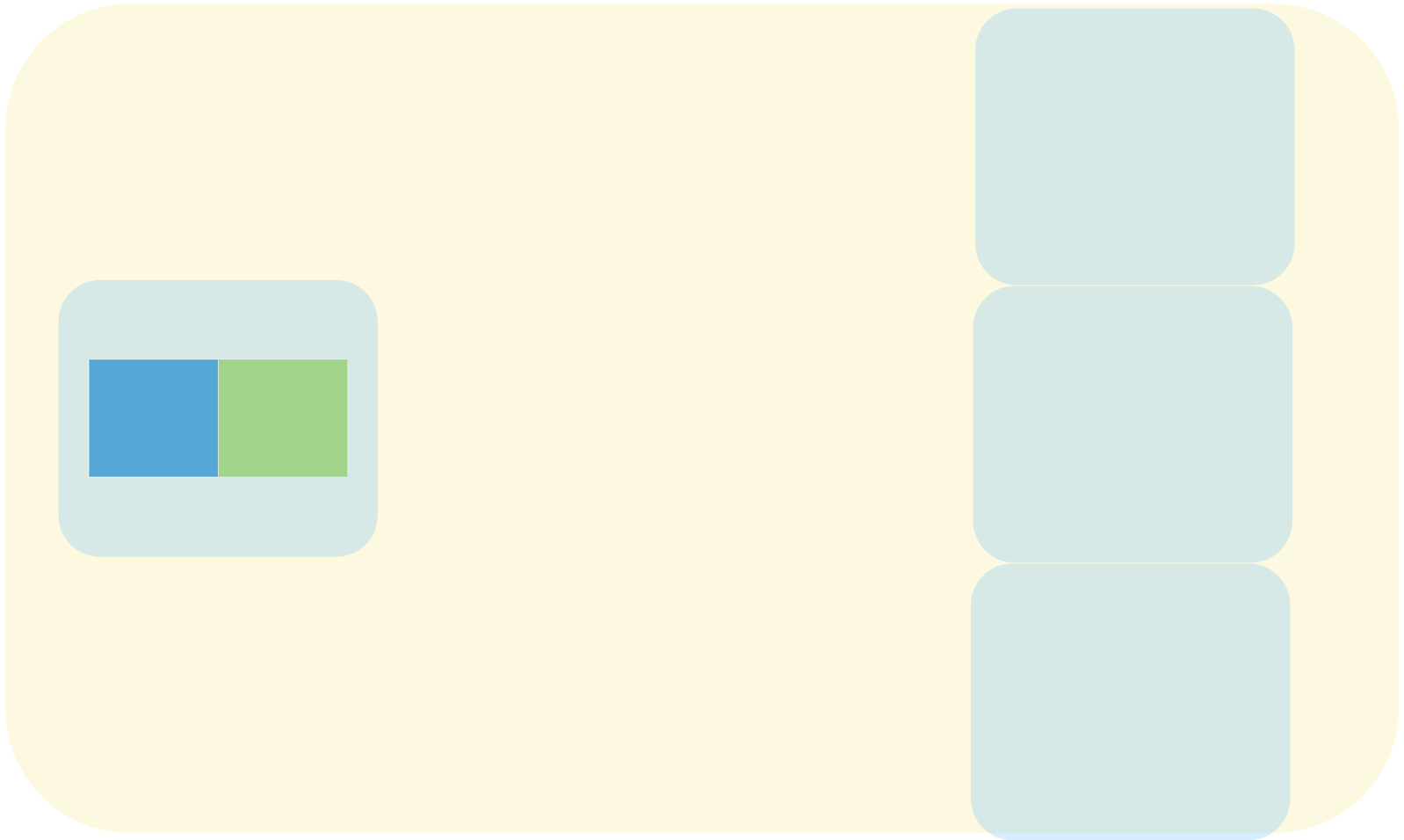
{B} -> 2

{R, Y} -> 3

Pass 1: Divide



N=6, B=4



Assign colors to 3 partitions using hash function.

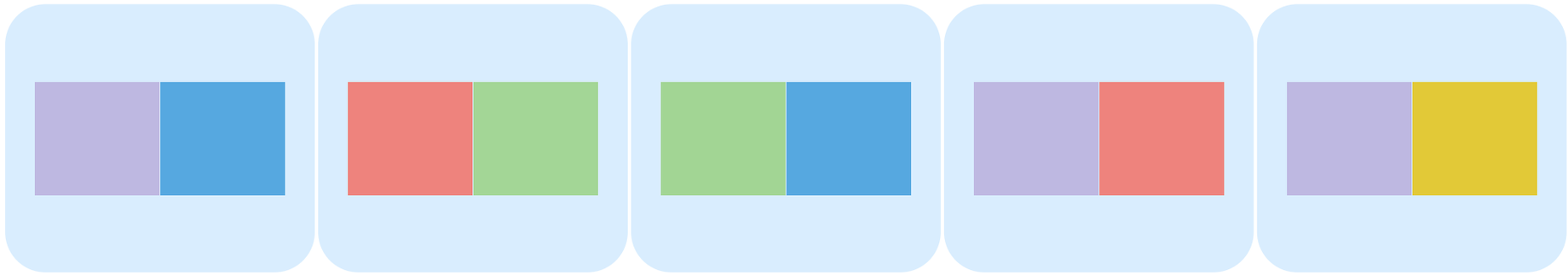
Our hash function:

{G,P} -> 1

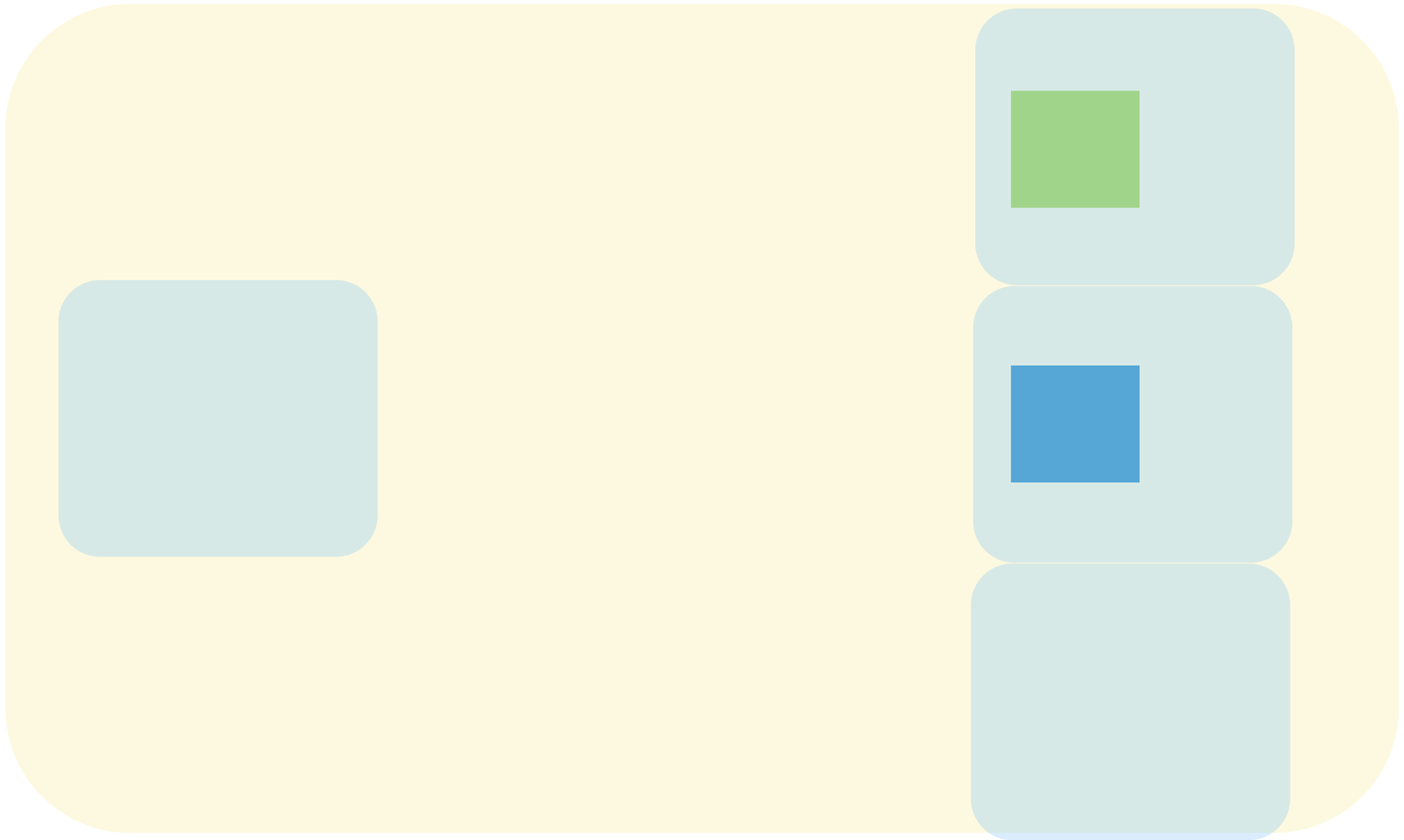
{B} -> 2

{R, Y} -> 3

Pass 1: Divide



N=6, B=4



Assign colors to 3 partitions using hash function.

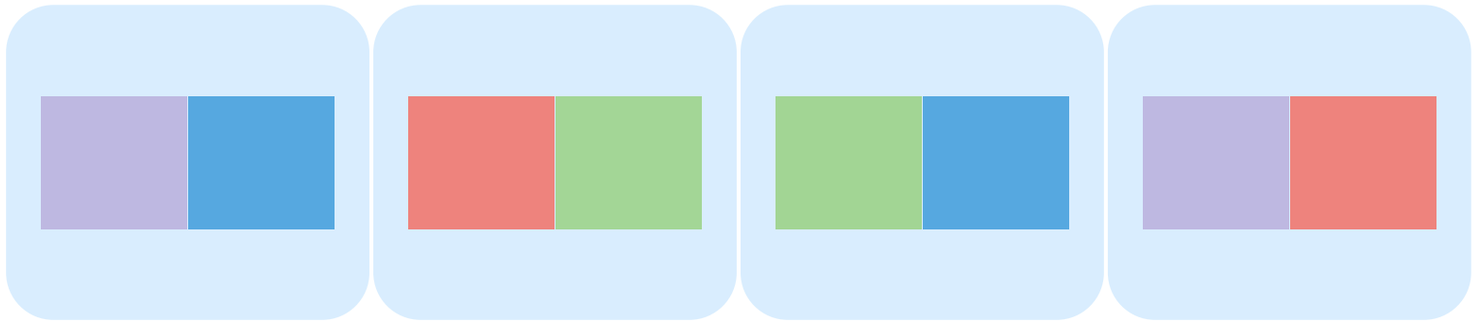
Our hash function:

{G,P} -> 1

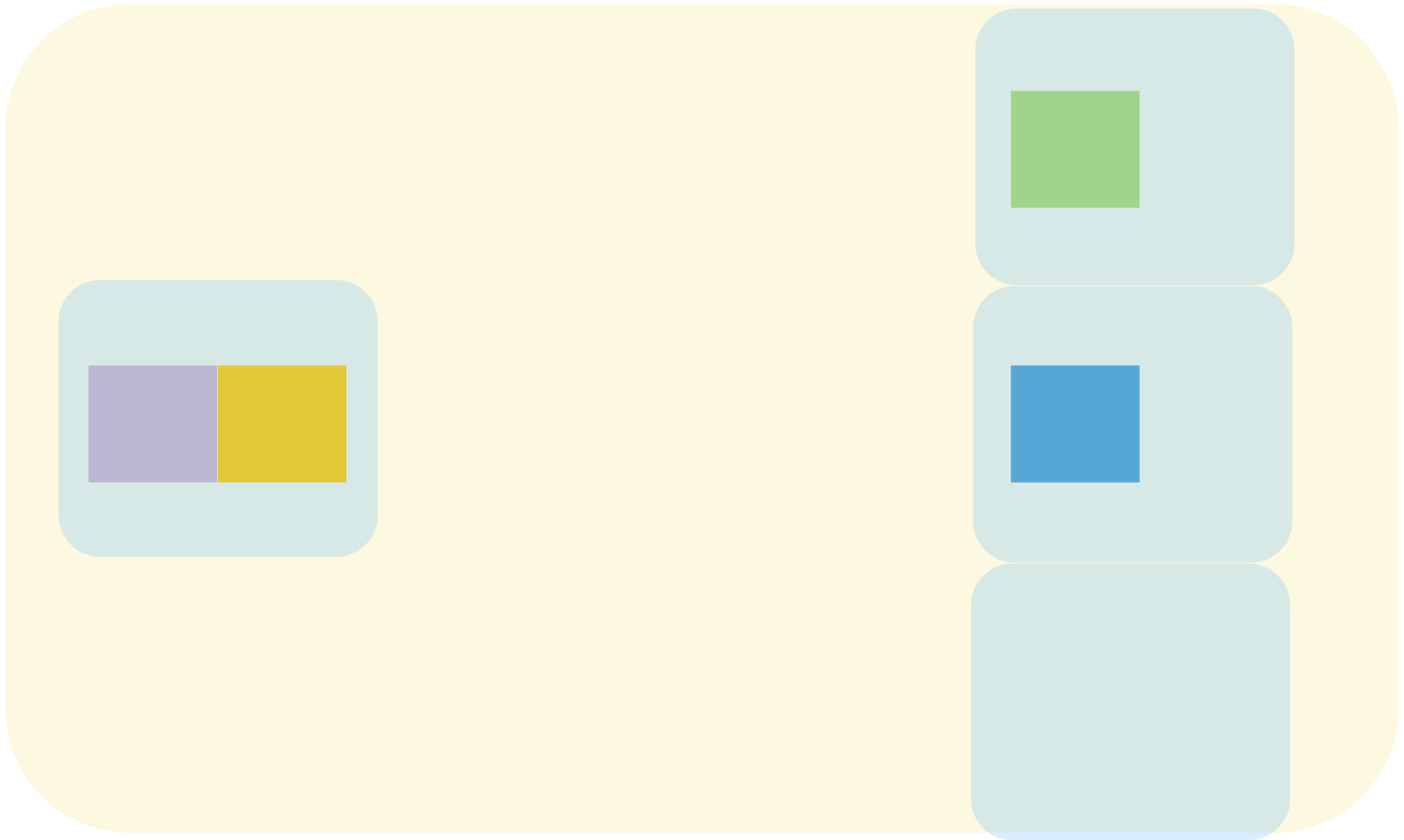
{B} -> 2

{R, Y} -> 3

Pass 1: Divide



N=6, B=4



Assign colors to 3 partitions using hash function.

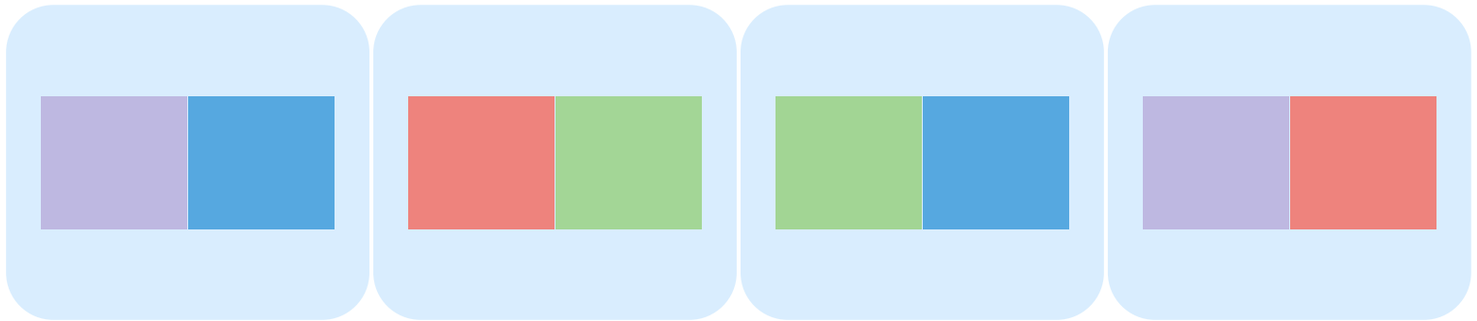
Our hash function:

{G,P} -> 1

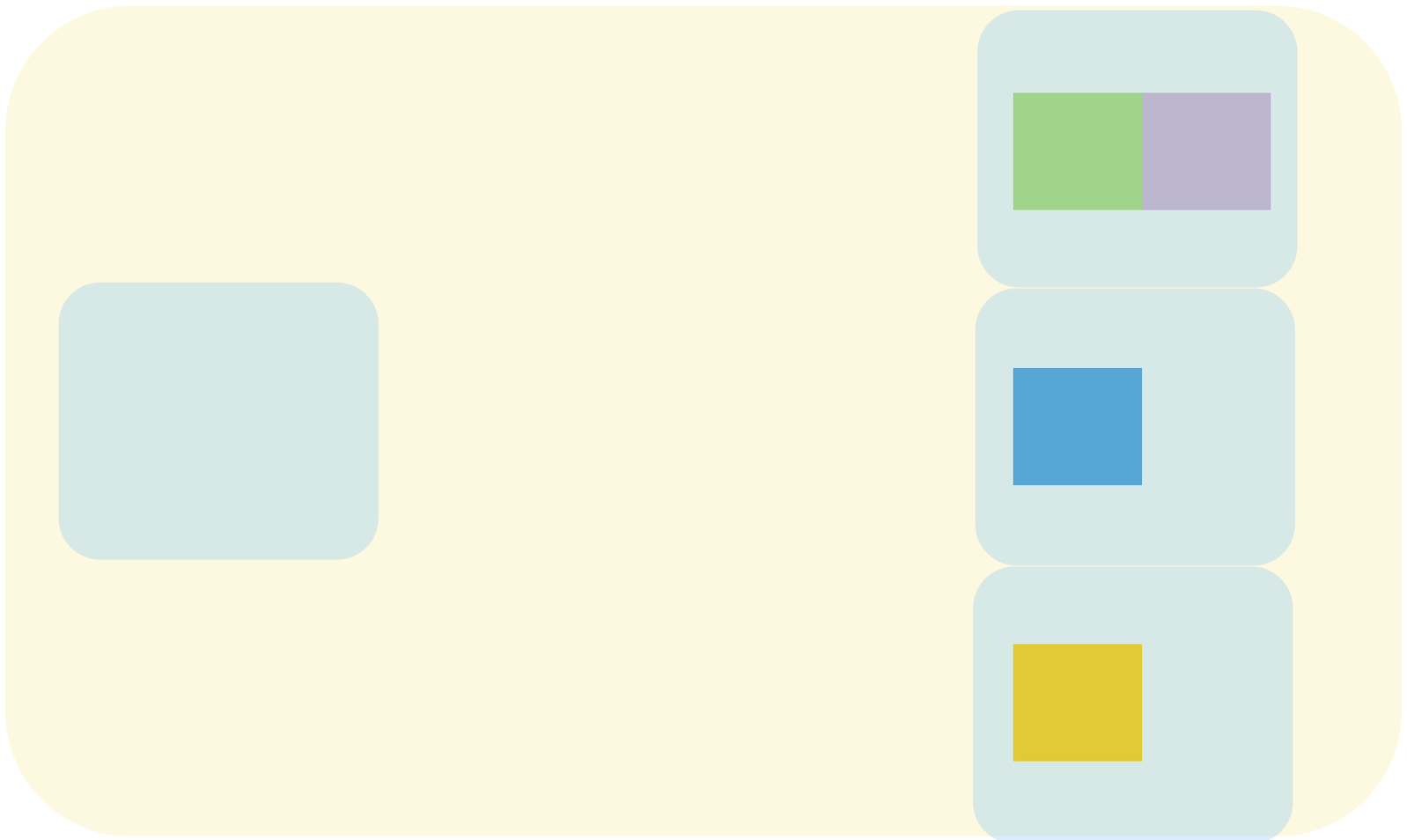
{B} -> 2

{R, Y} -> 3

Pass 1: Divide



N=6, B=4



Assign colors to 3 partitions using hash function.

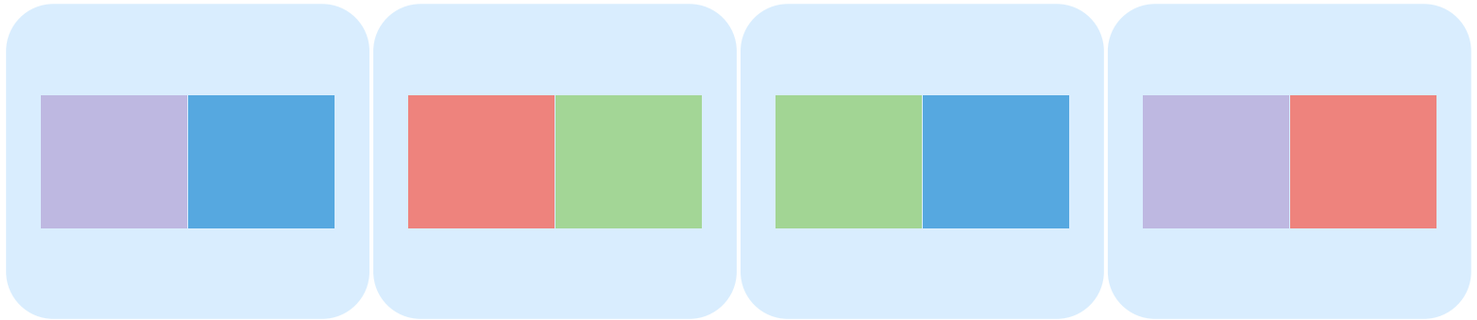
Our hash function:

{G,P} -> 1

{B} -> 2

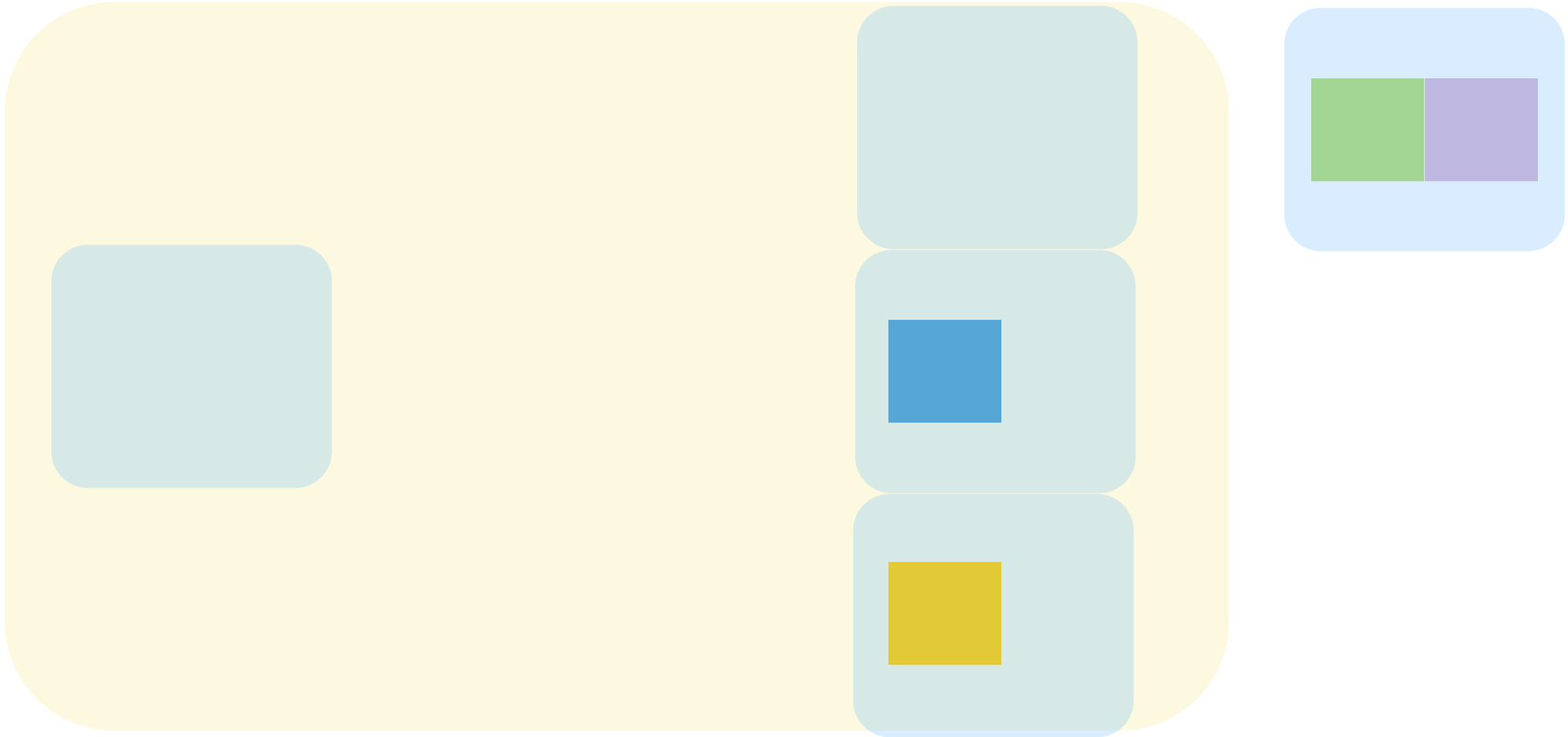
{R, Y} -> 3

Pass 1: Divide

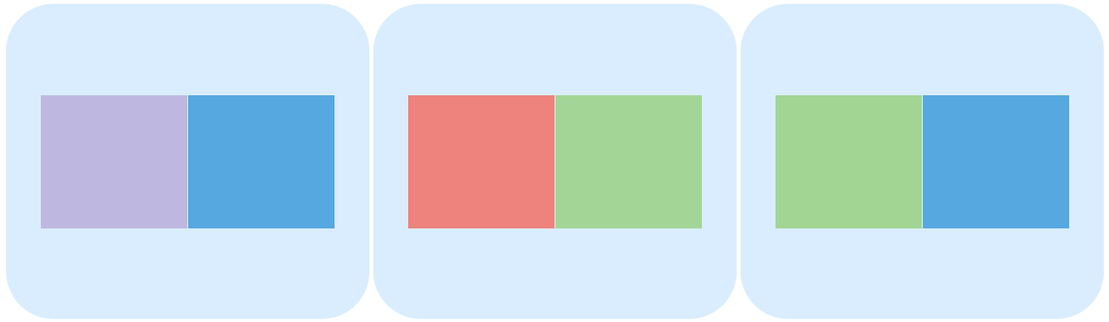


N=6, B=4

Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3

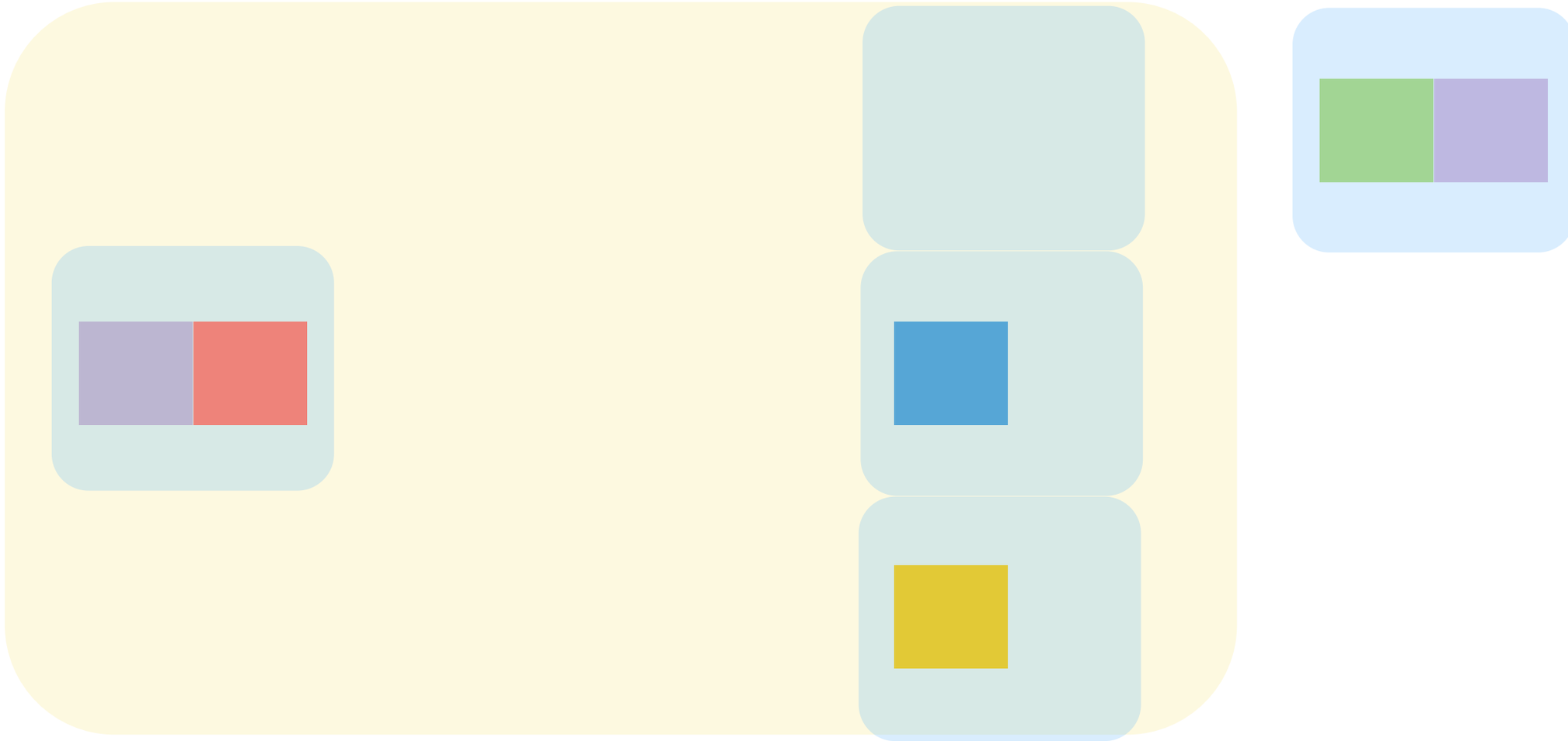


Pass 1: Divide

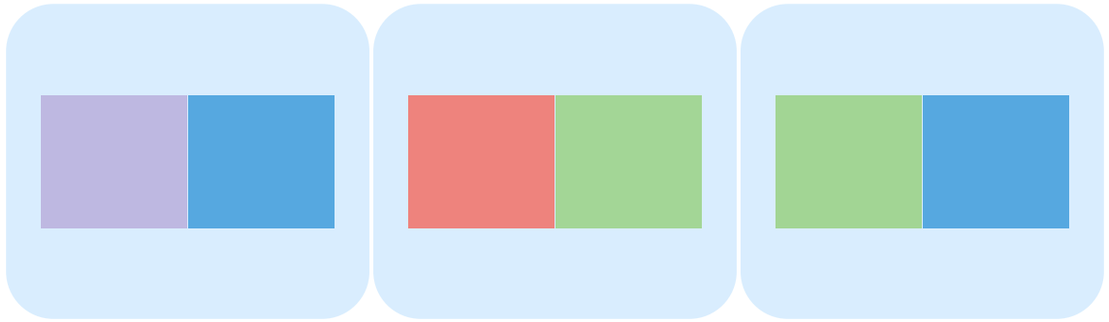


N=6, B=4

Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3

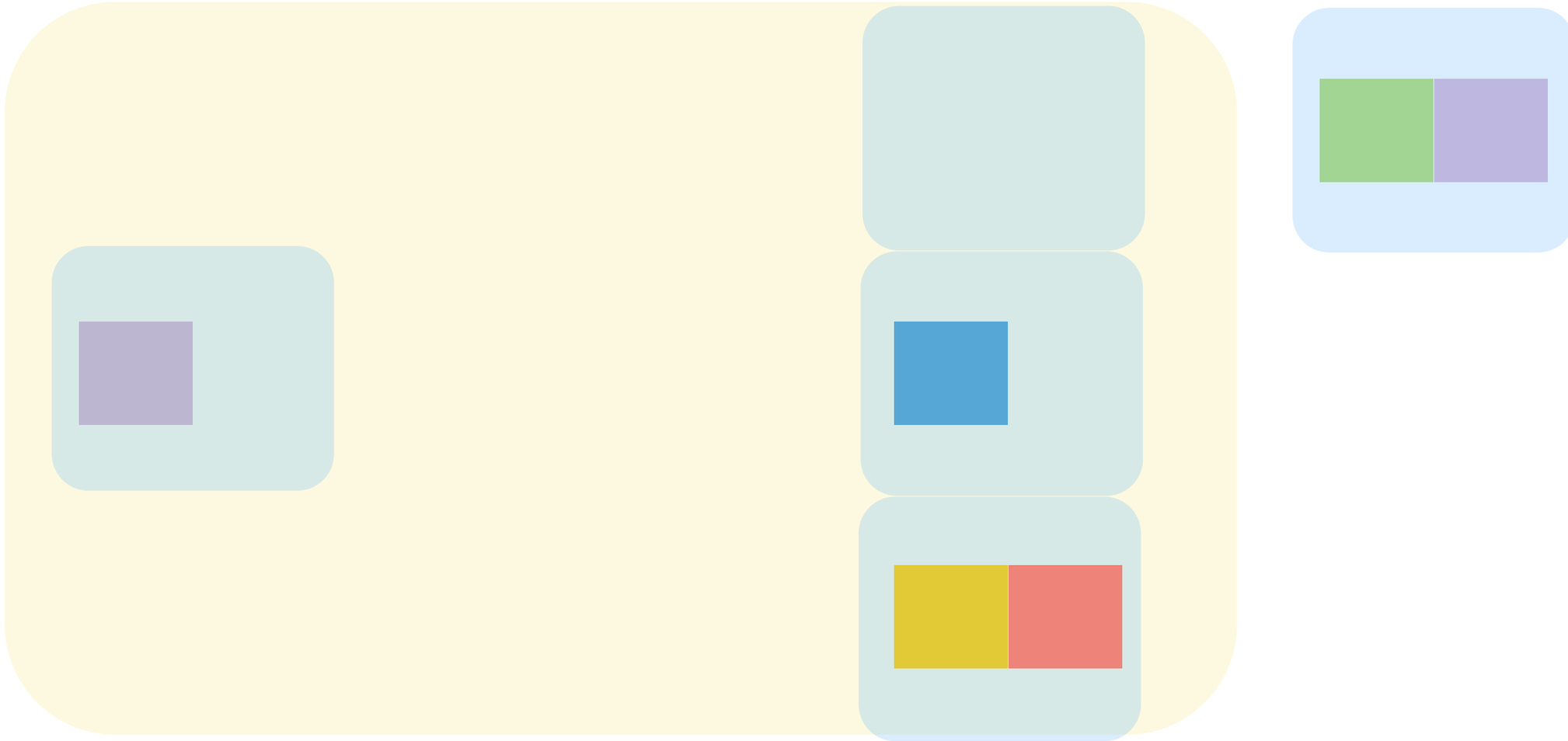


Pass 1: Divide

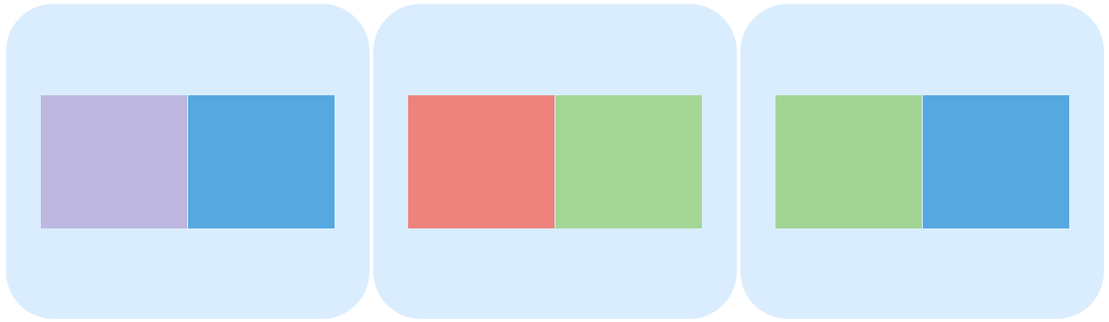


N=6, B=4

Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3

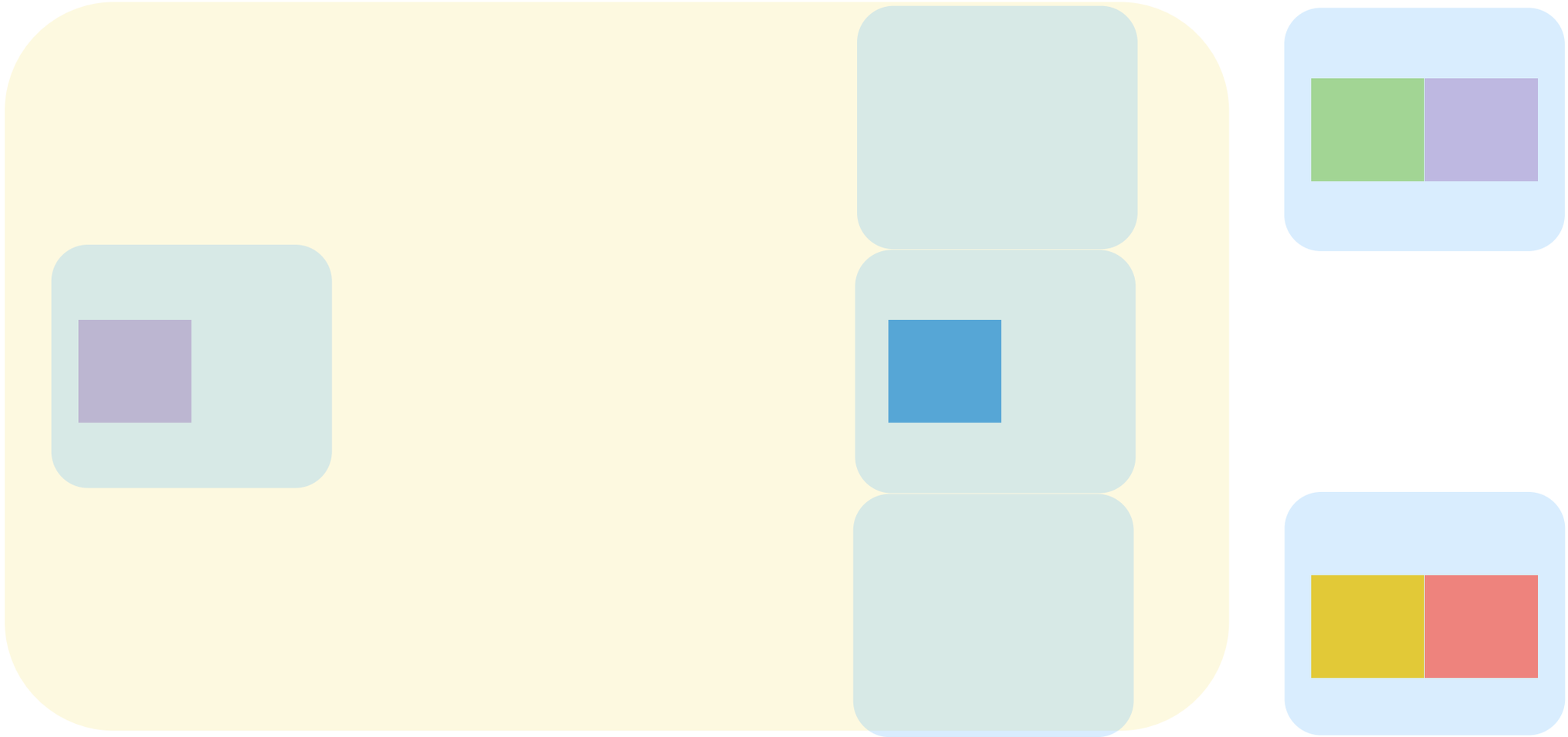


Pass 1: Divide

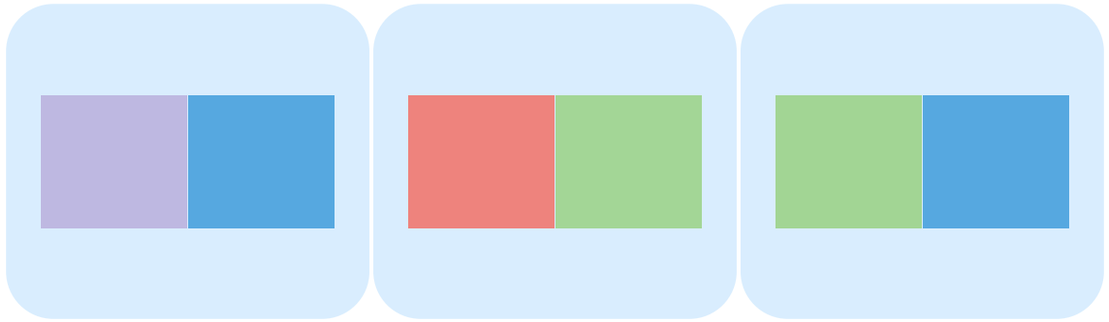


N=6, B=4

Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3

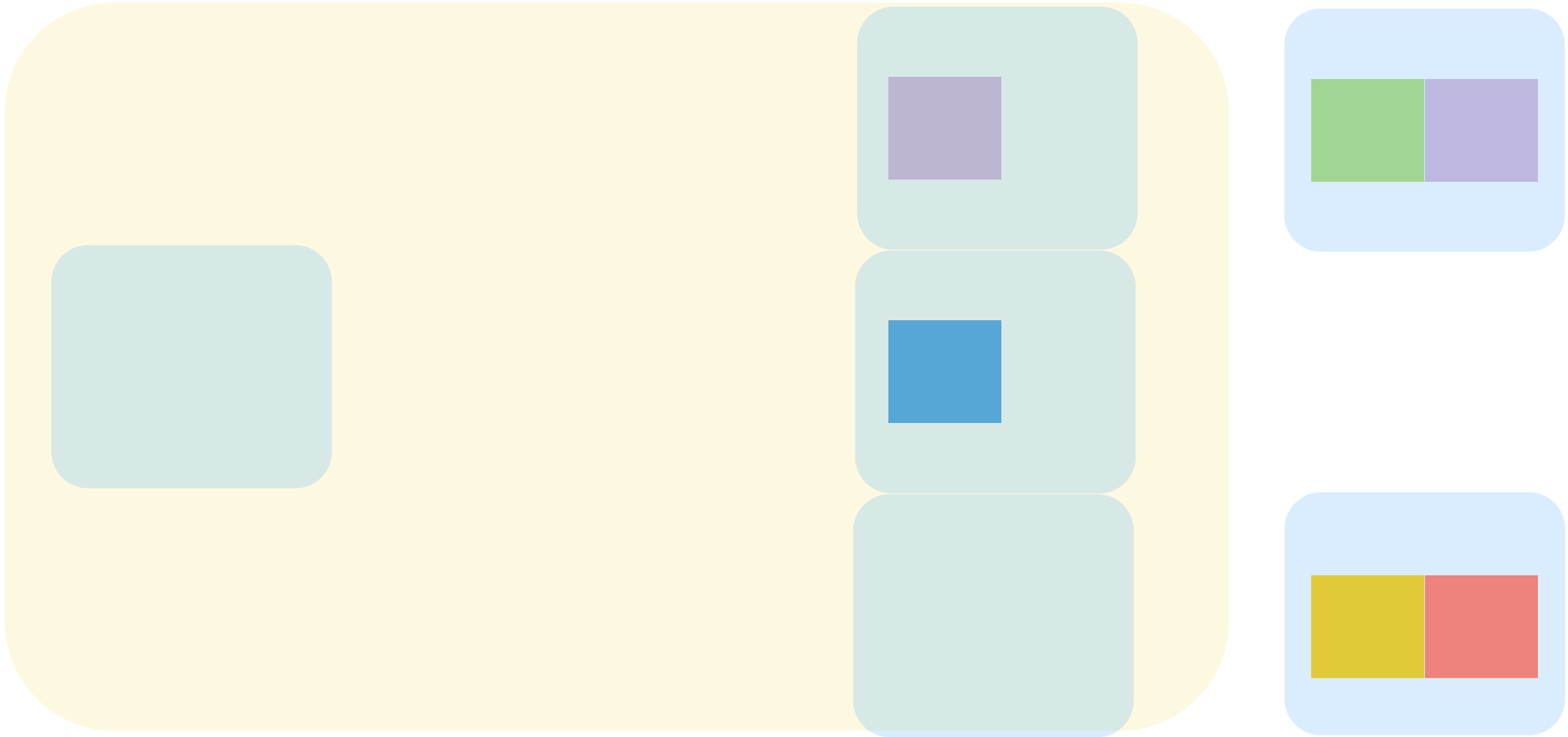


Pass 1: Divide

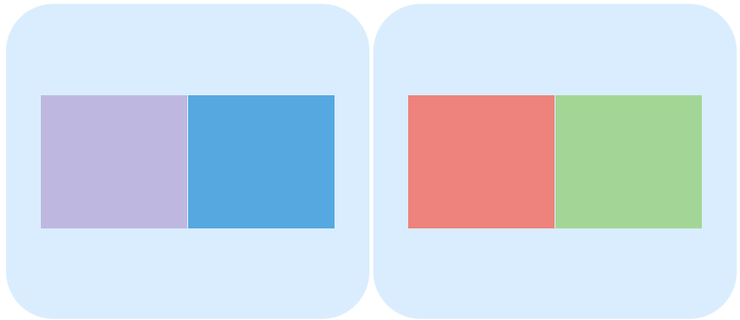


N=6, B=4

Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3

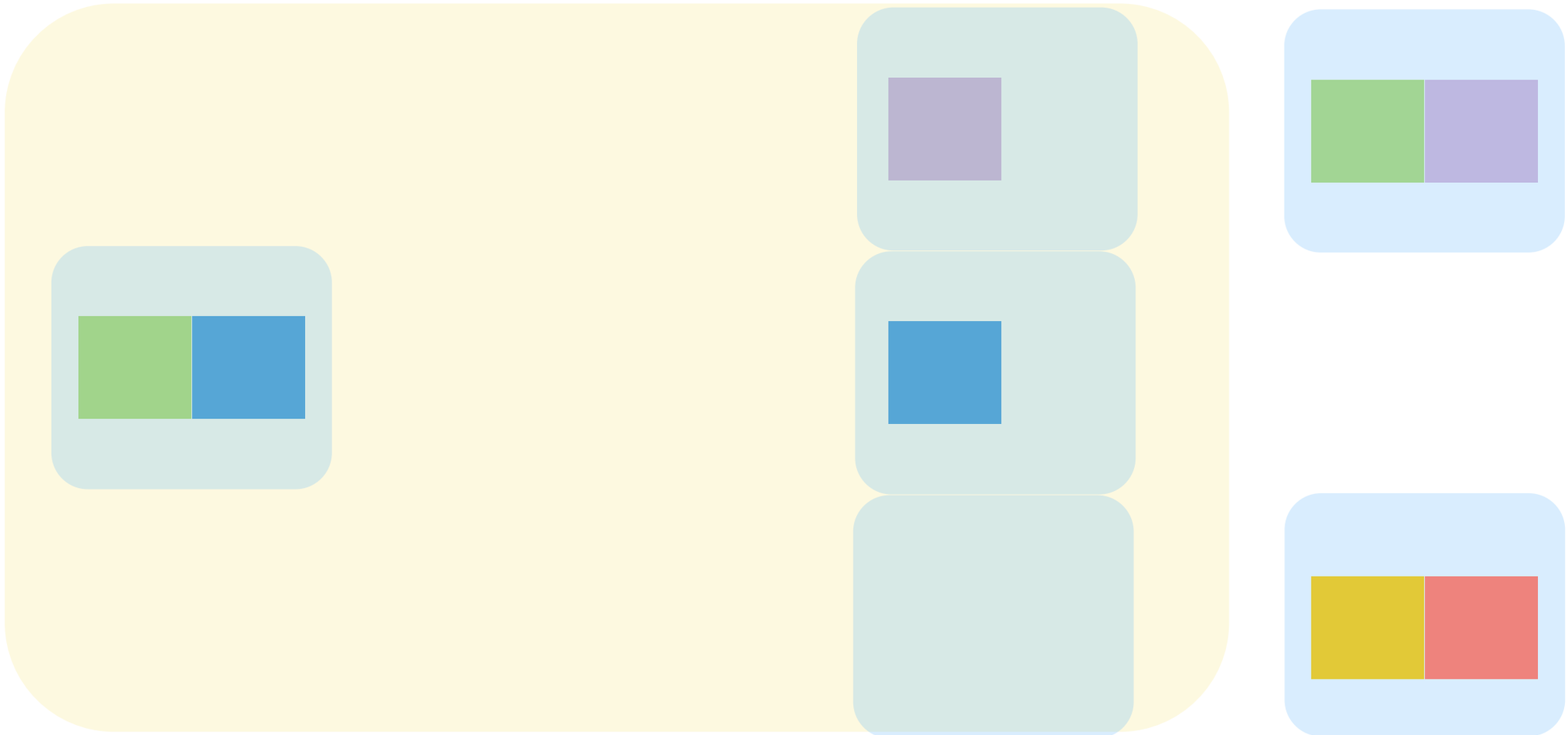


Pass 1: Divide

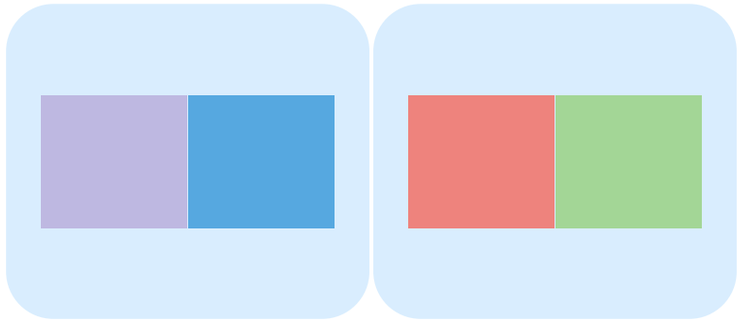


N=6, B=4

Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3

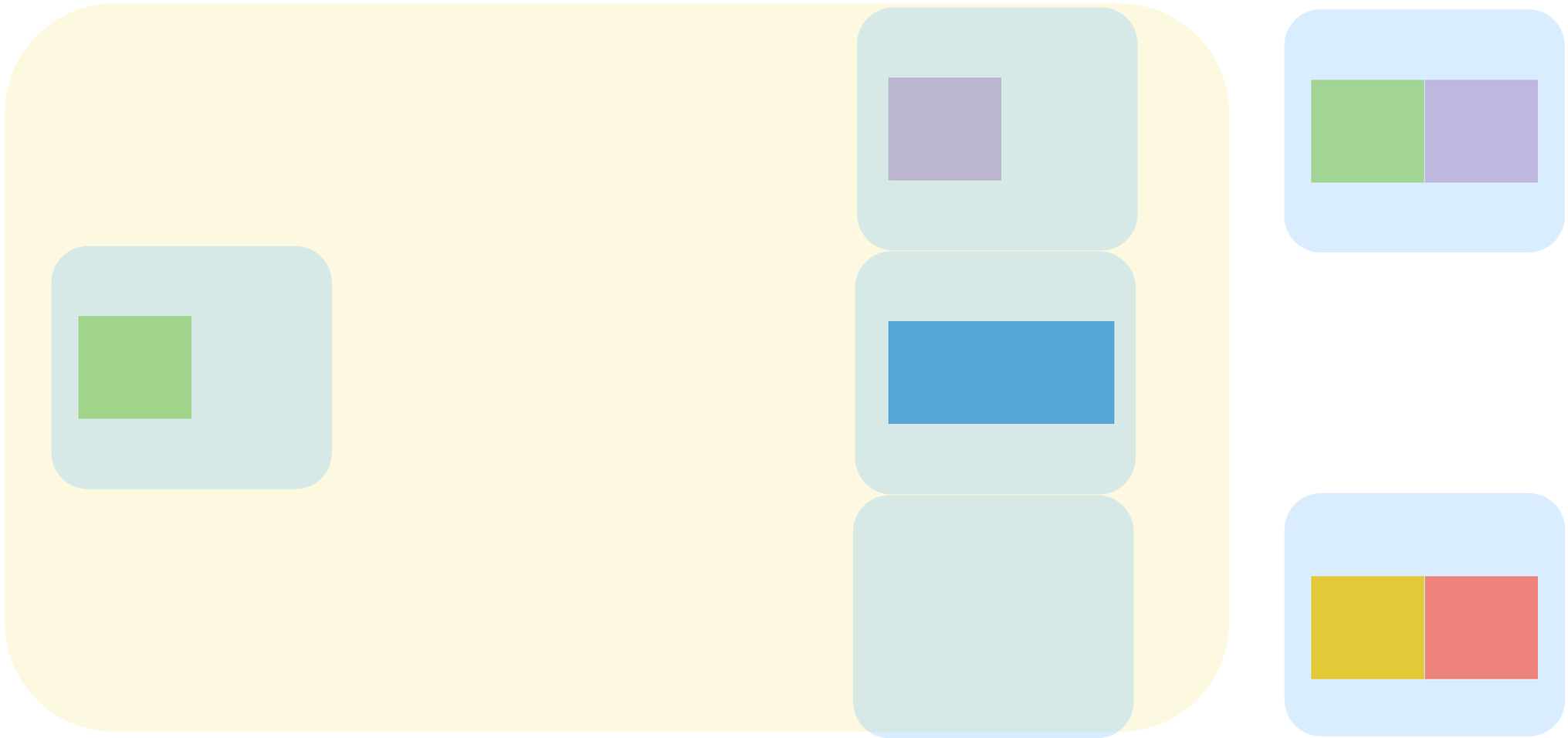


Pass 1: Divide

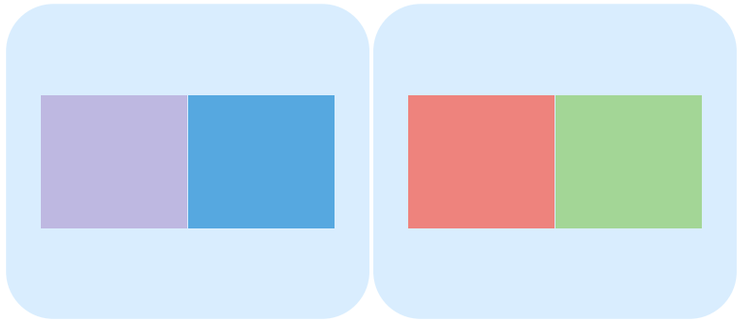


N=6, B=4

Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3

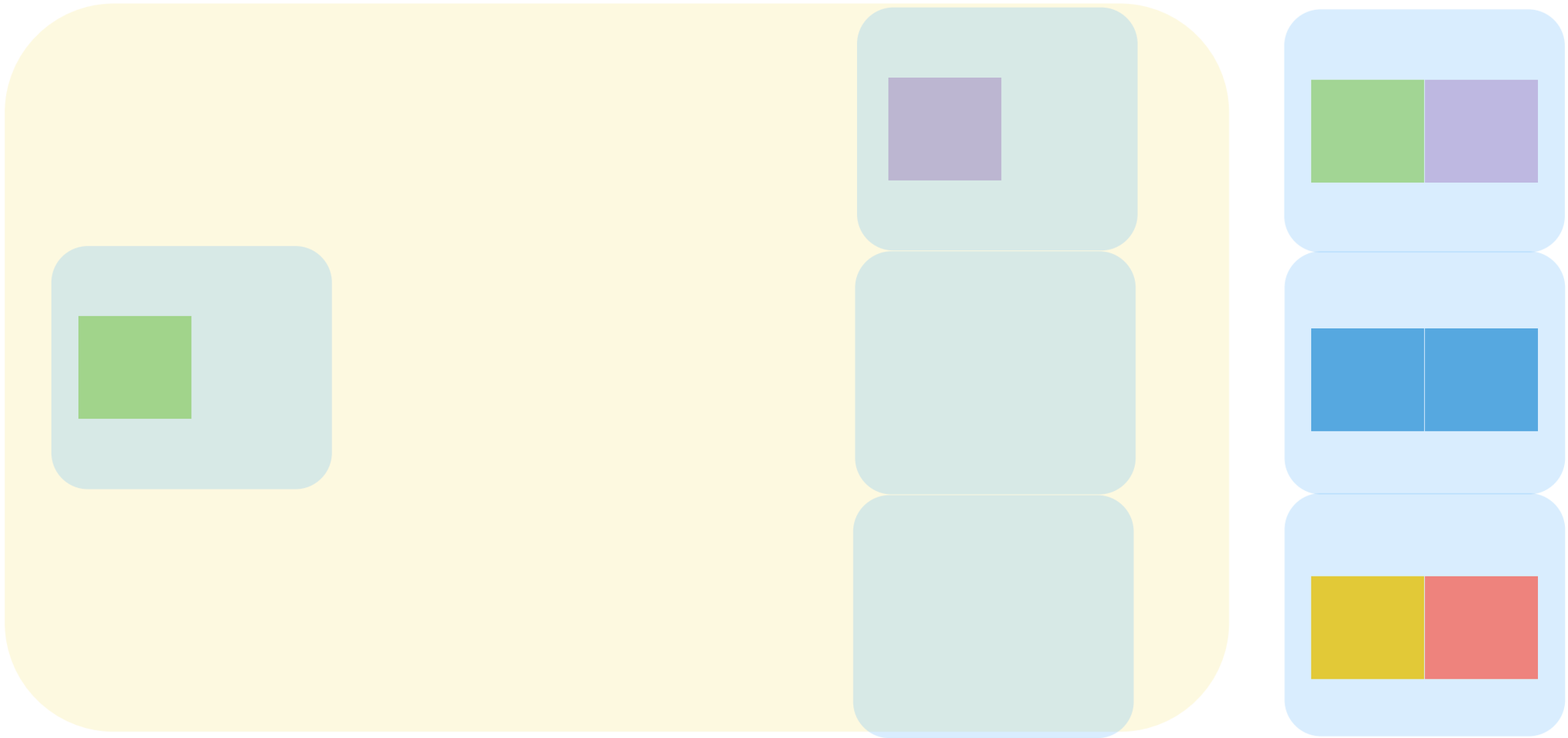


Pass 1: Divide

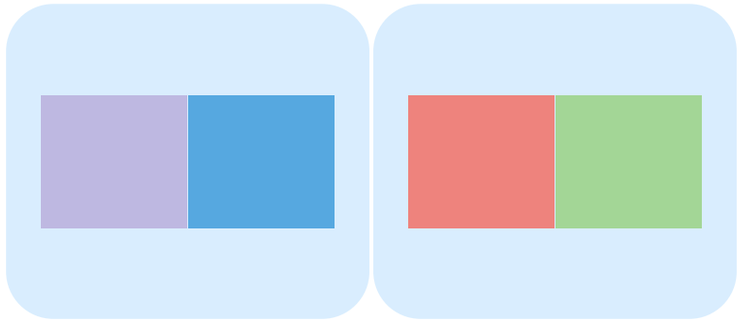


N=6, B=4

Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3

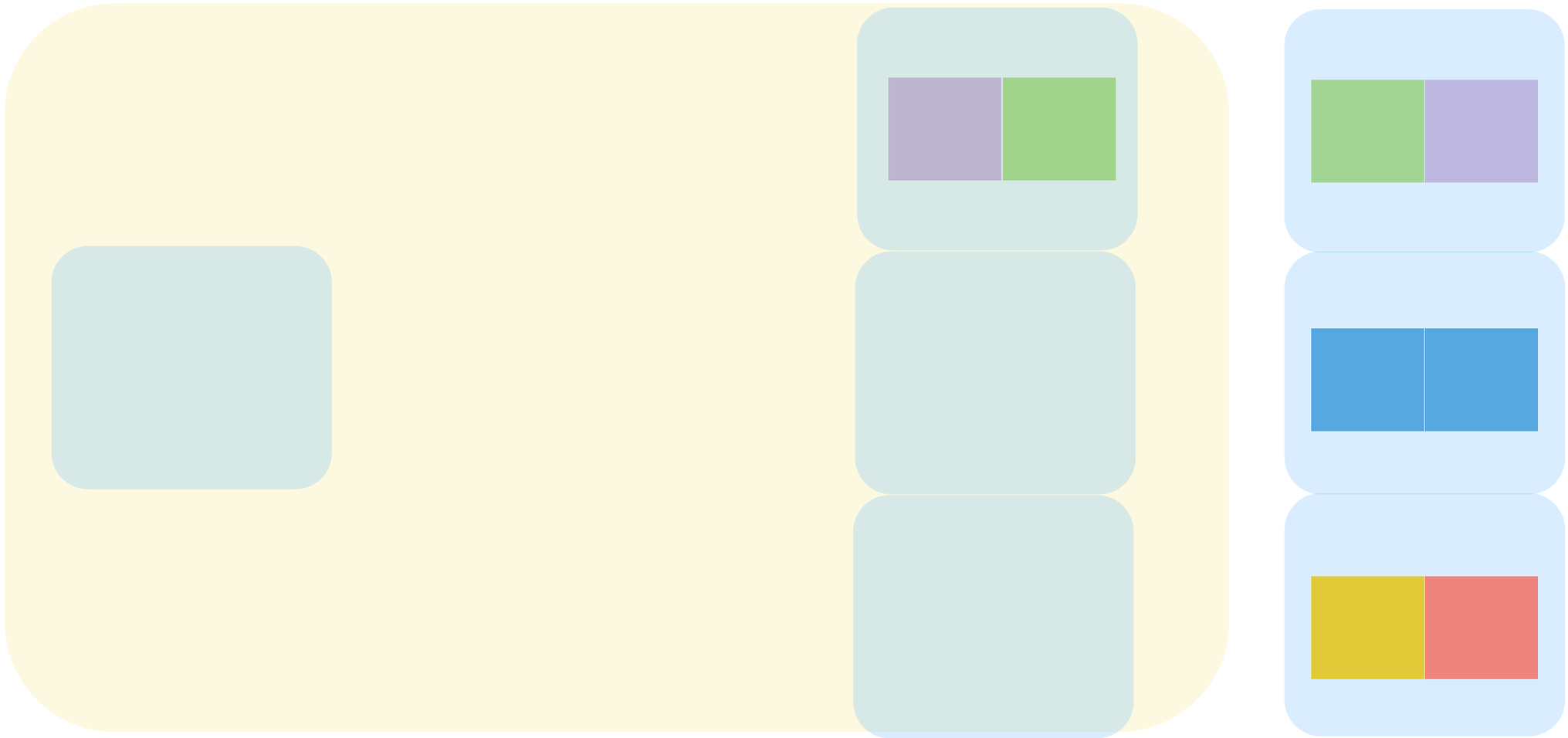


Pass 1: Divide

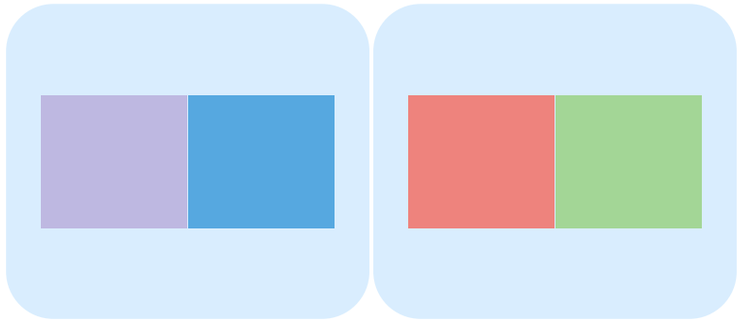


N=6, B=4

Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3

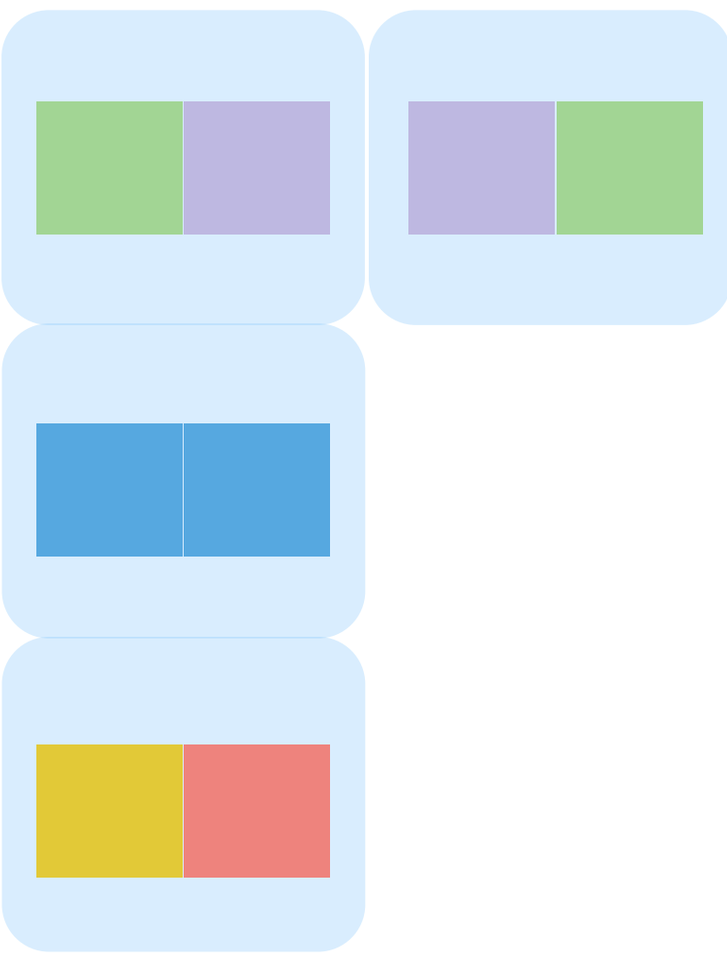


Pass 1: Divide

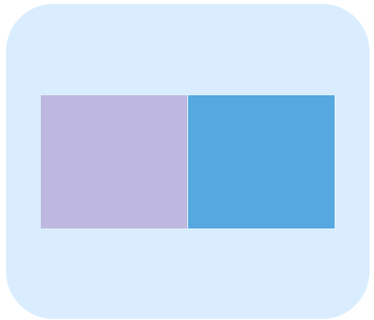


N=6, B=4

Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3

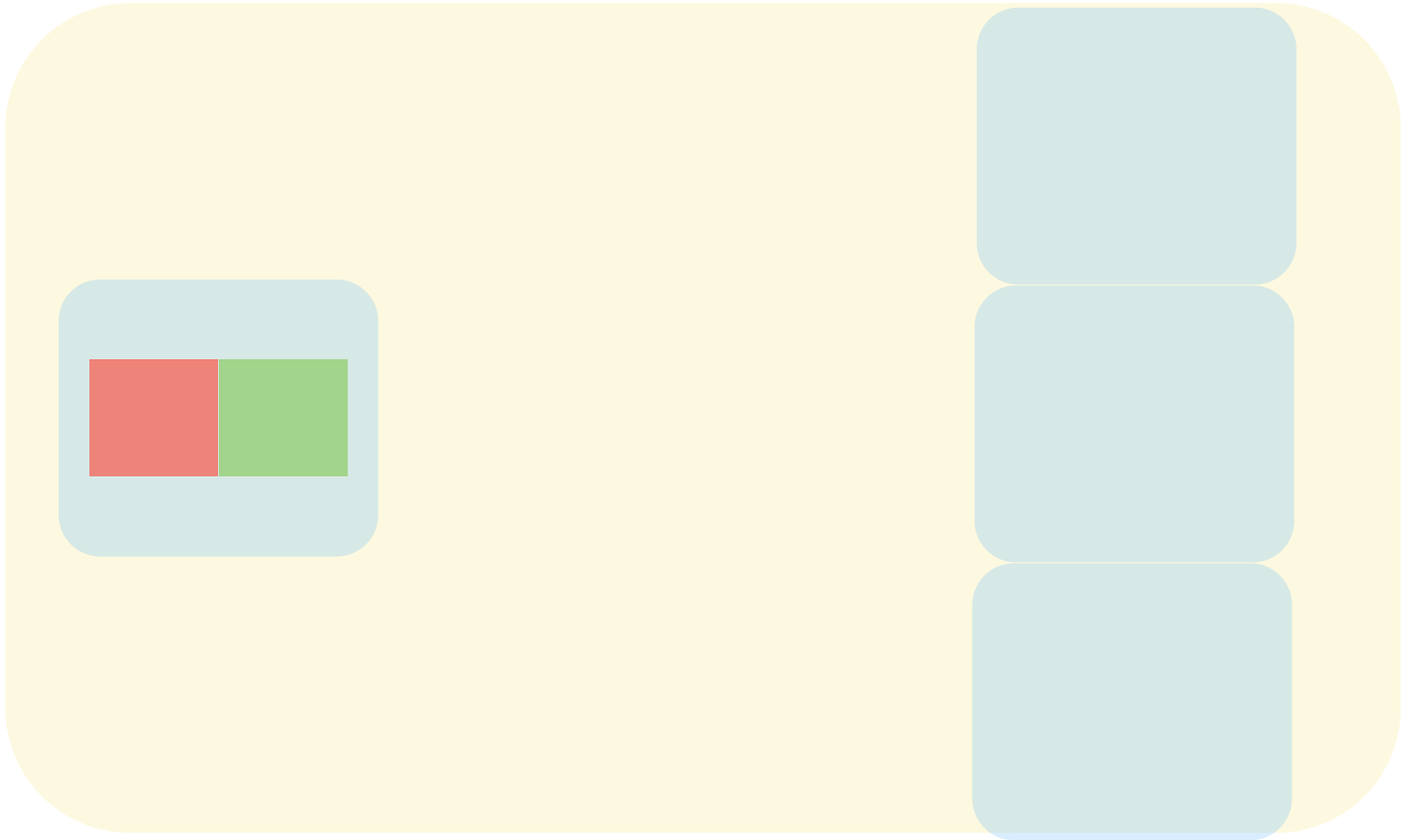


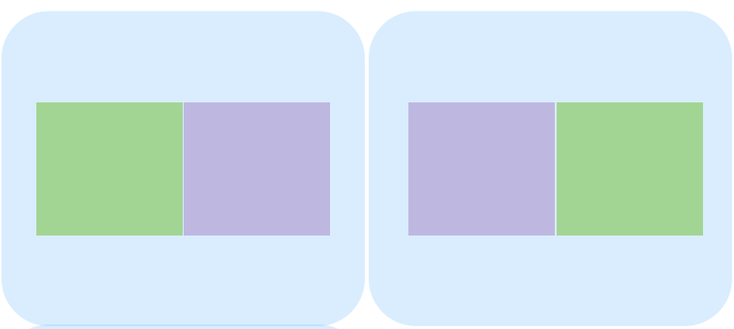
Pass 1: Divide

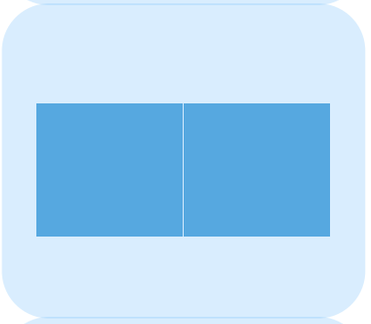


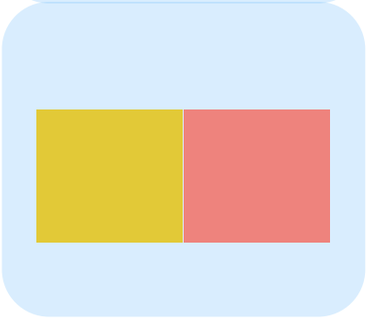
N=6, B=4

Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3

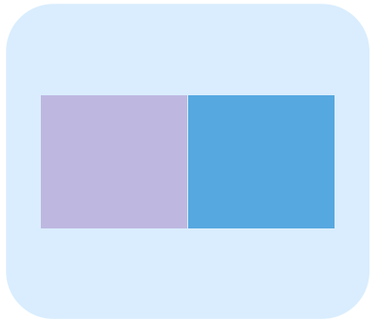






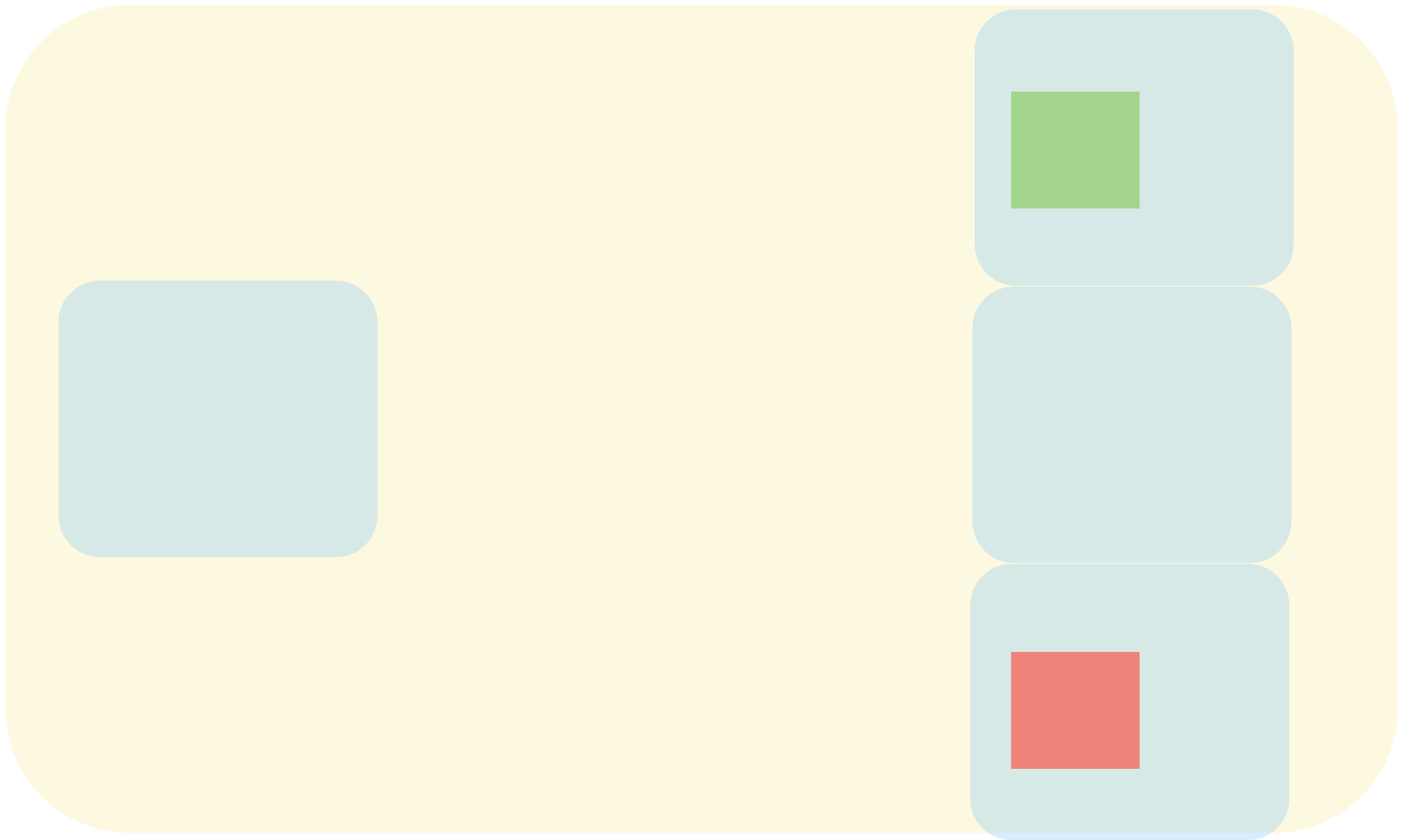


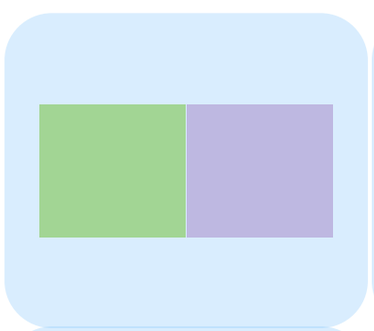
Pass 1: Divide

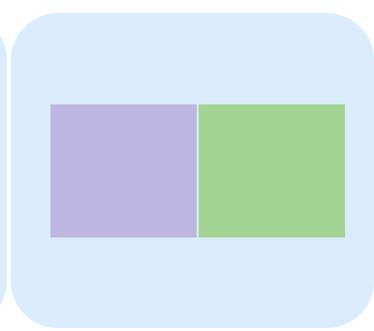


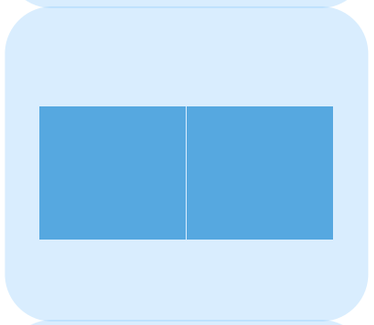
N=6, B=4

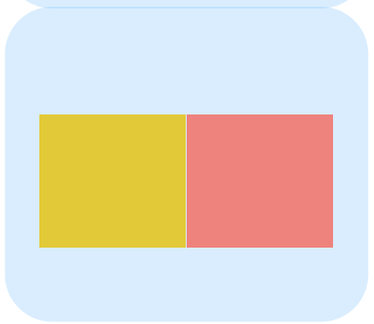
Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3





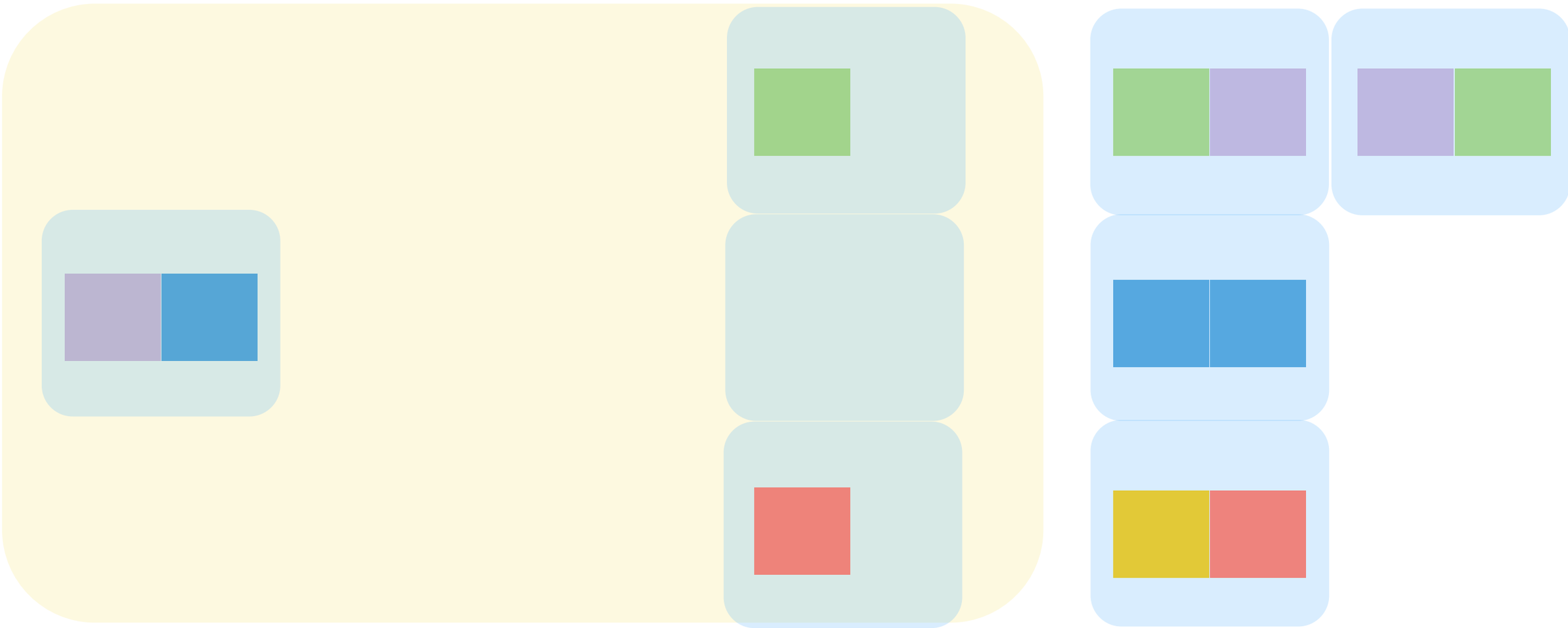






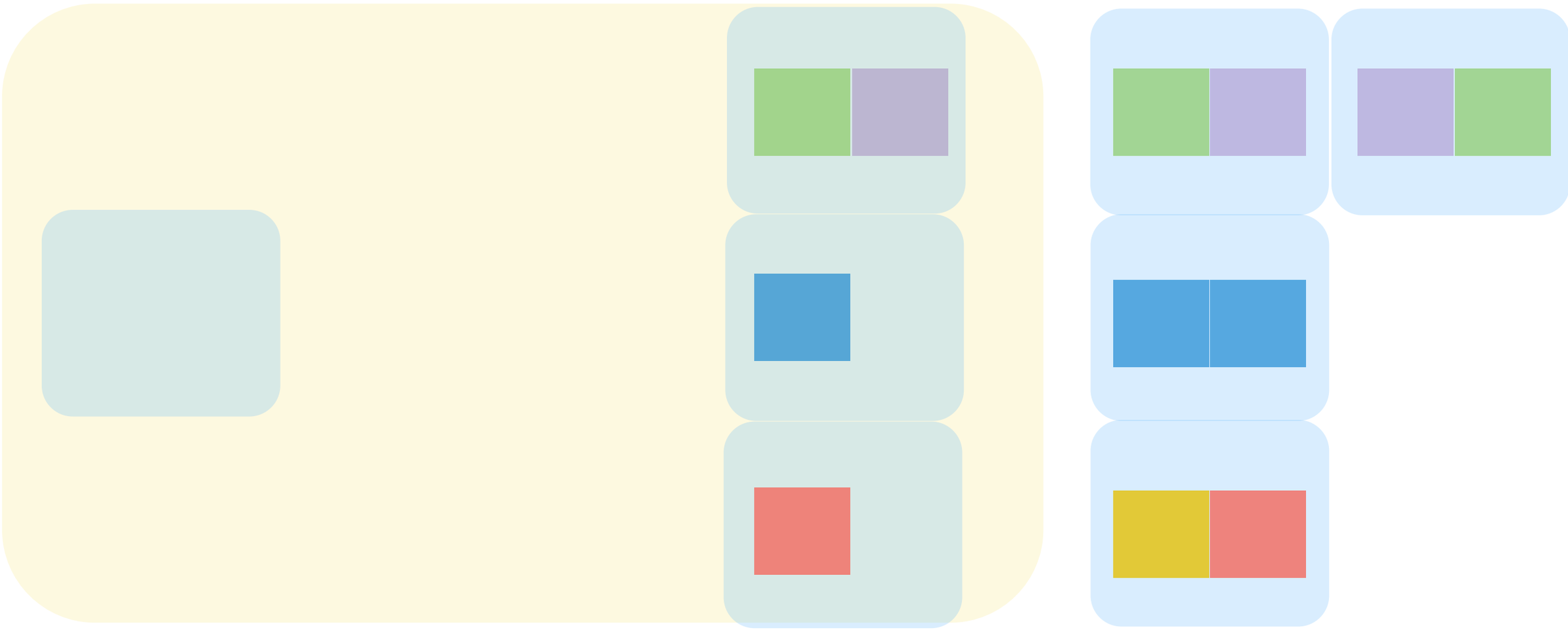
Pass 1: Divide

N=6, B=4 Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3



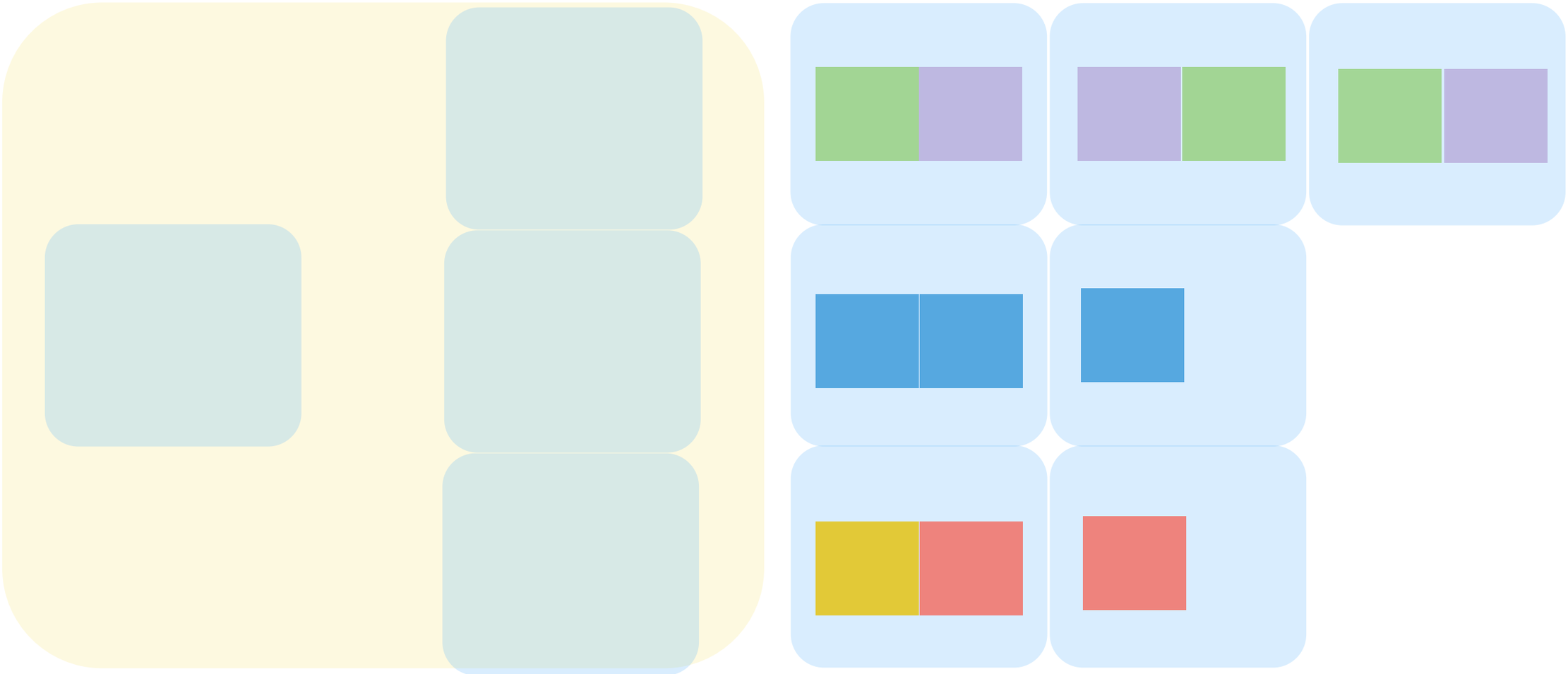
Pass 1: Divide

N=6, B=4 Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3



Pass 1: Divide

N=6, B=4 Our hash function: {G,P} -> 1, {B} -> 2,{R, Y} -> 3



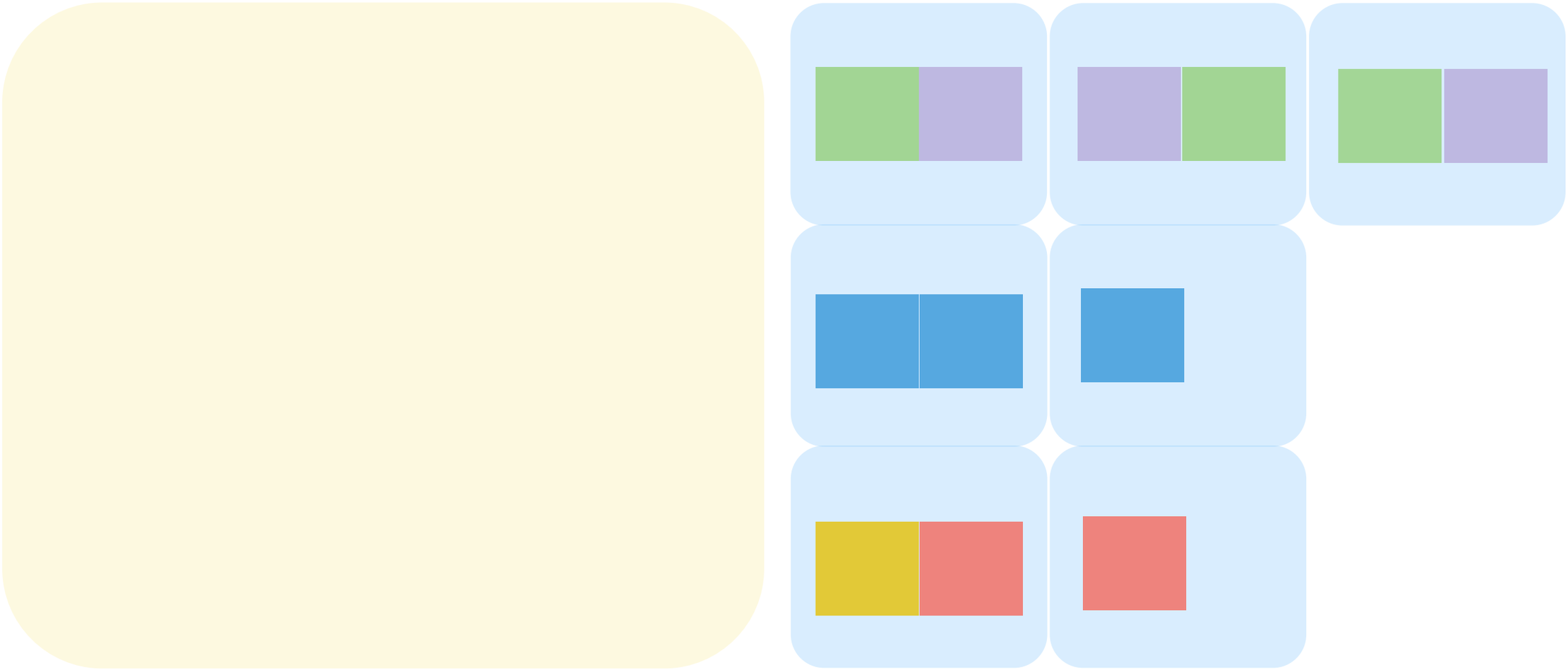
Pass 2: Conquer

- Rehash each partition.
- For a partition to fit in memory, it can only have B pages.
- To hash larger tables, use the partition algorithm recursively until the partition fits into memory
- # I/O's = 2N

Pass 2: Conquer

Create in-memory table for each partition.

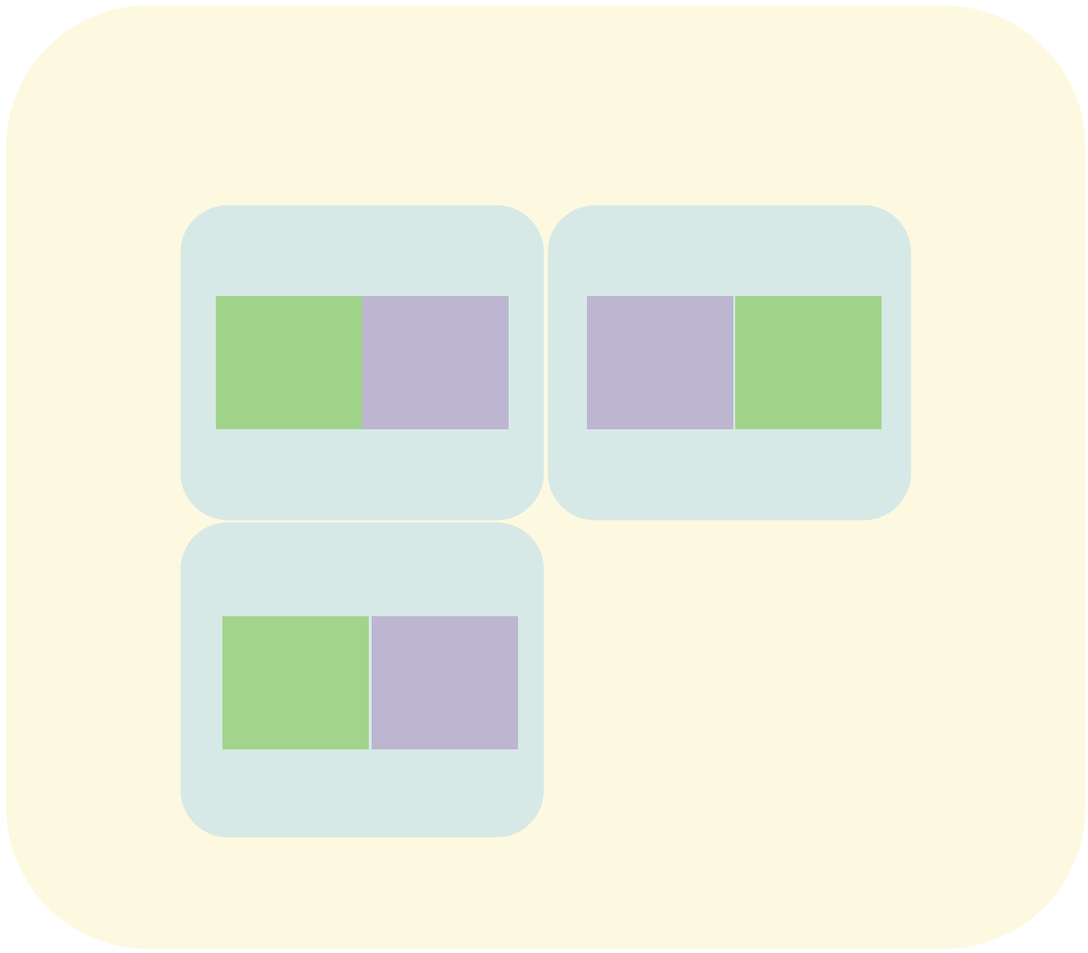
N=6, B=4

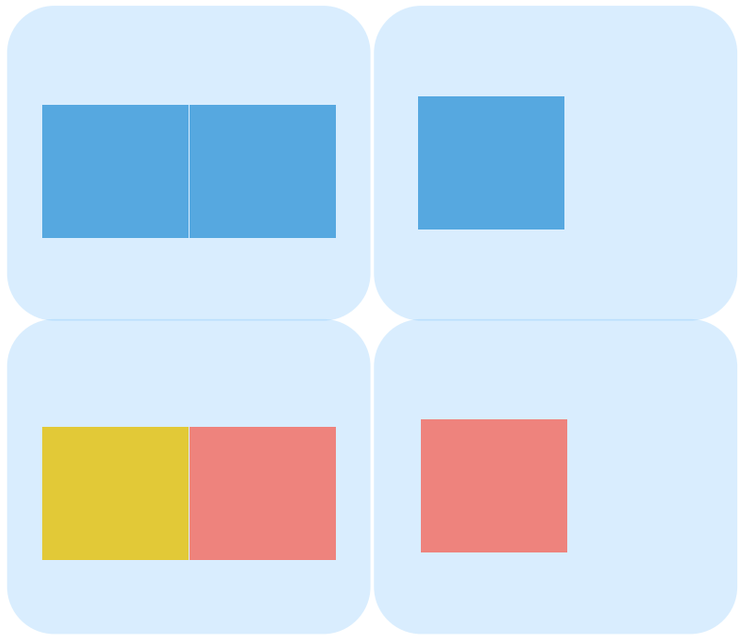


Pass 2: Conquer

Create in-memory table for each partition.

N=6, B=4





Pass 2: Conquer

Create in-memory table for each partition.

N=6, B=4

Green

Purple