Introduction à l’informatique CM4

Antonio E. Porreca
aeporreca.org/introinfo

Opérations sur les tableaux

Ajouter des éléments au fond d'un tableau

```
def copier(A):
    n = len(A)
    B = []
    for i in range(n):
        B.append(A[i])
    return B
```

Ajouter des éléments au fond d'un tableau

```
def copier(A):
    n = len(A)
    B = []
    for i in range(n):
        B.append(A[i])
    return B
```

Parcourir un tableau au contraire

```
def inverser(A):
    n = len(A)
    B = []
    for i in reversed(range(n)):
        B.append(A[i])
    return B
```

Parcourir un tableau au contraire

```
def inverser(A):
    n = len(A)
    B = []
    for i in reversed(range(n)):
        B.append(A[i])
    return B
```

intervalle
n-1, n-2, …, 0

Créer un tableau de taille donnée

```
def intervalle(n):
    A = [0 for i in range(n)]
    for i in range(n):
        A[i] = i
    return A
```

Créer un tableau de taille donnée

```
def intervalle(n):
    A = [0 for i in range(n)]
    for i in range(n):
        A[i] = i
    return A
```

tableau de longueur `n` rempli de `0`

Matrices

```
M = [[1,  2,  3,  4],
     [5,  6,  7,  8],
     [9, 10, 11, 12]]
```

```
>>> M[0]
[1, 2, 3, 4]
>>> M[1]
[5, 6, 7, 8]
>>> M[1][2]
7
>>> M[0][3]
4
```

Matrices

une matrice
est un tableau
de tableaux

```
M = [[1,  2,  3,  4],
     [5,  6,  7,  8],
     [9, 10, 11, 12]]
```

```
>>> M[0]
[1, 2, 3, 4]
>>> M[1]
[5, 6, 7, 8]
>>> M[1][2]
7
>>> M[0][3]
4
```

Matrices

```
M = [[1,  2,  3,  4],
     [5,  6,  7,  8],
     [9, 10, 11, 12]]
```

une matrice est un tableau de tableaux

accéder à une ligne de la matrice

```
>>> M[0]
[1, 2, 3, 4]
>>> M[1]
[5, 6, 7, 8]
>>> M[1][2]
7
>>> M[0][3]
4
```

Matrices

```
M = [[1,  2,  3,  4],
     [5,  6,  7,  8],
     [9, 10, 11, 12]]
```

une matrice est un tableau de tableaux

```
>>> M[0]
[1, 2, 3, 4]
>>> M[1]
[5, 6, 7, 8]
>>> M[1][2]
7
>>> M[0][3]
4
```

accéder à une ligne de la matrice

accéder à un élément spécifique

Algorithmes de tri

Algorithmes de tri pour accélérer la recherche dans un tableau

- La recherche dans un tableau non trié prend temps $O(n)$ avec la recherche séquentielle (ou linéaire)

- Par contre, on peut faire une recherche dichotomique dans un tableau trié en temps $O(\log_2 n)$

- Donc ça vaut la peine de trier le tableau si on a beaucoup de recherches à faire

Algorithmes de tri dans le commerce électronique

amazonie.fr

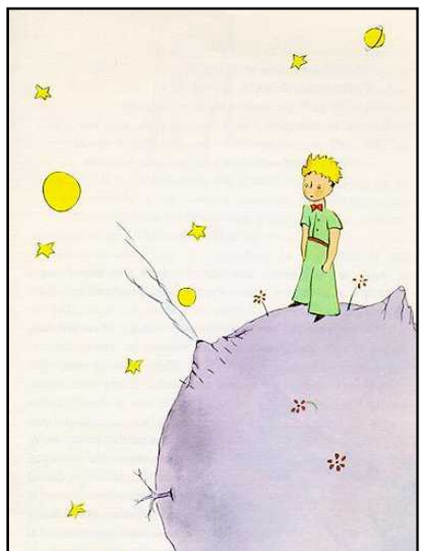
amazonie

Chercher : Le Petit Prince

Résultats 1–20 sur 928572785 pour « Le Petit Prince »

Trier par :

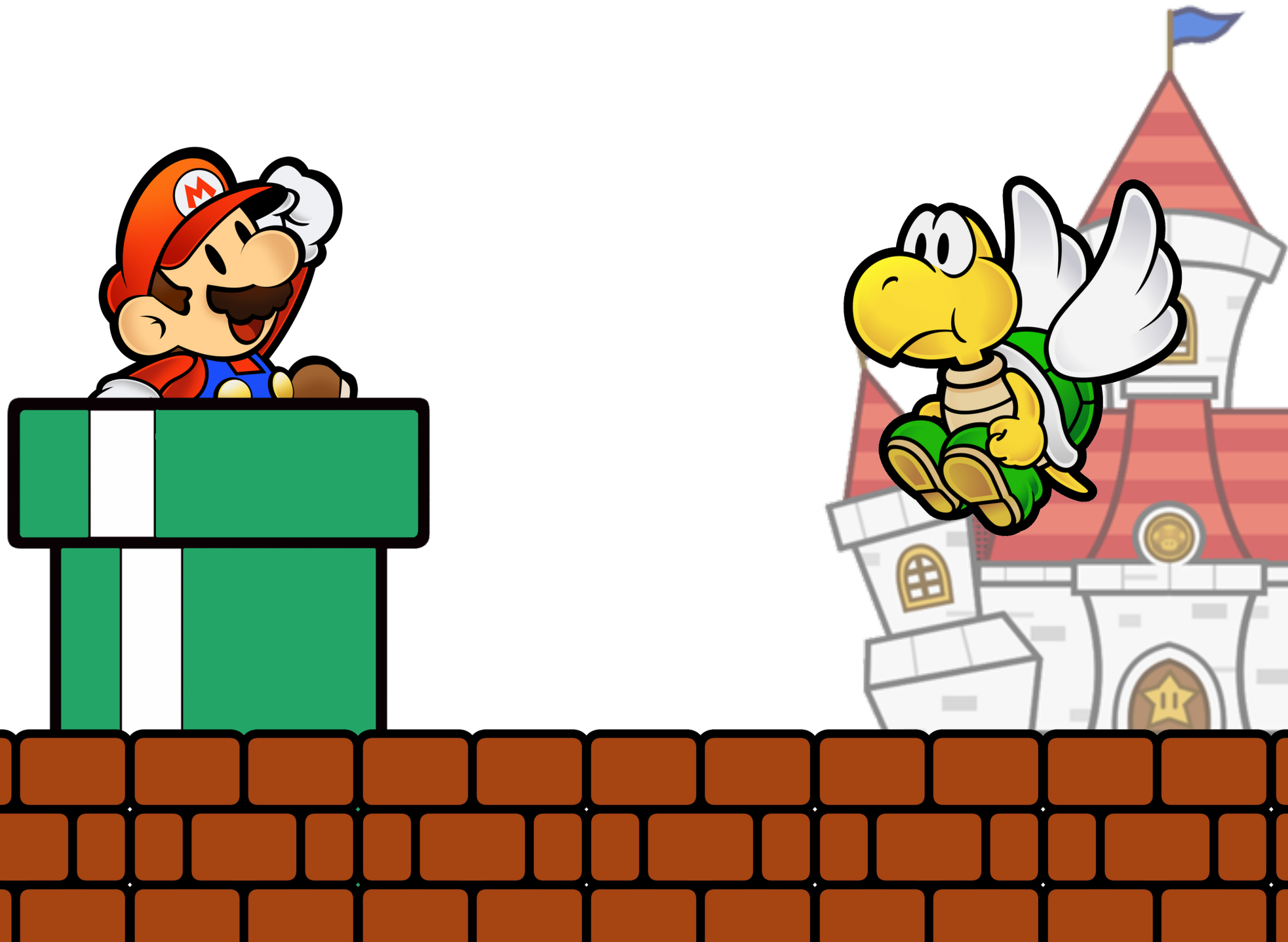
- prix croissant
- prix décroissant
- note moyenne
- nouveauté



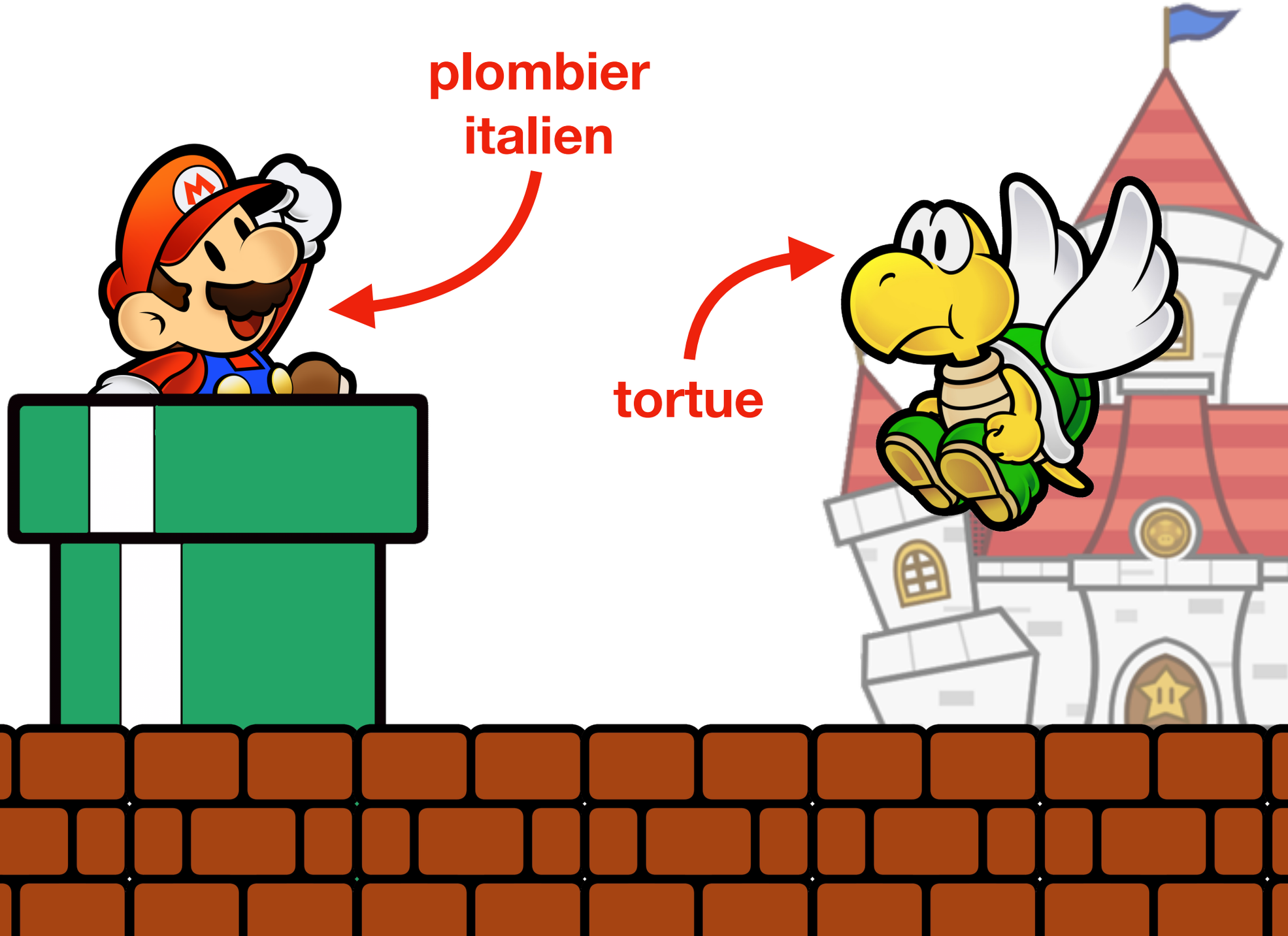
Le Petit Prince
de Antoine de Saint-Exu

Format poche 6,90 €
Format Kinder 6,49 €

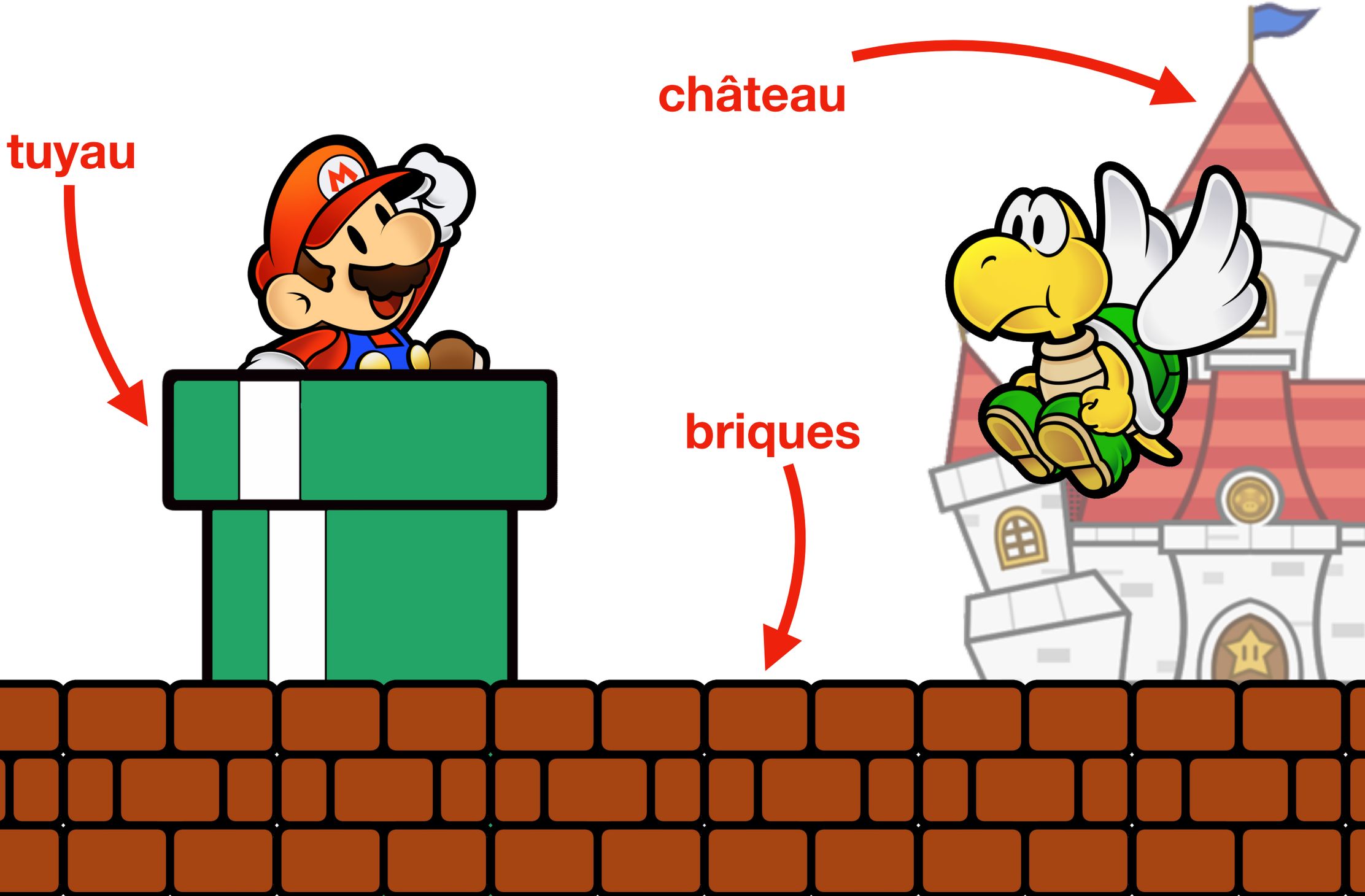
Algos de tri dans le jeux vidéo « Super Plombiers Italiens »



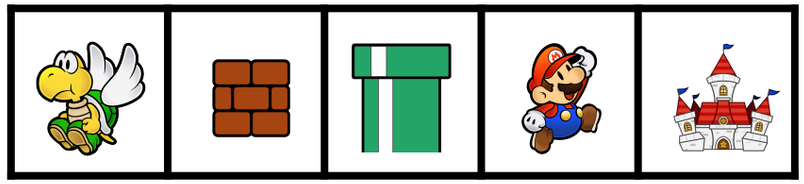
Algos de tri dans le jeux vidéo « Super Plombiers Italiens »



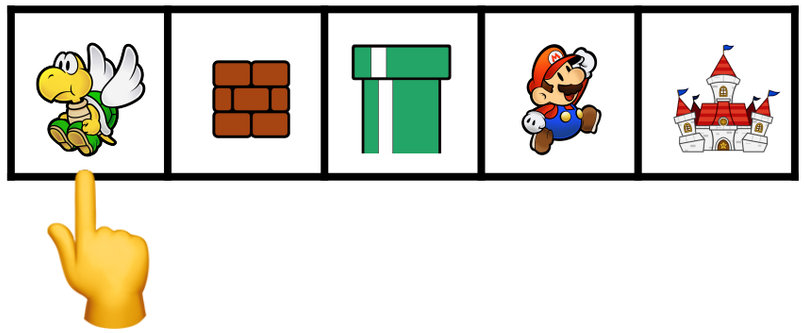
Algos de tri dans le jeux vidéo « Super Plombiers Italiens »



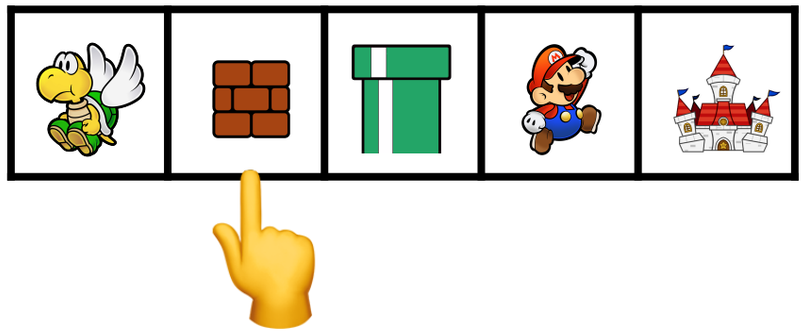
Affichage des objets dans le mauvais ordre



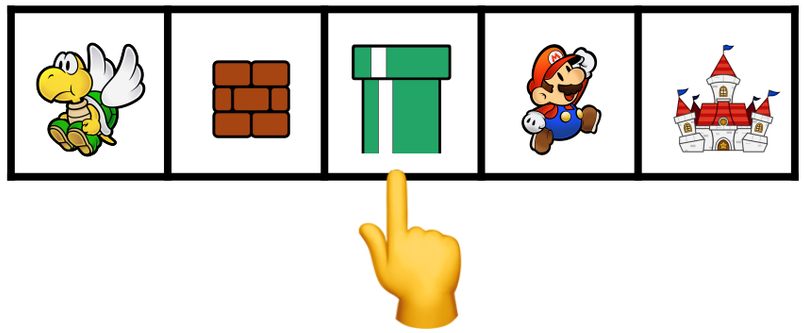
Affichage des objets dans le mauvais ordre

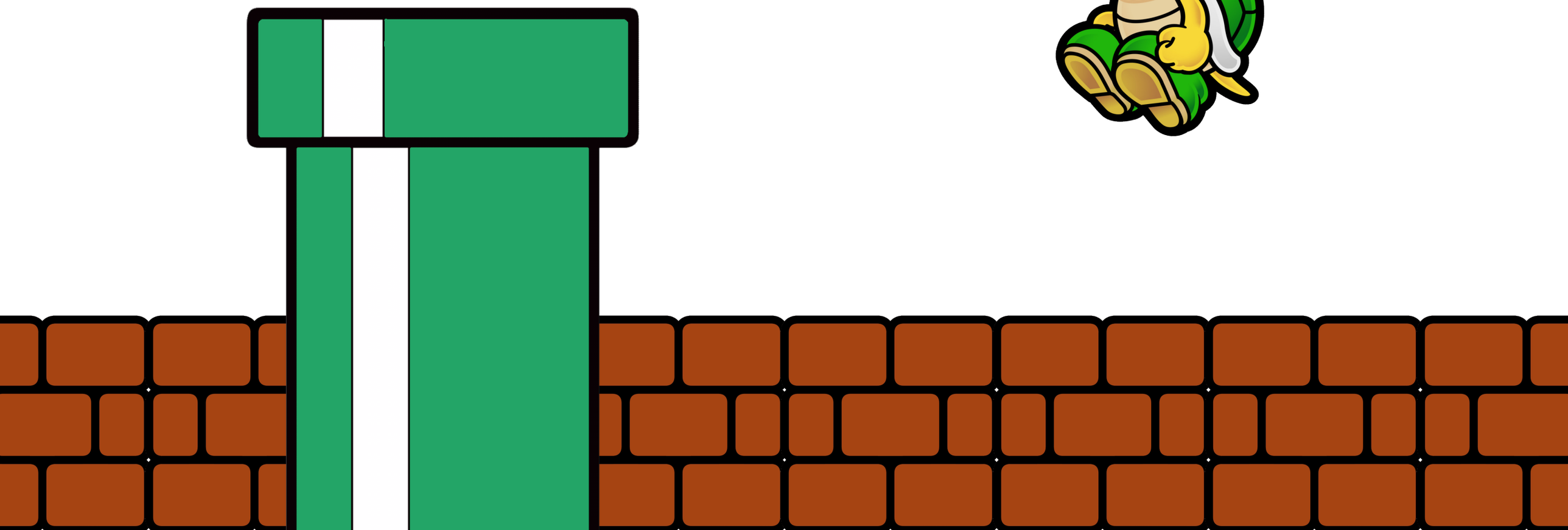


Affichage des objets dans le mauvais ordre

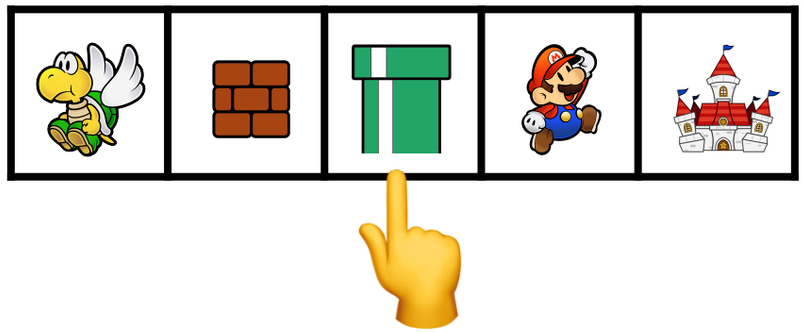


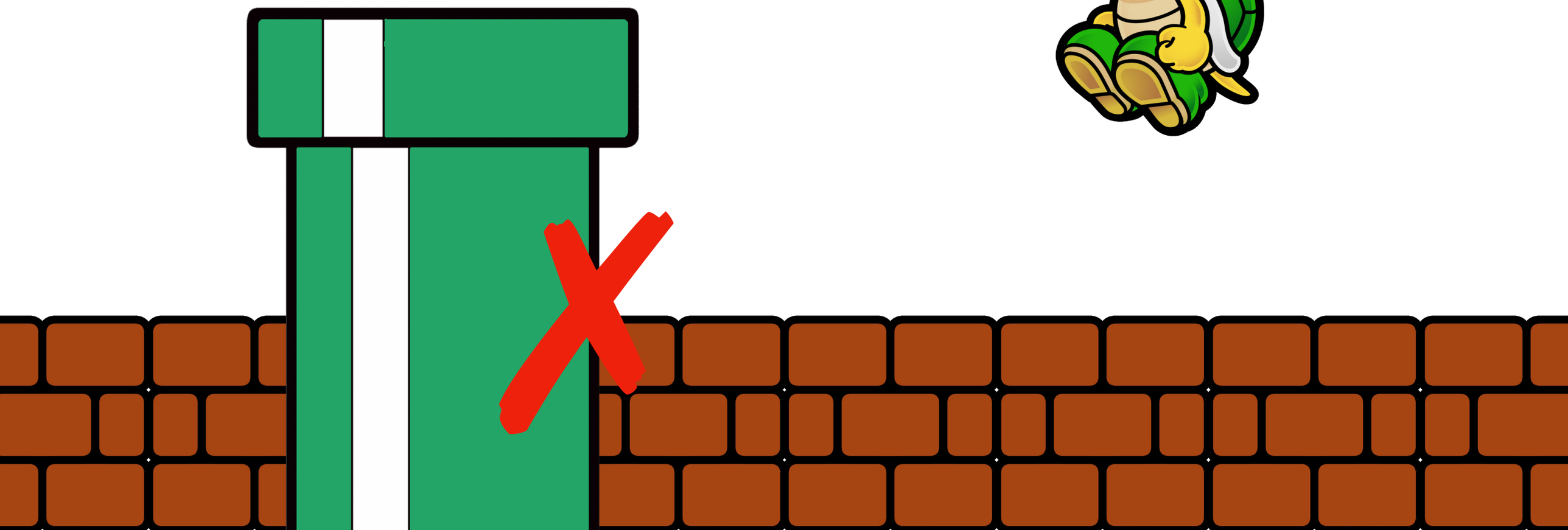
Affichage des objets dans le mauvais ordre



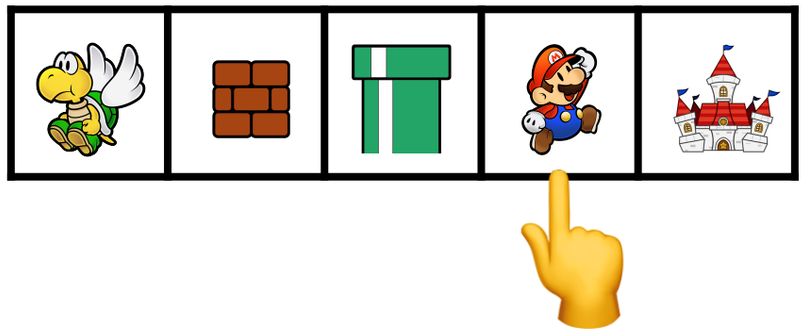


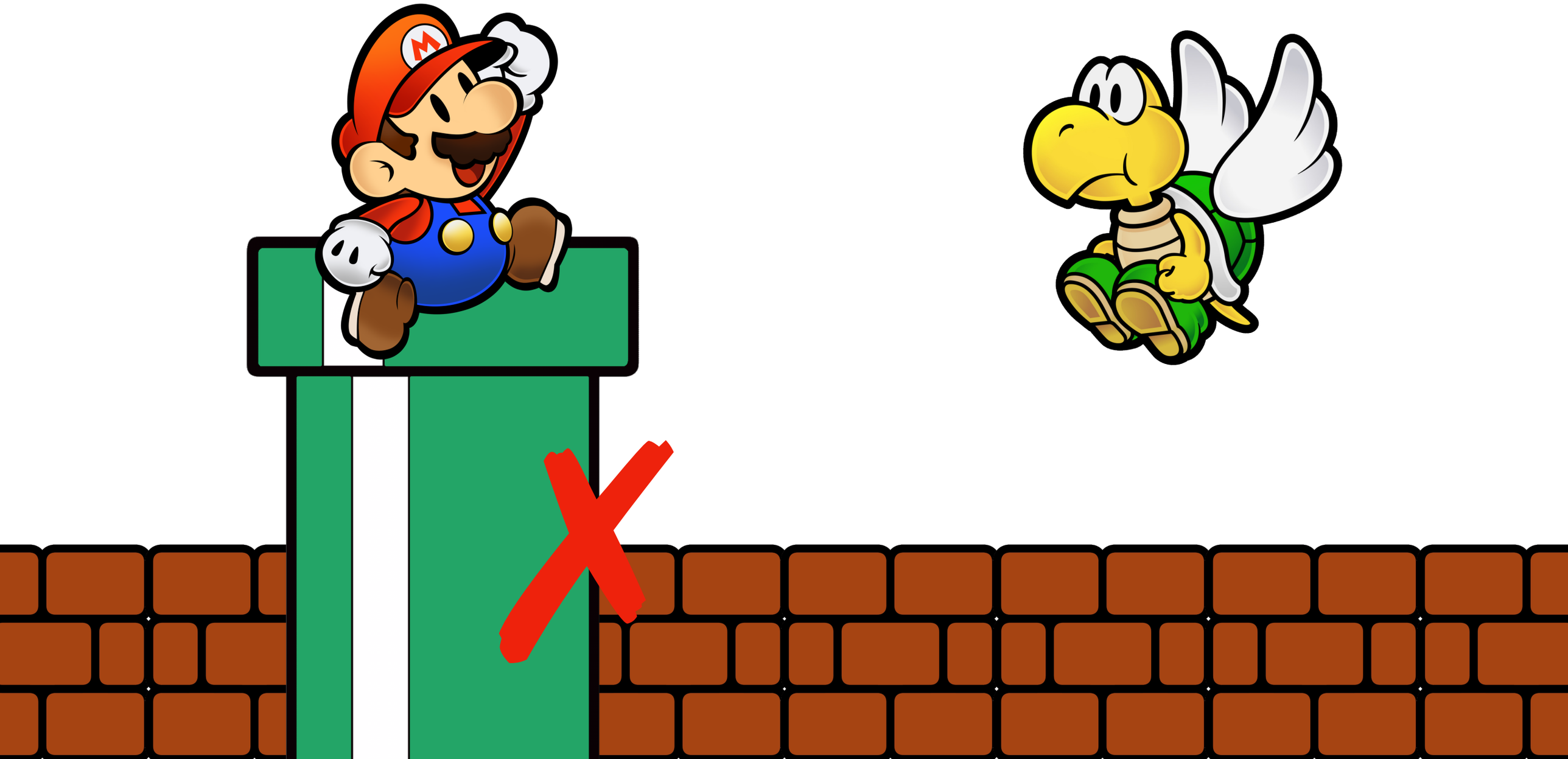
Affichage des objets dans le mauvais ordre



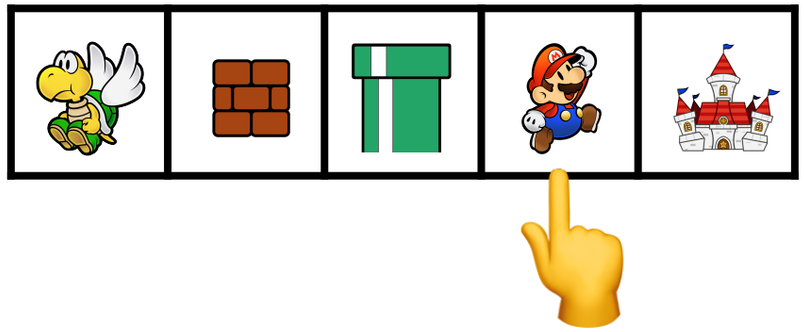


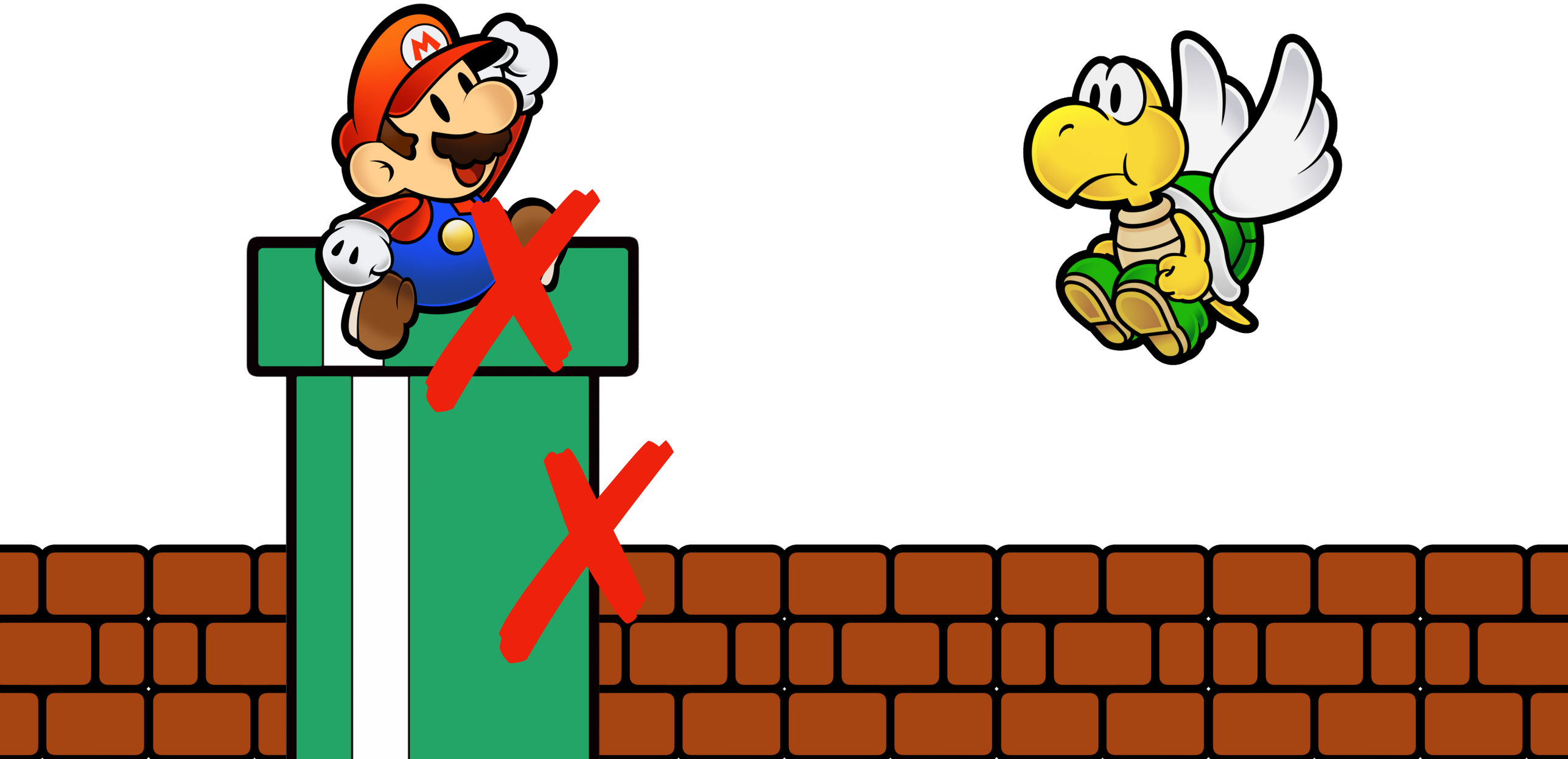
Affichage des objets dans le mauvais ordre



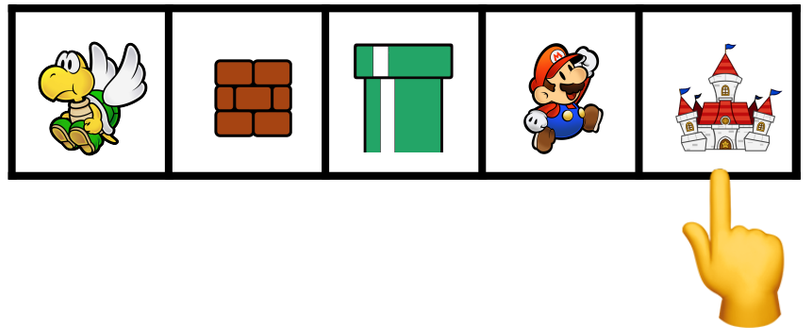


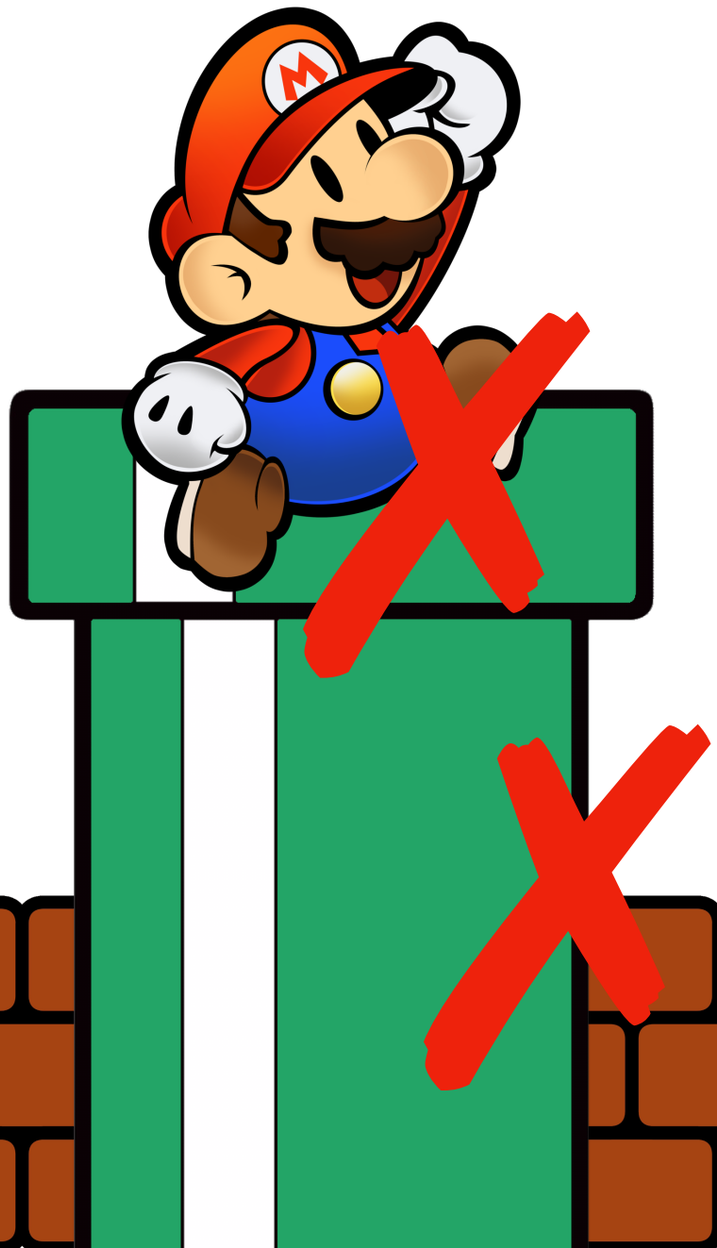
Affichage des objets dans le mauvais ordre



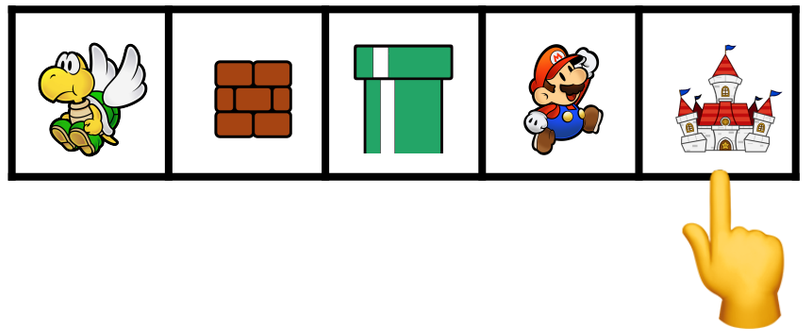


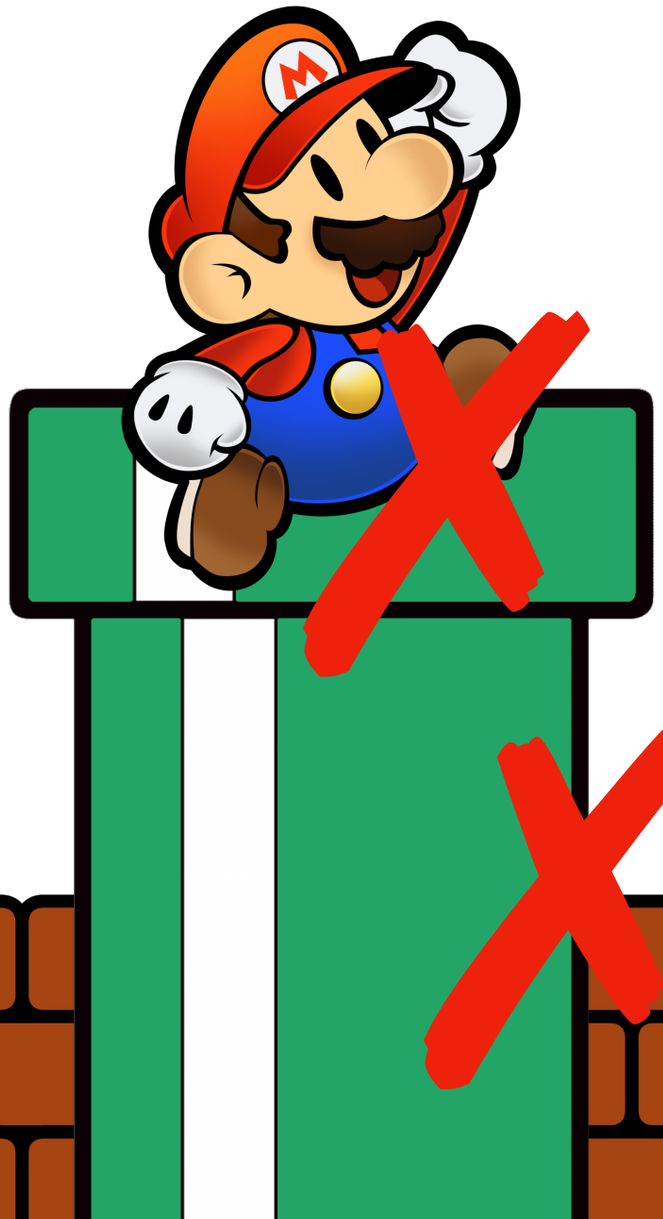
Affichage des objets dans le mauvais ordre



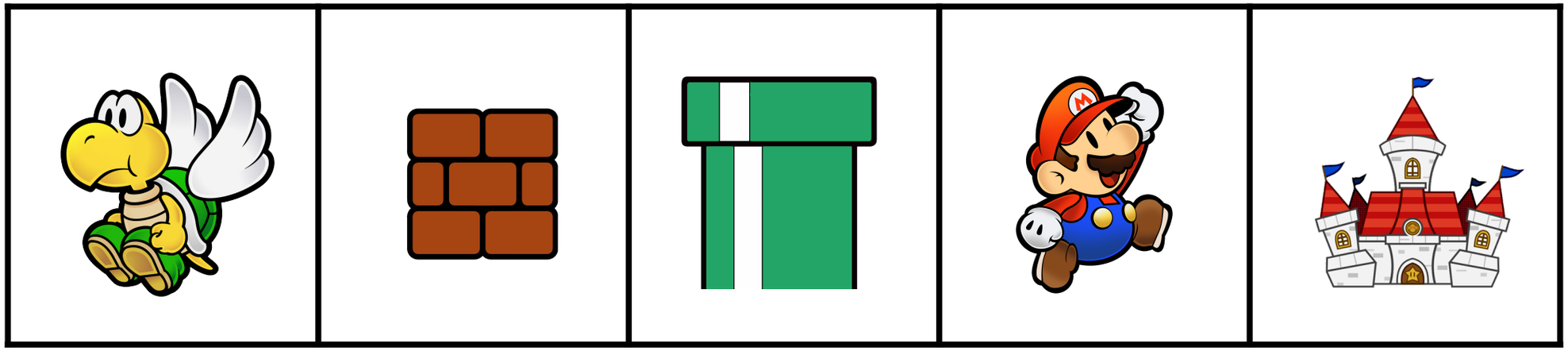


Affichage des objets dans le mauvais ordre



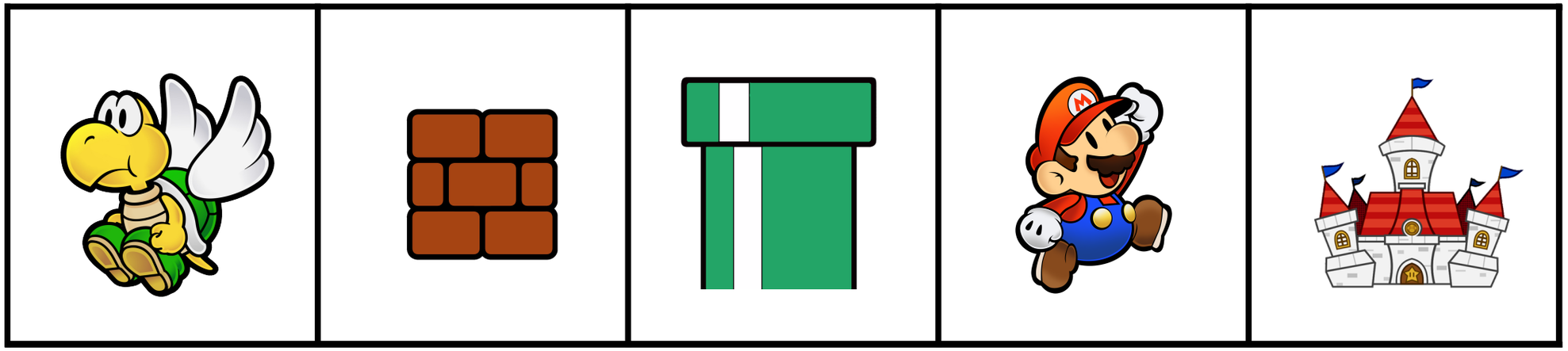


Affichage des objets dans le mauvais ordre



0 1 2 3 4

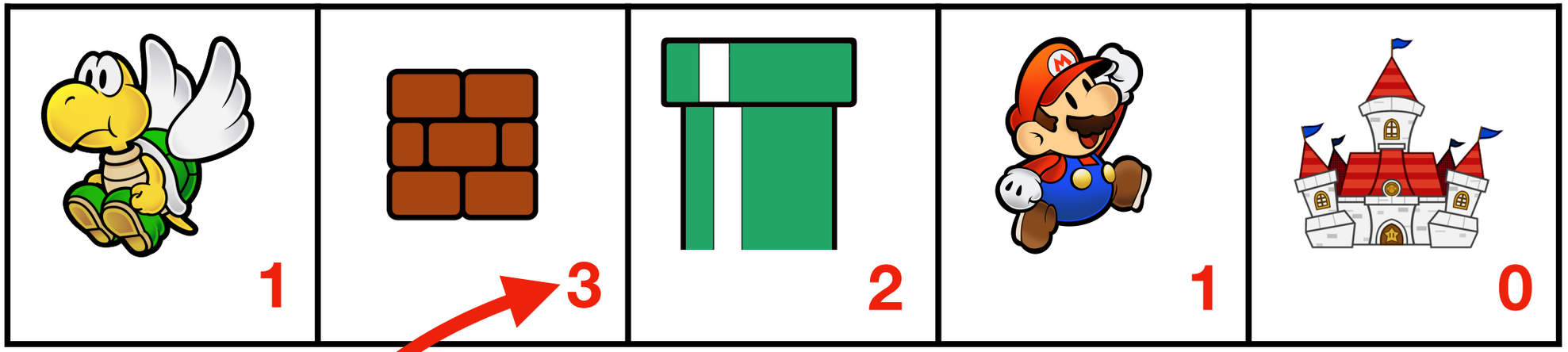
Affichage des objets dans le mauvais ordre



0 1 2 3 4

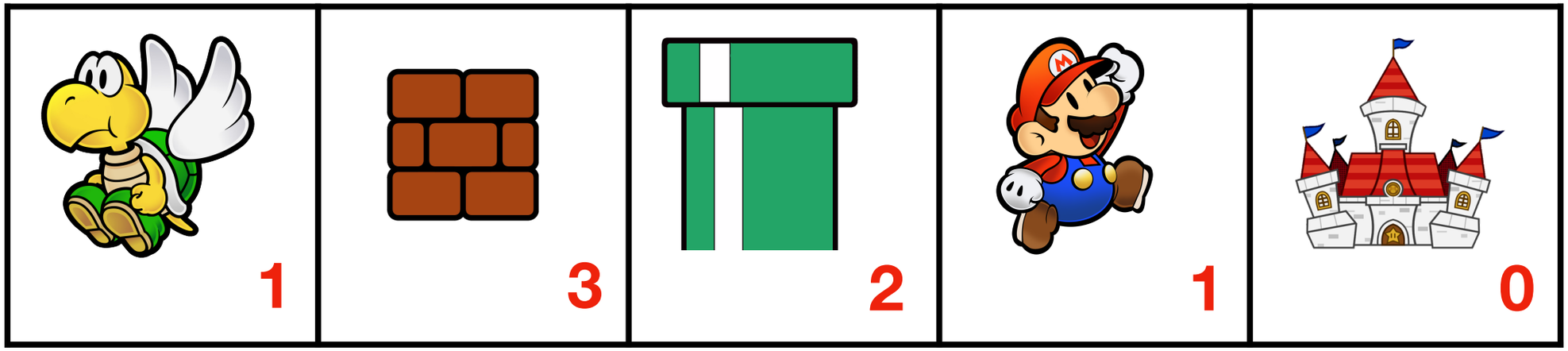
ordre d’affichage

Affichage des objets dans le mauvais ordre



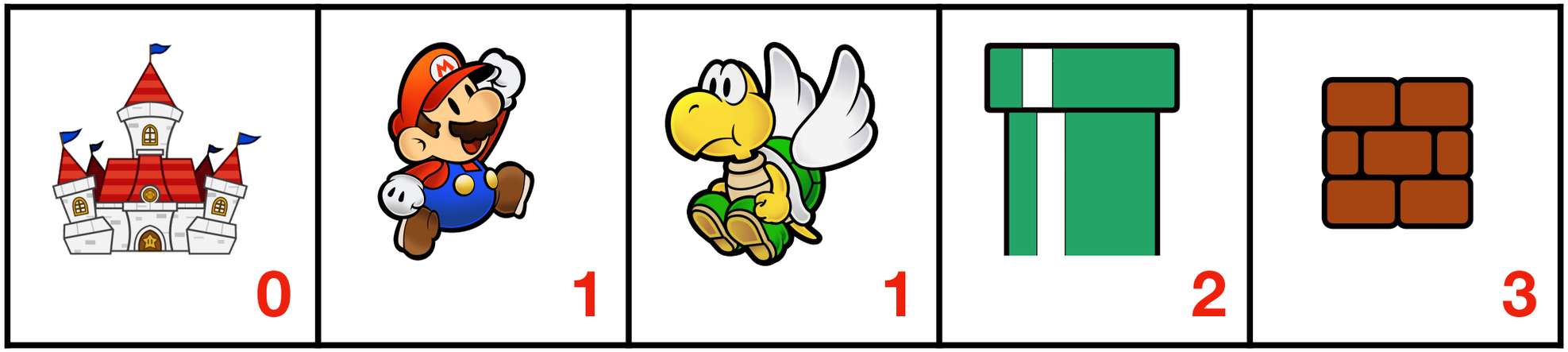
Tri ! 

Affichage des objets dans le mauvais ordre



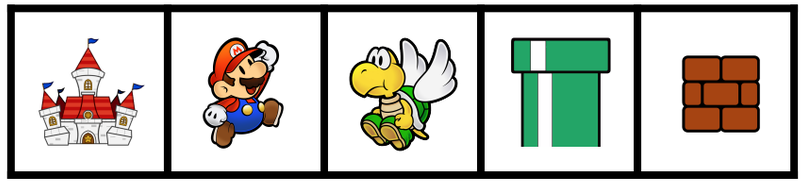
0 1 2 3 4

Affichage des objets dans le bon ordre

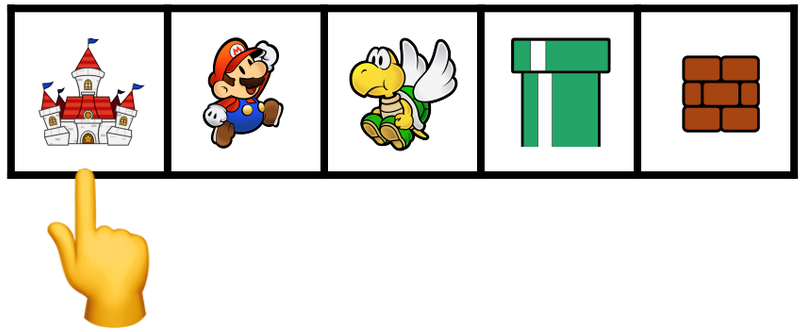


0 1 2 3 4

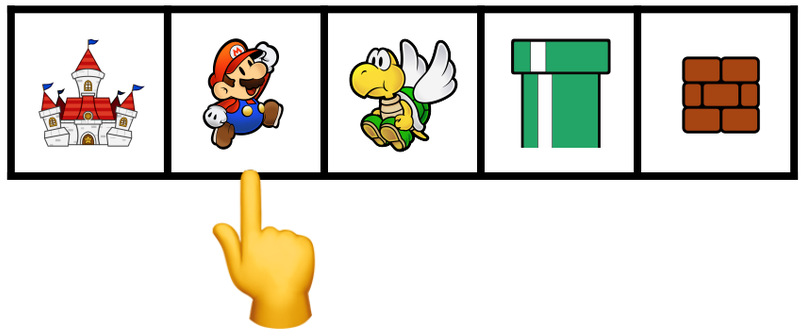
Affichage des objets dans le bon ordre



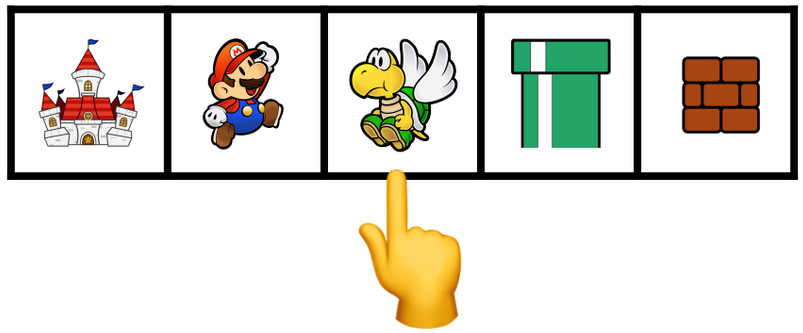
Affichage des objets dans le bon ordre

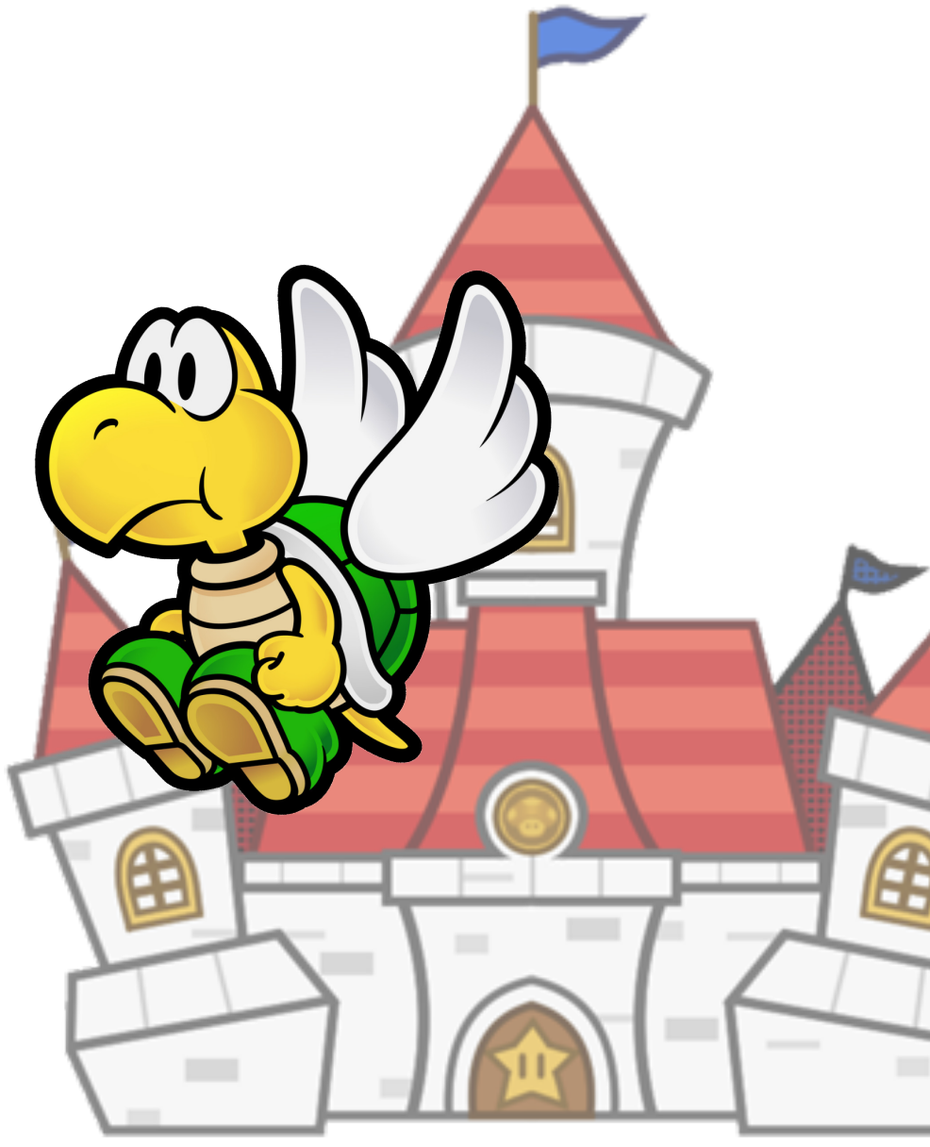


Affichage des objets dans le bon ordre

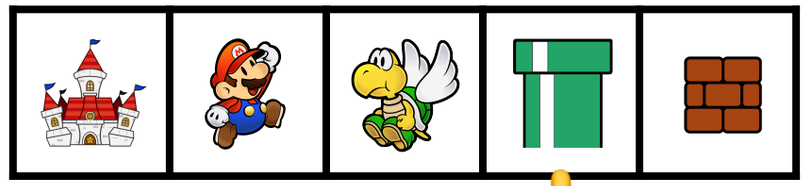


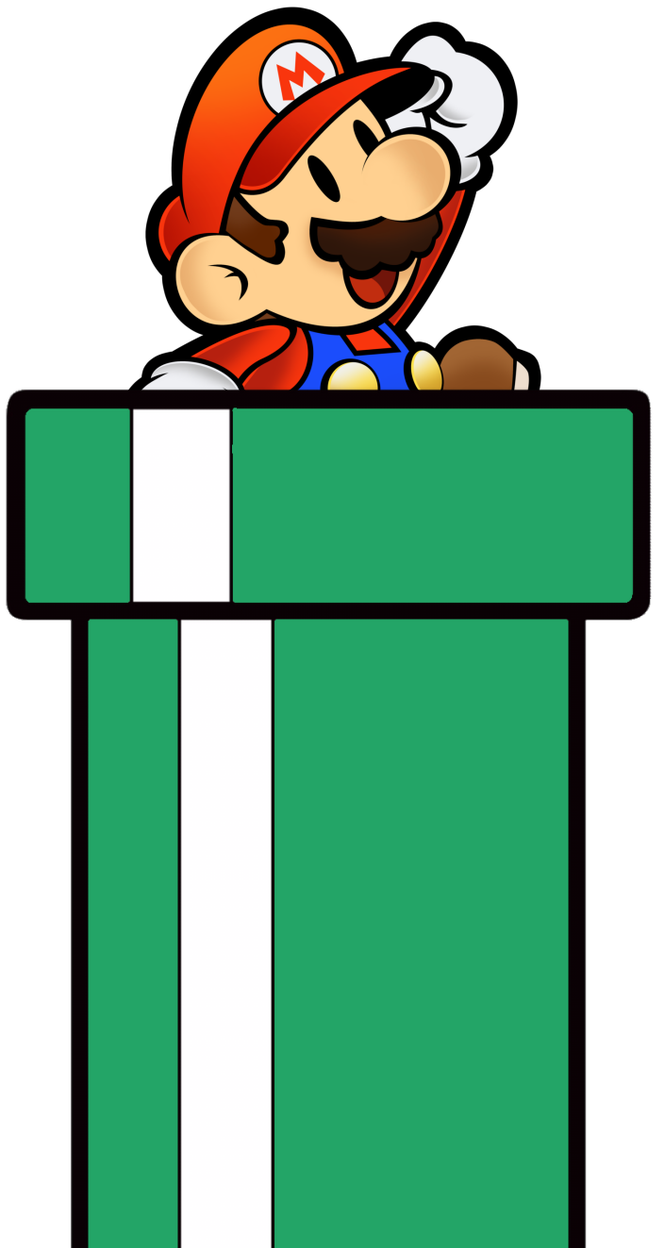
Affichage des objets dans le bon ordre



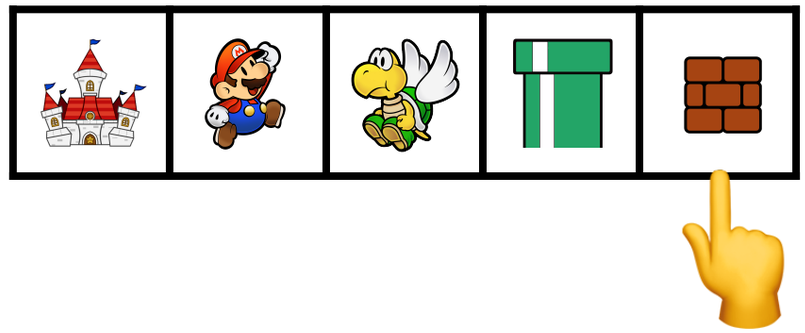


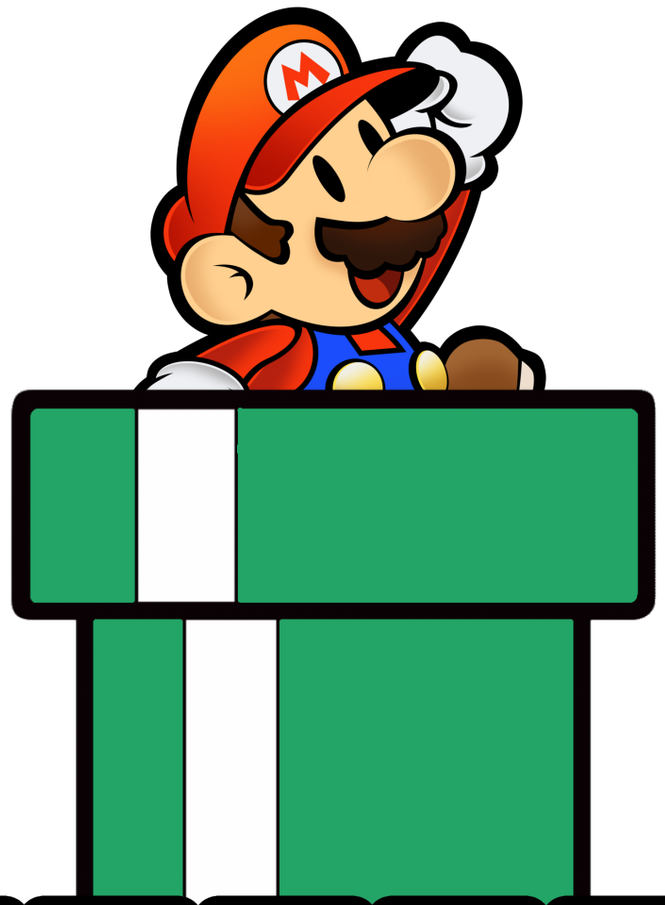
Affichage des objets dans le bon ordre

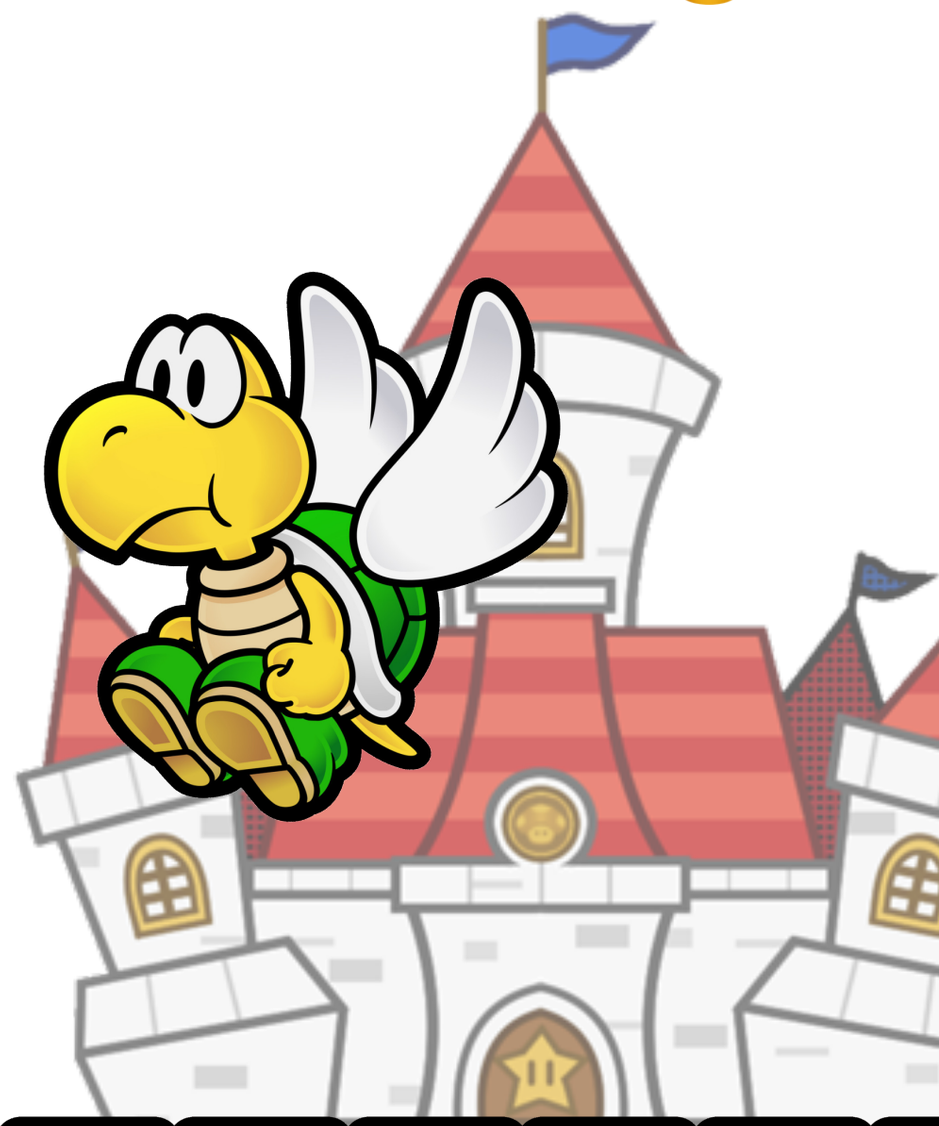




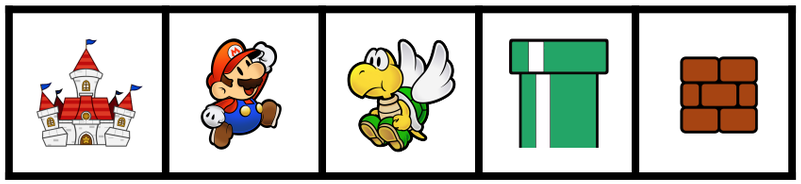
Affichage des objets dans le bon ordre

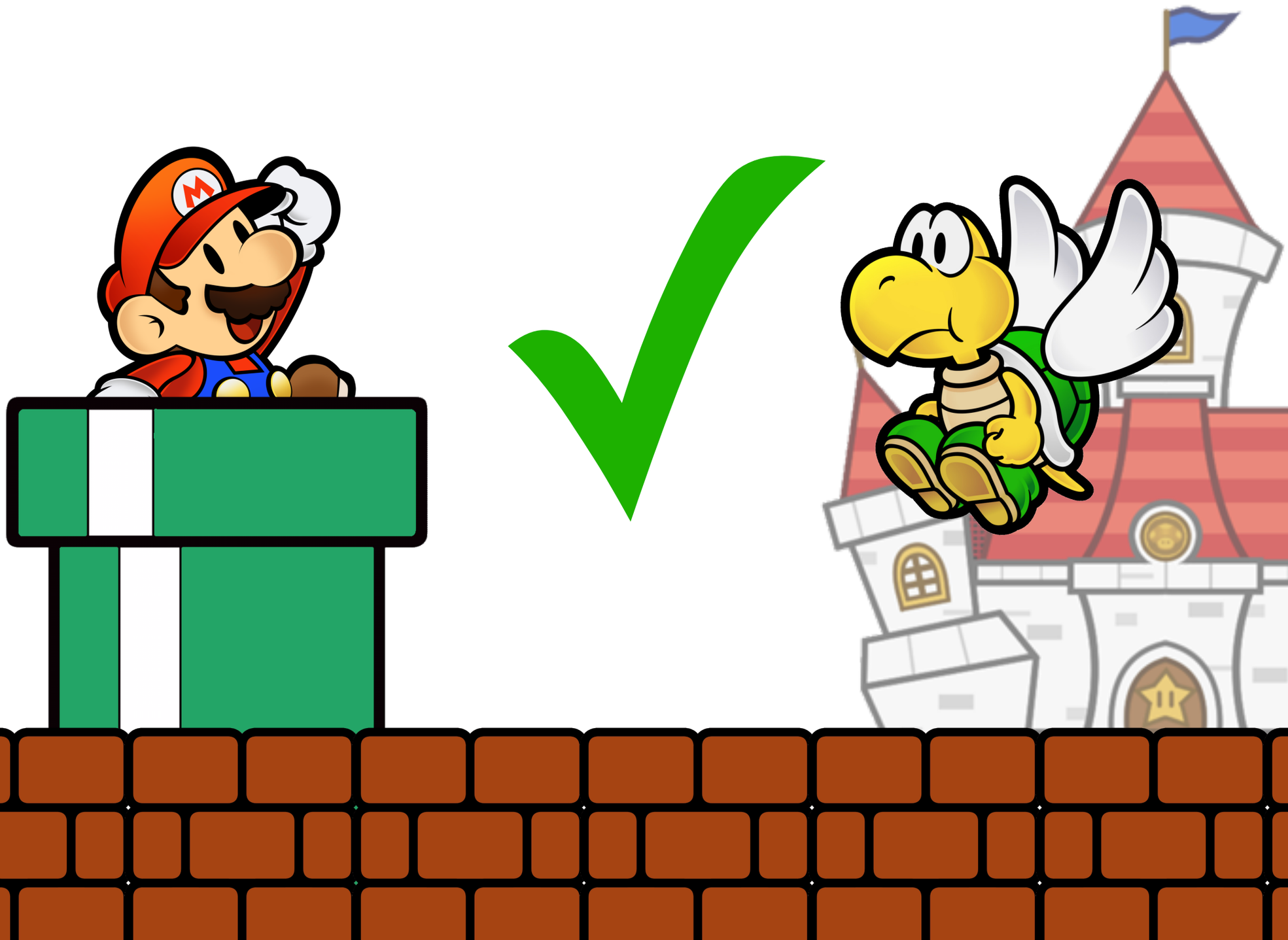




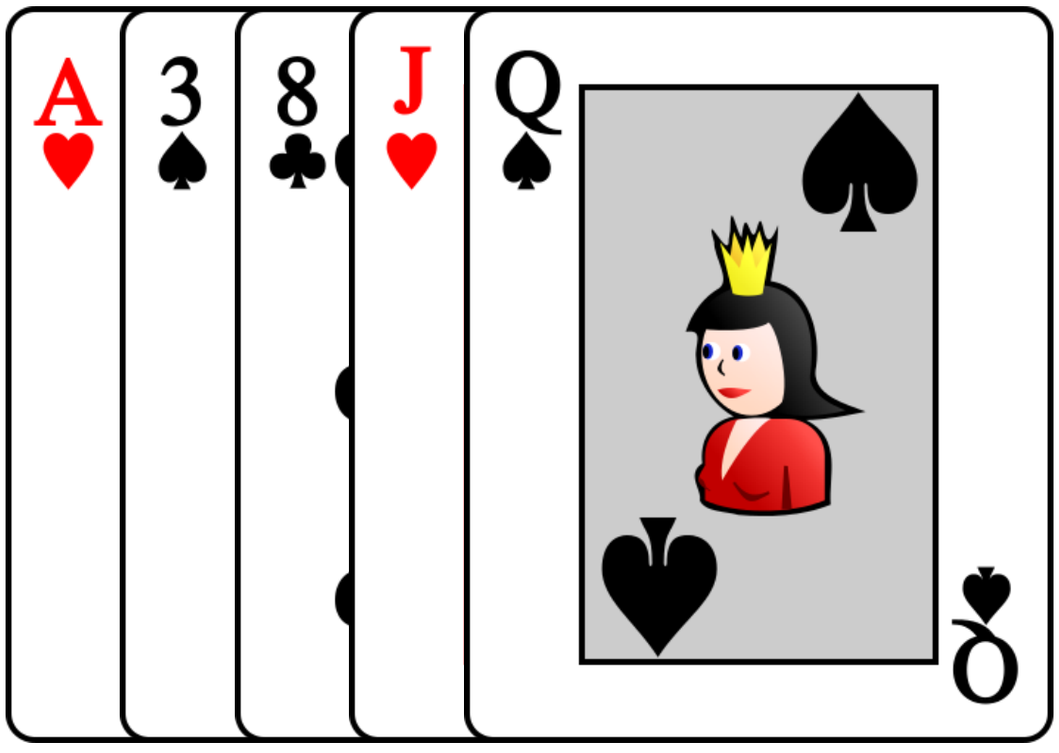


Affichage des objets dans le bon ordre



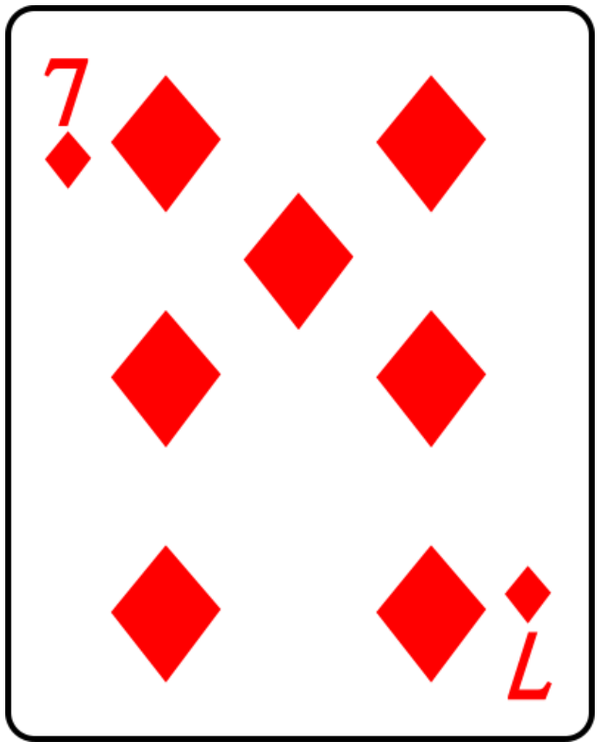


Comment trier un jeu de cartes ?

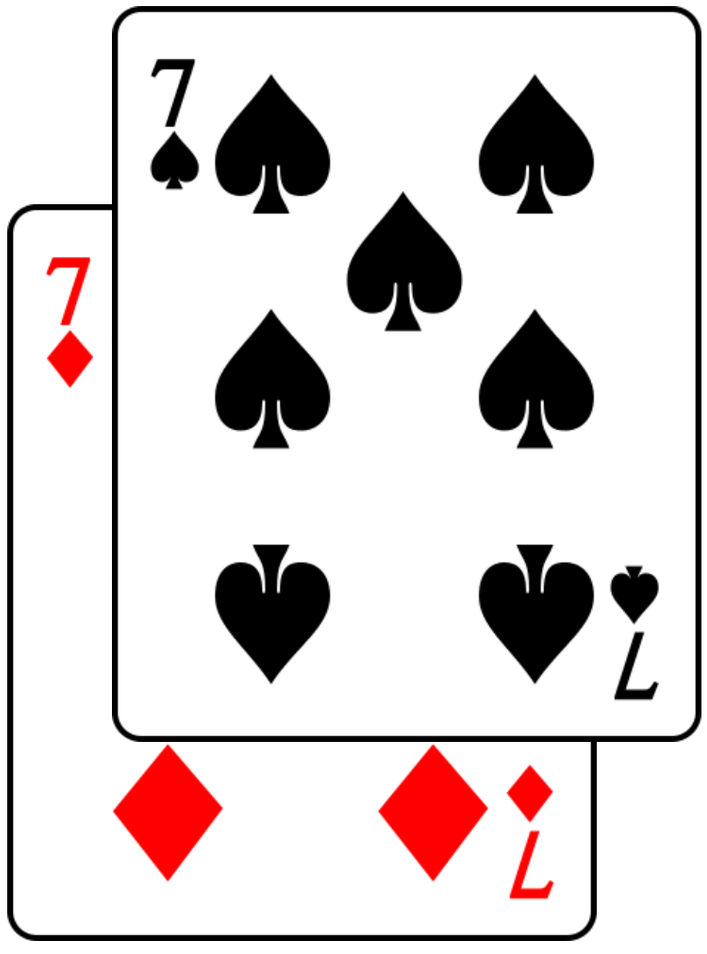


Tri par insertion

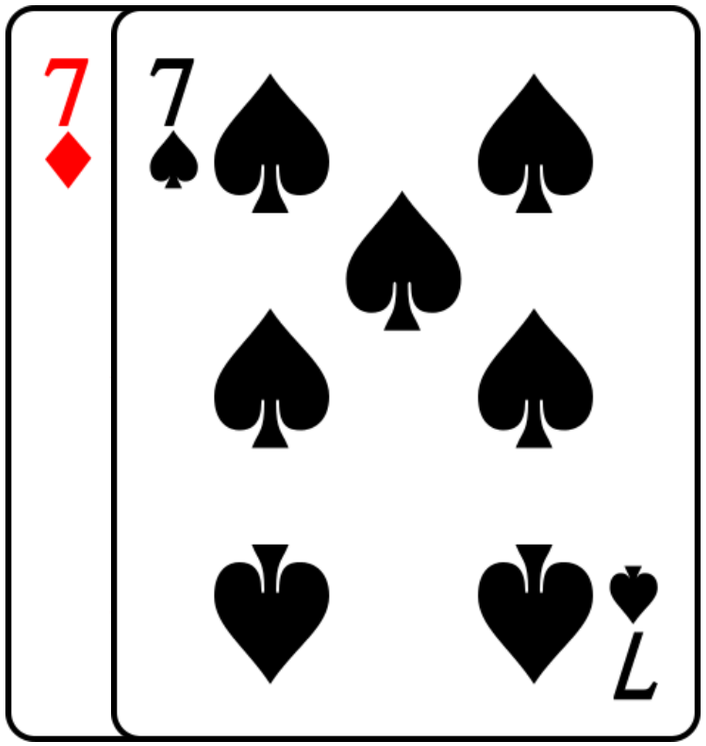
Tri par insertion



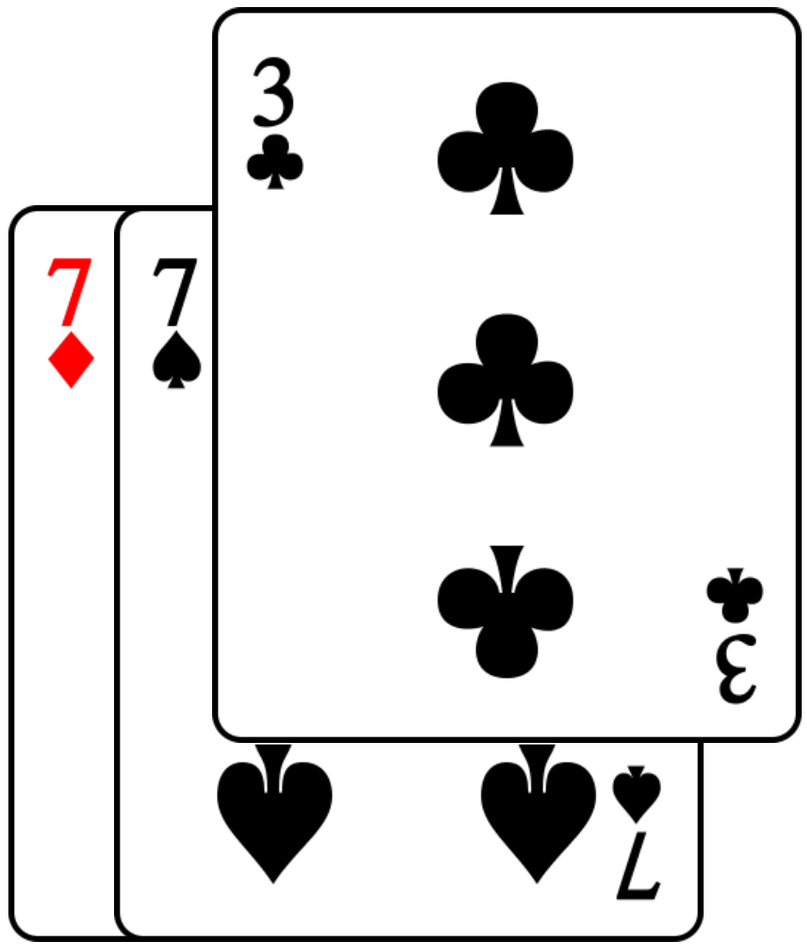
Tri par insertion



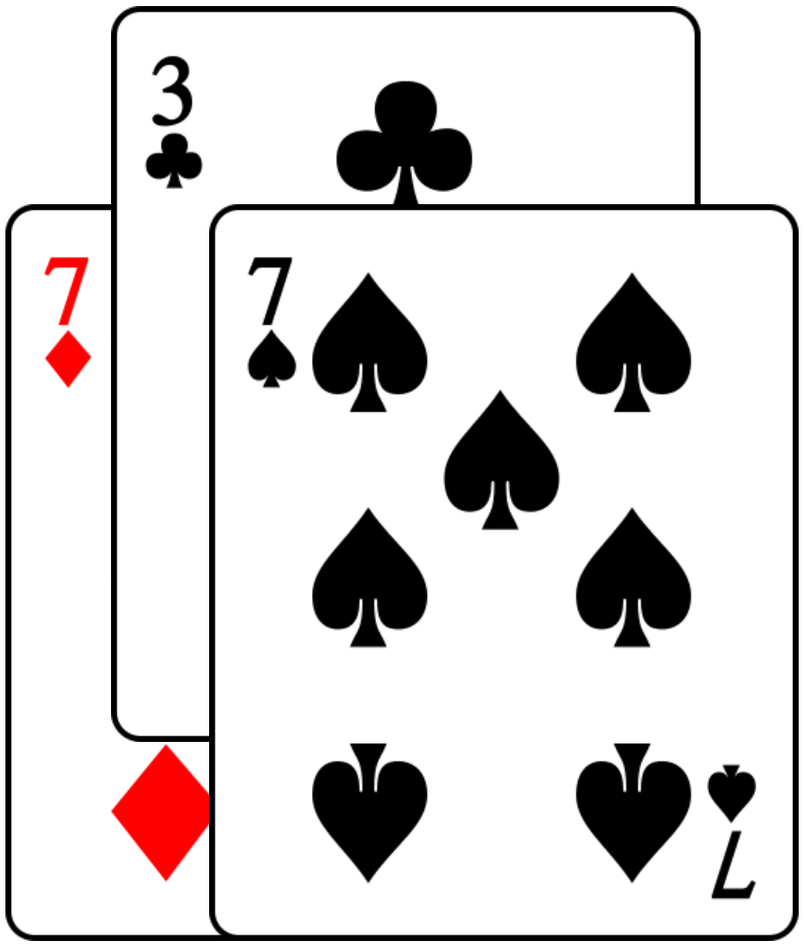
Tri par insertion



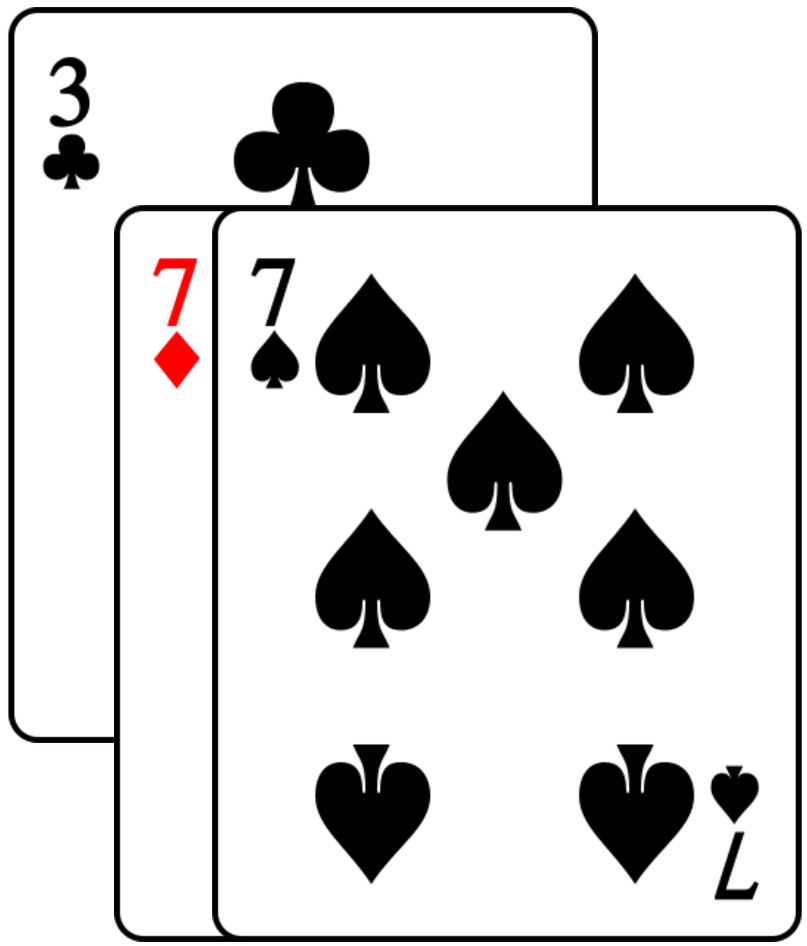
Tri par insertion



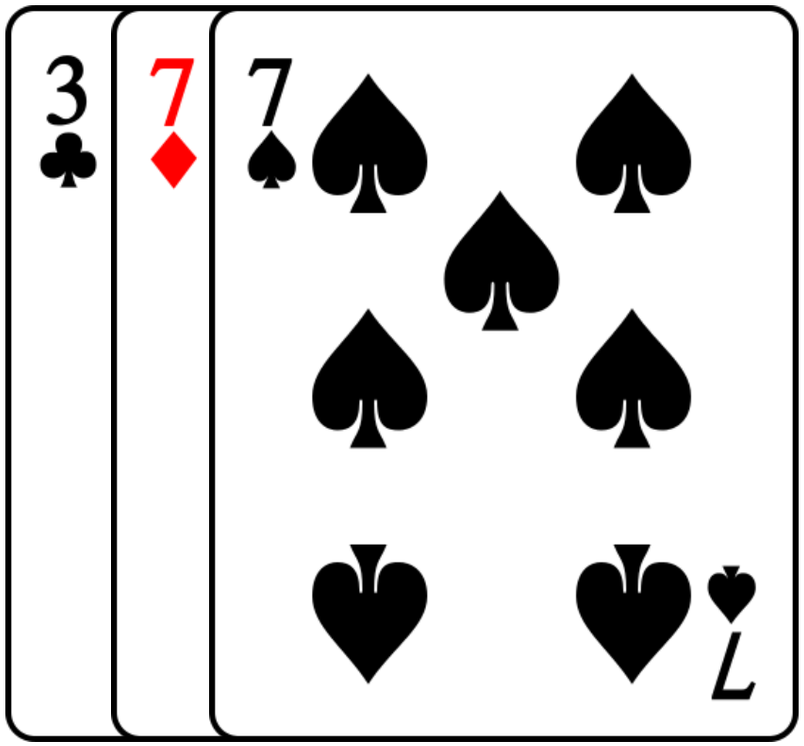
Tri par insertion



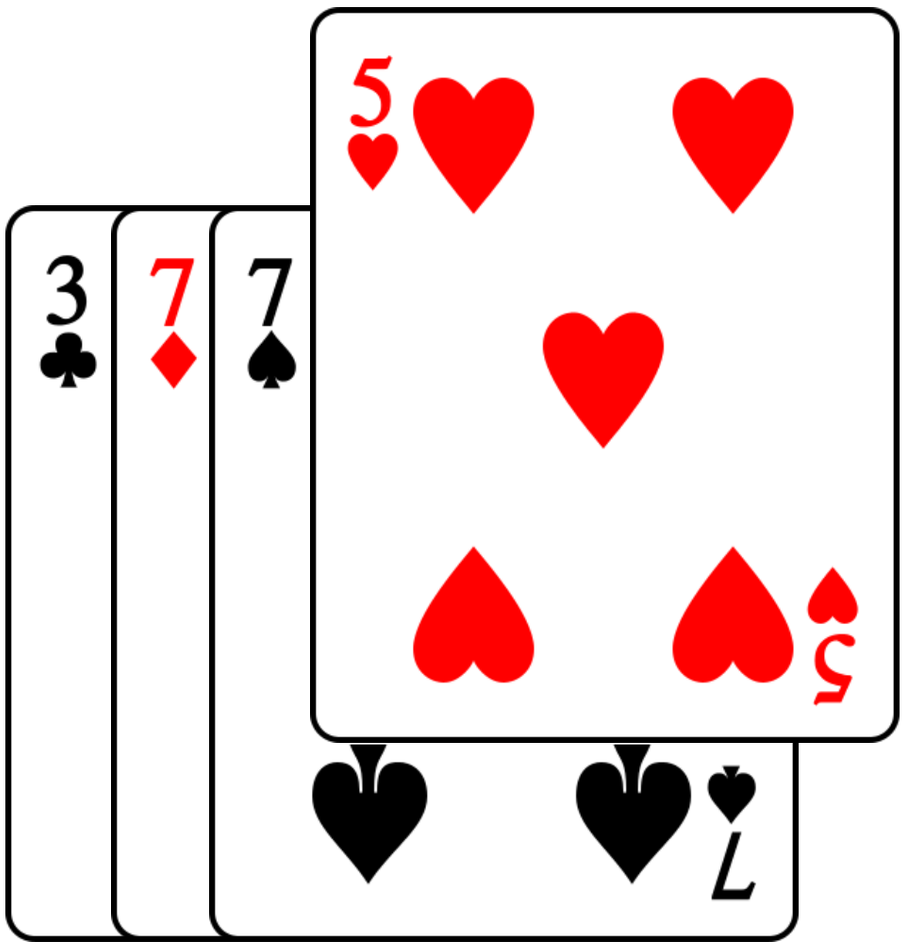
Tri par insertion



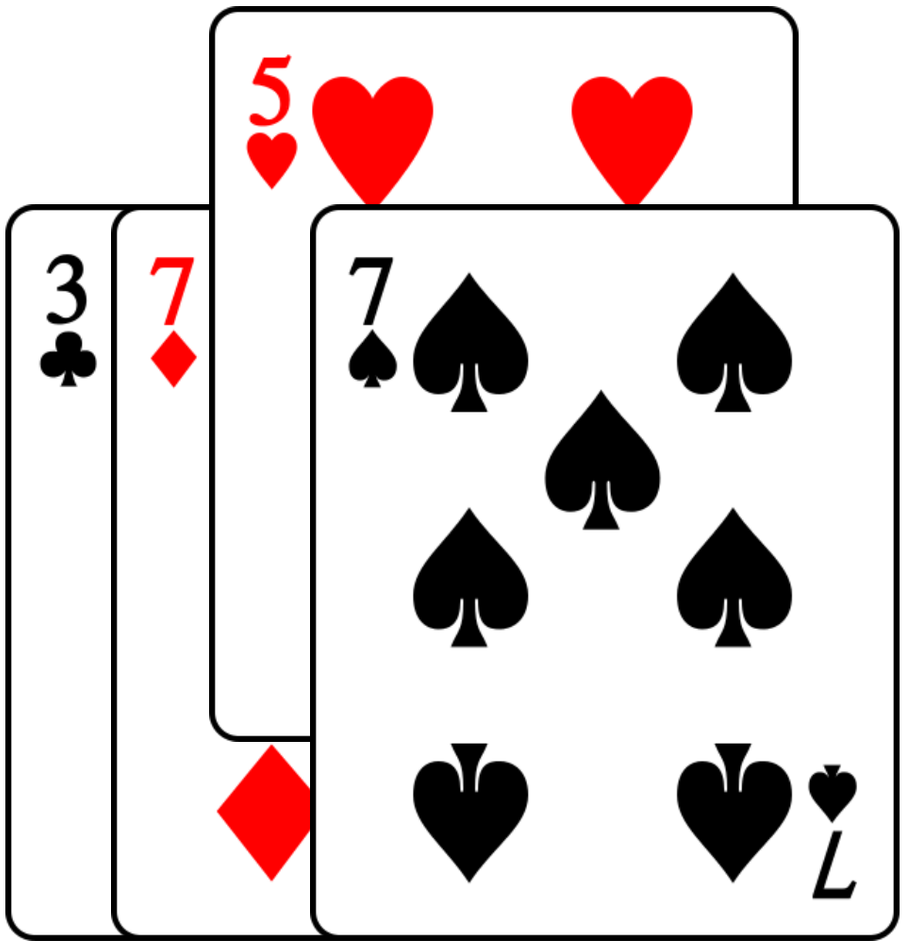
Tri par insertion



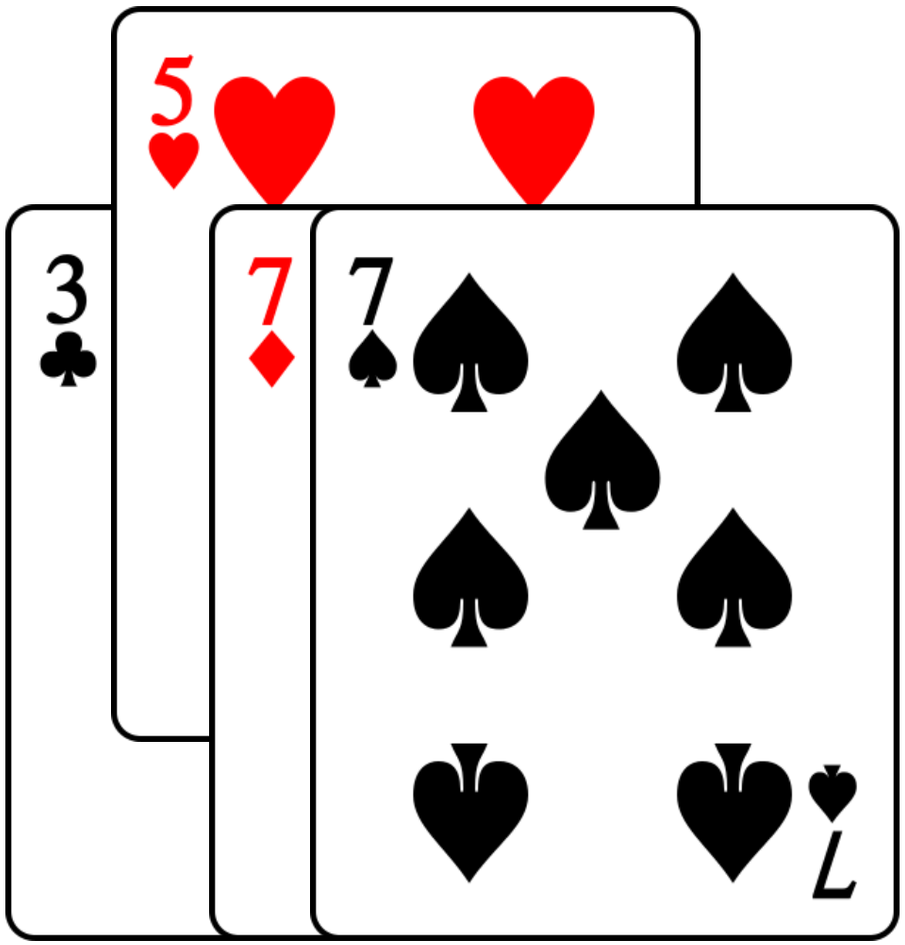
Tri par insertion



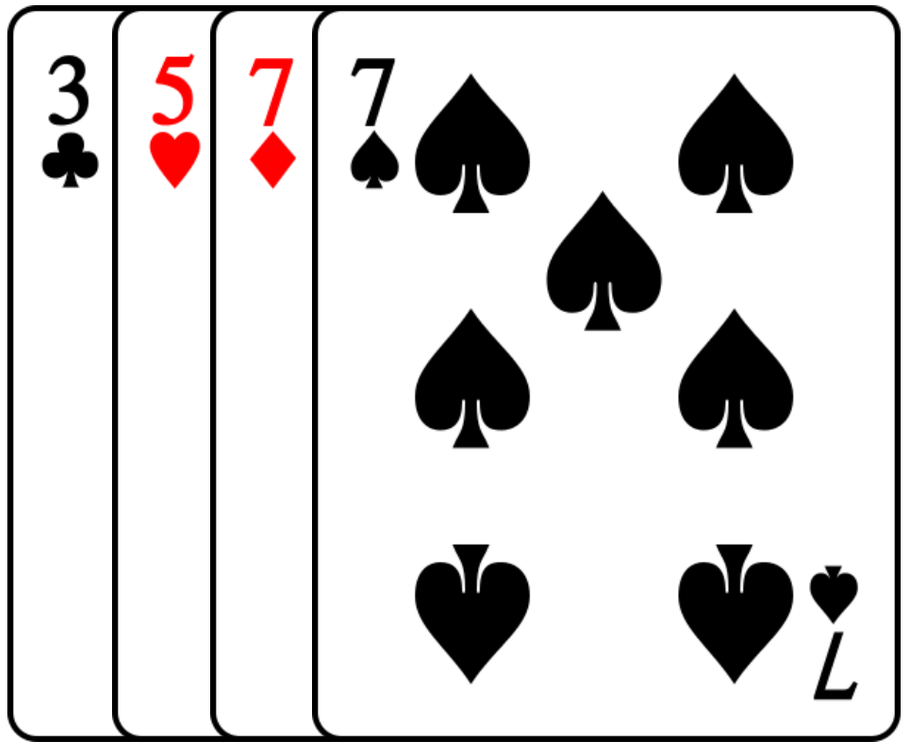
Tri par insertion



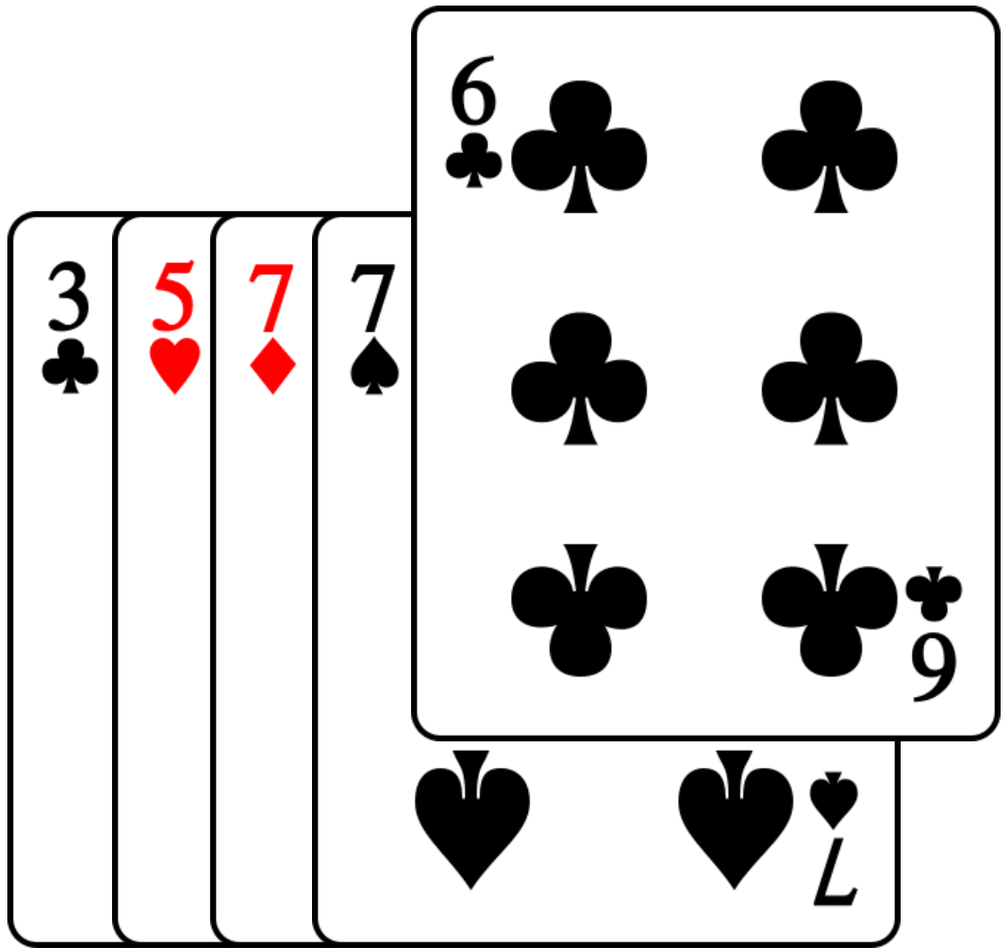
Tri par insertion



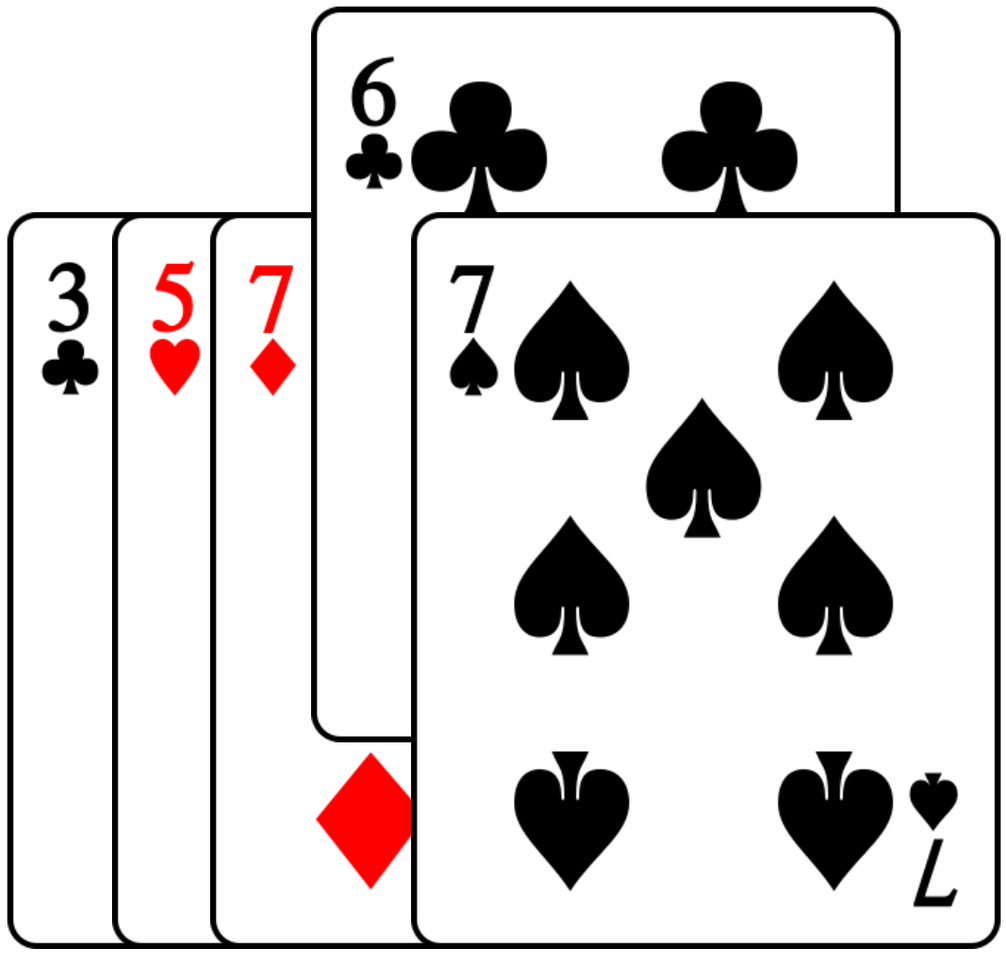
Tri par insertion



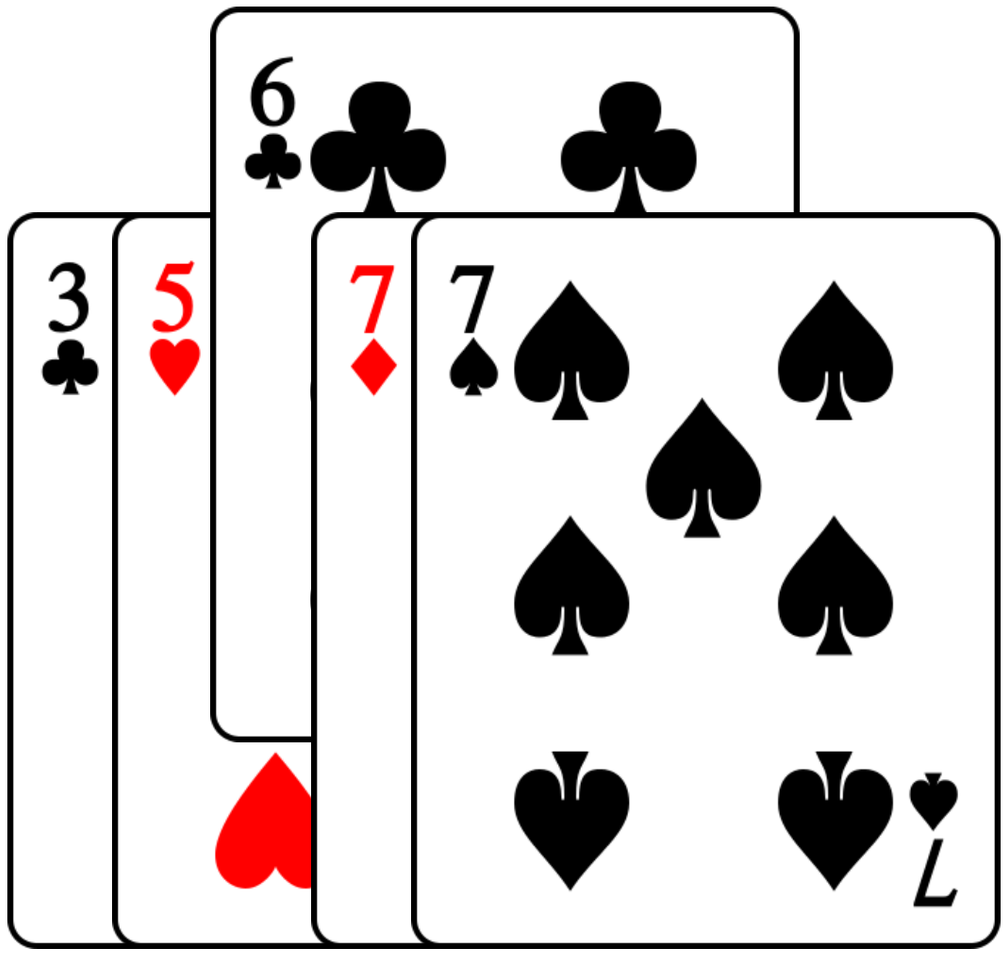
Tri par insertion



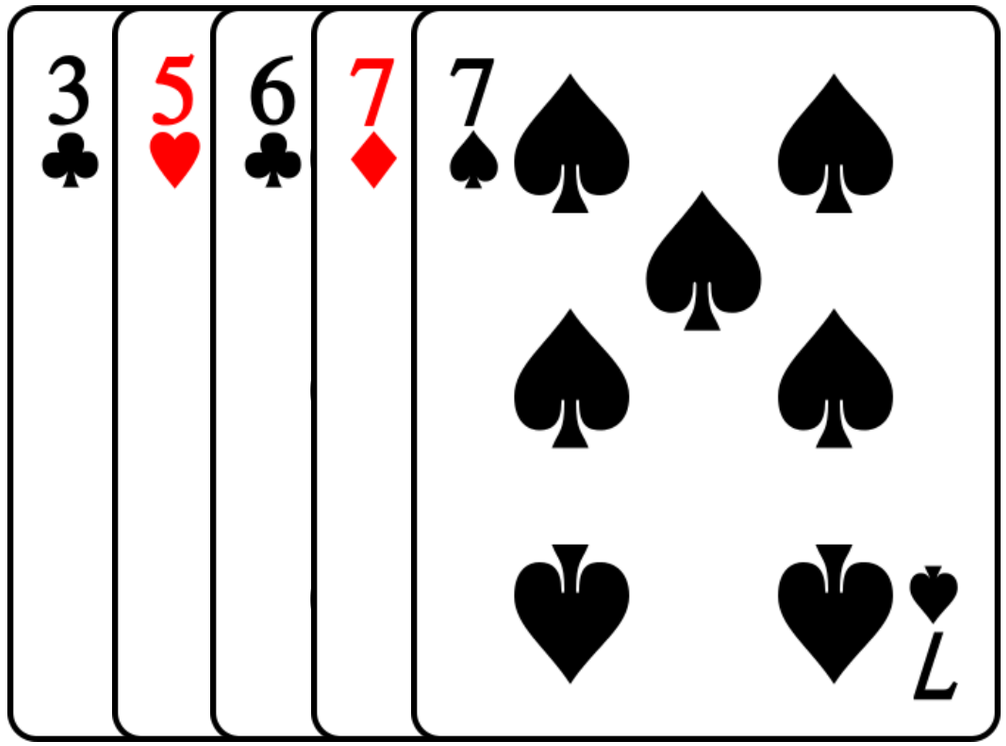
Tri par insertion



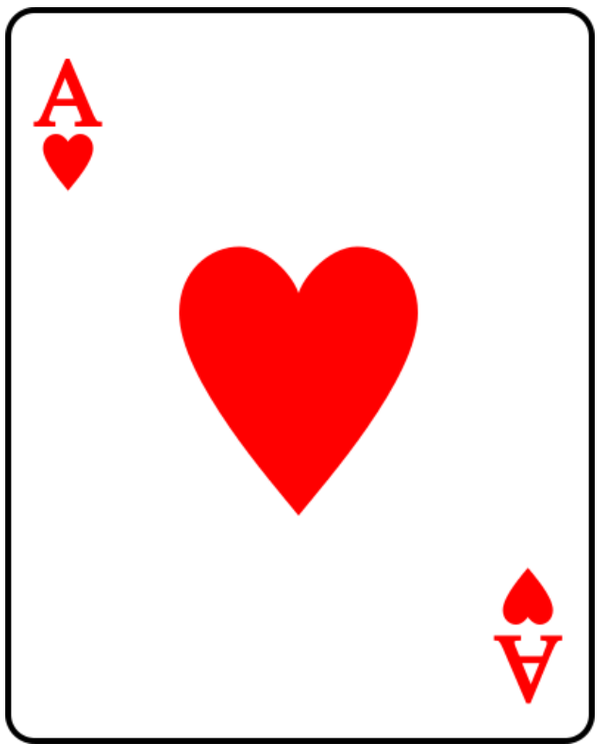
Tri par insertion



Tri par insertion

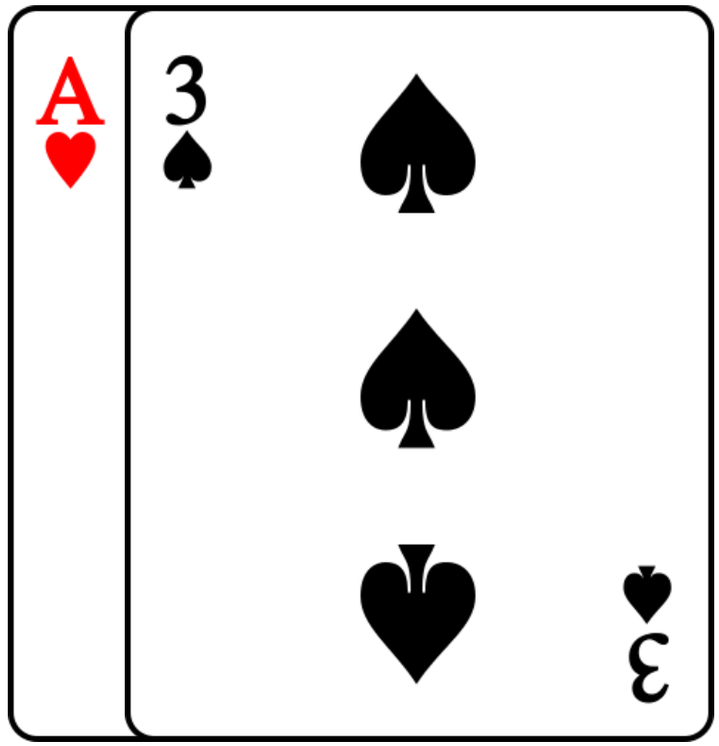


Le meilleur des cas



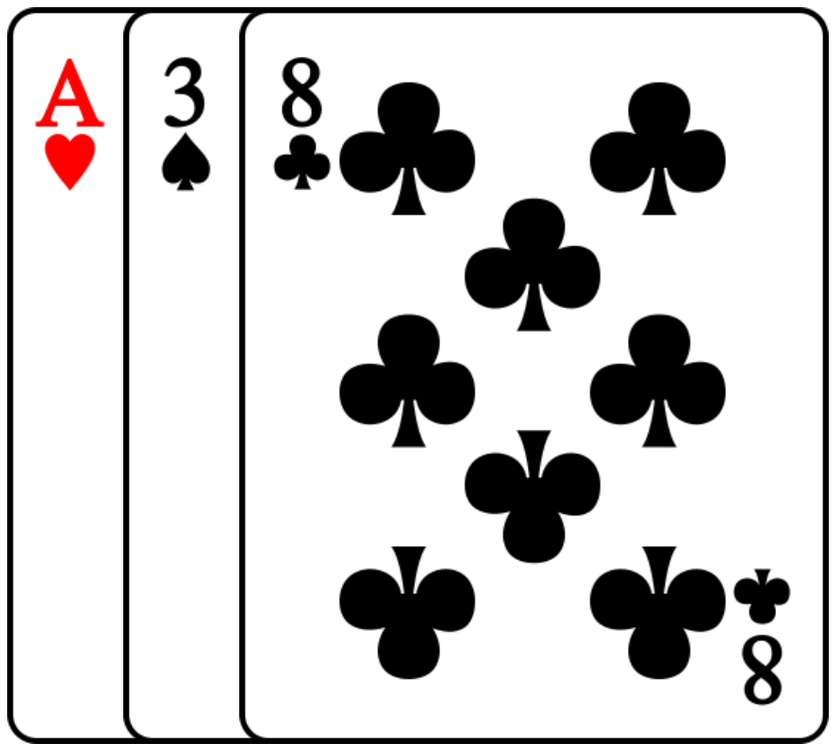
№ operations = 1

Le meilleur des cas



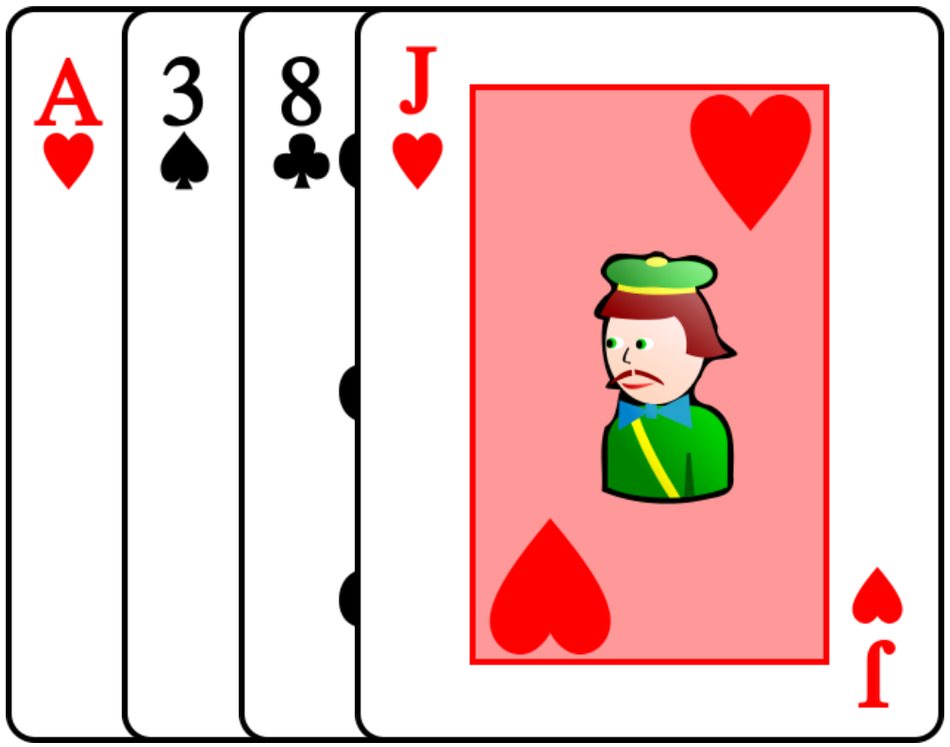
№ operations = 2

Le meilleur des cas



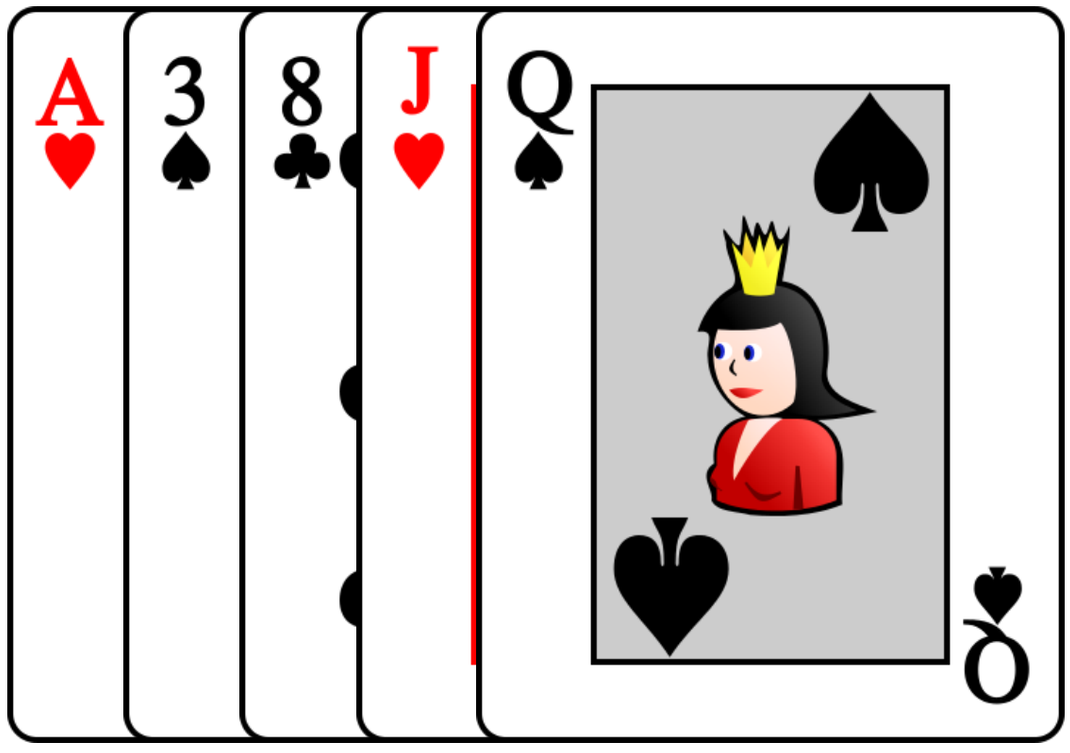
№ operations = 3

Le meilleur des cas



№ operations = 4

Le meilleur des cas



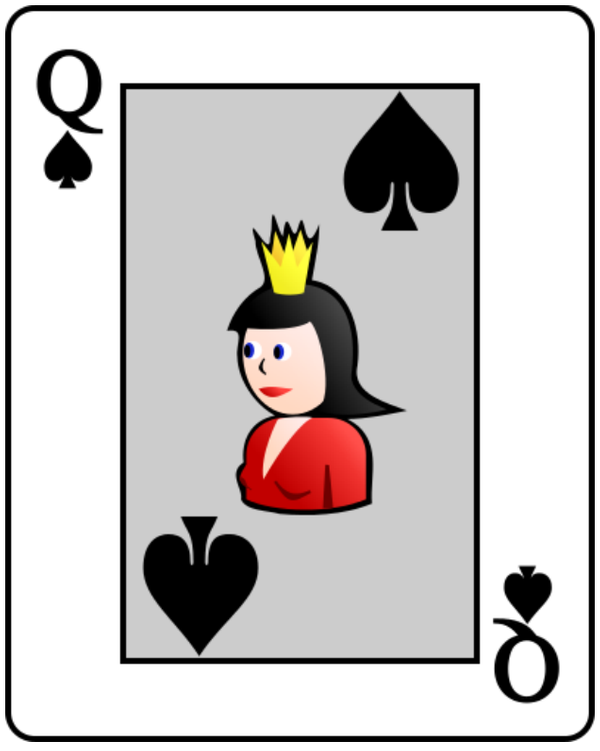
№ operations = 5

Le meilleur des cas

- Les cartes arrivent déjà triées

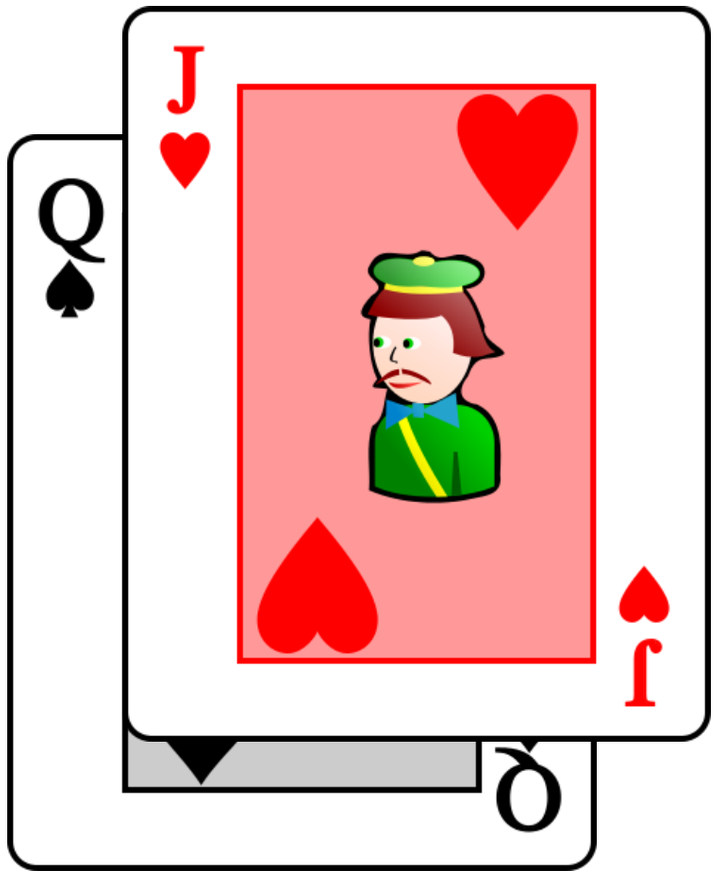
- On fait n opérations (déplacements de cartes)

Le pire des cas



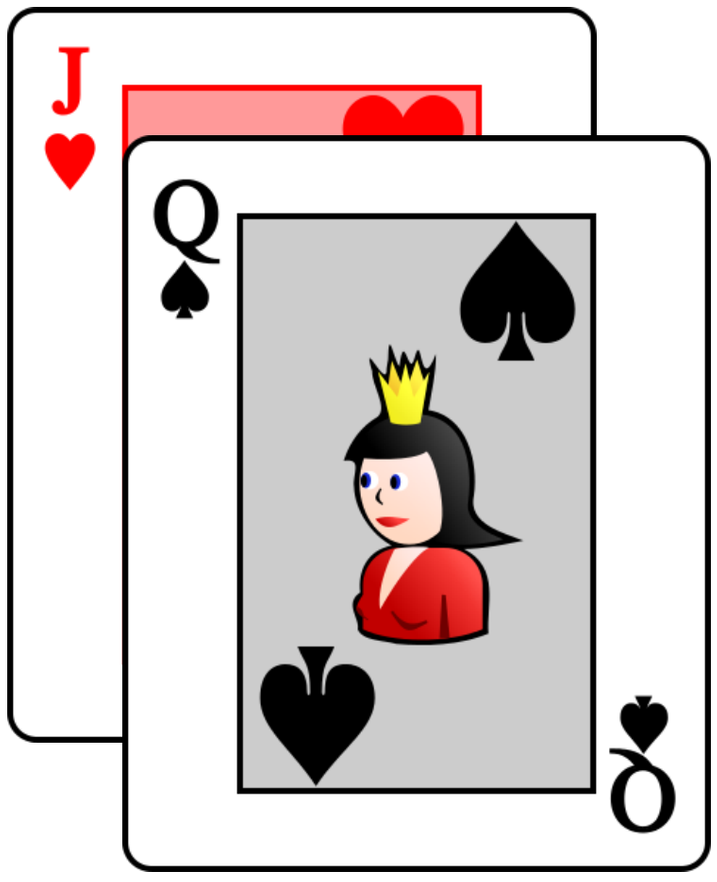
№ operations = 1

Le pire des cas



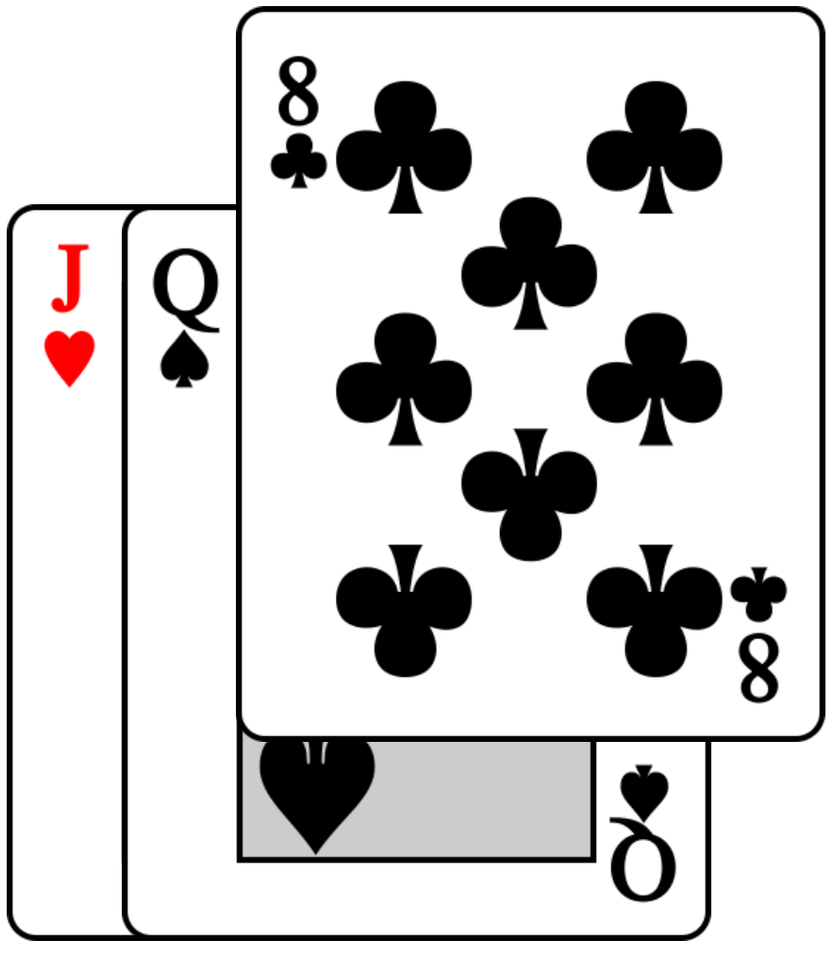
№ operations = 1 + 1

Le pire des cas



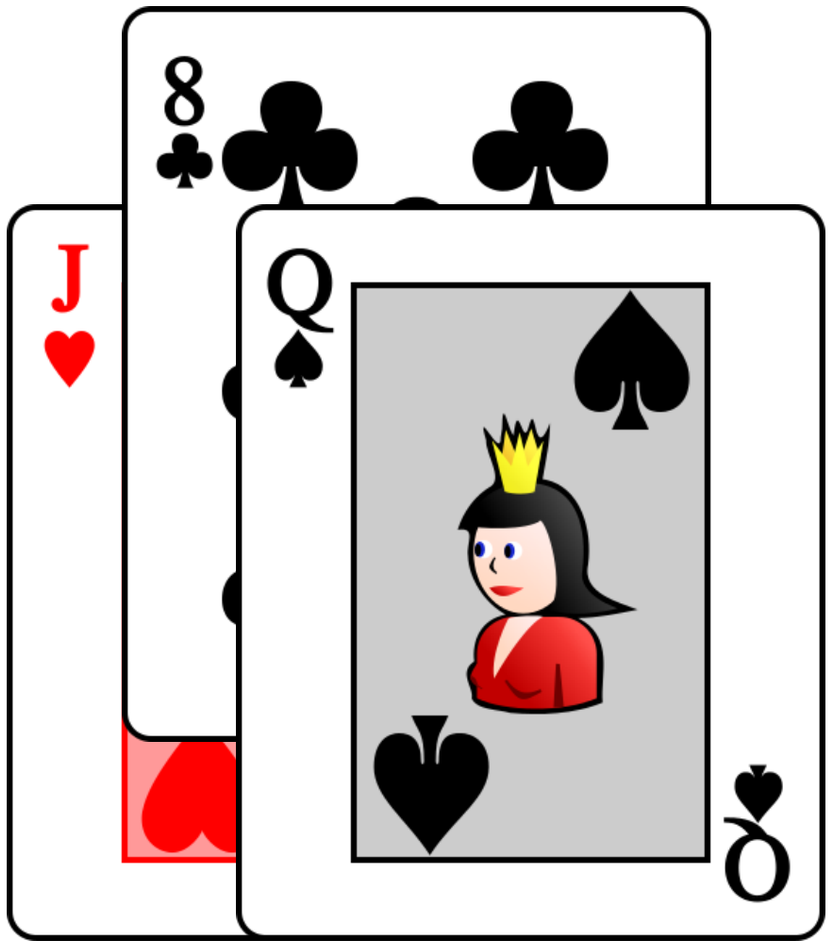
№ operations = 1 + 2

Le pire des cas



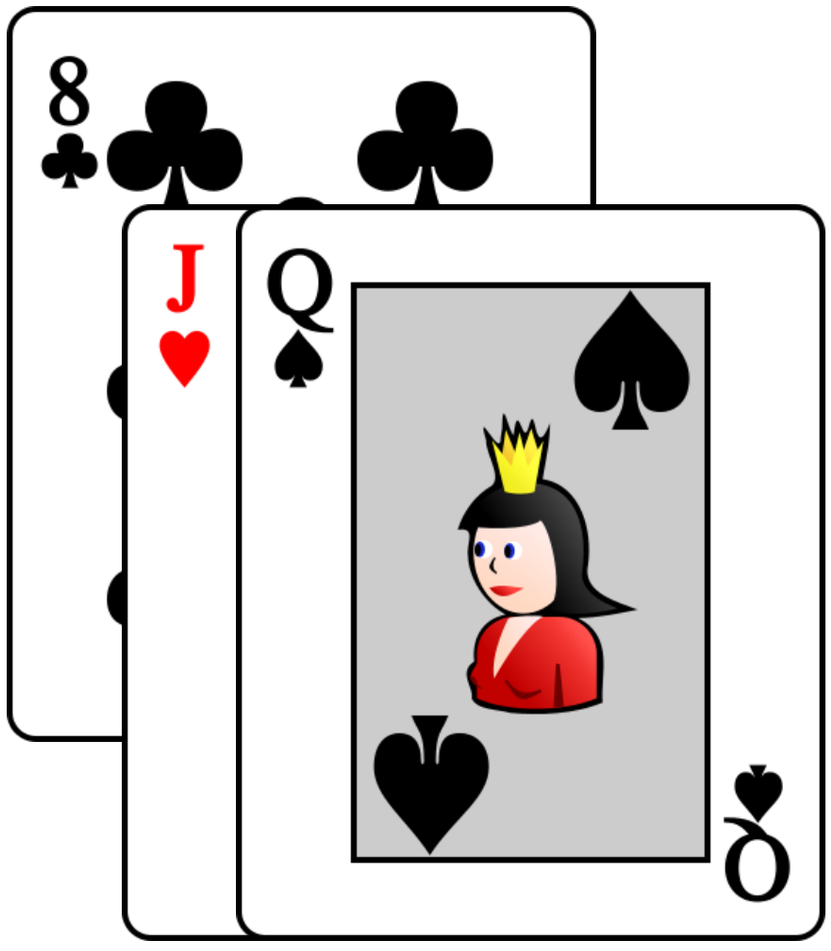
№ operations = 1 + 2 + 1

Le pire des cas



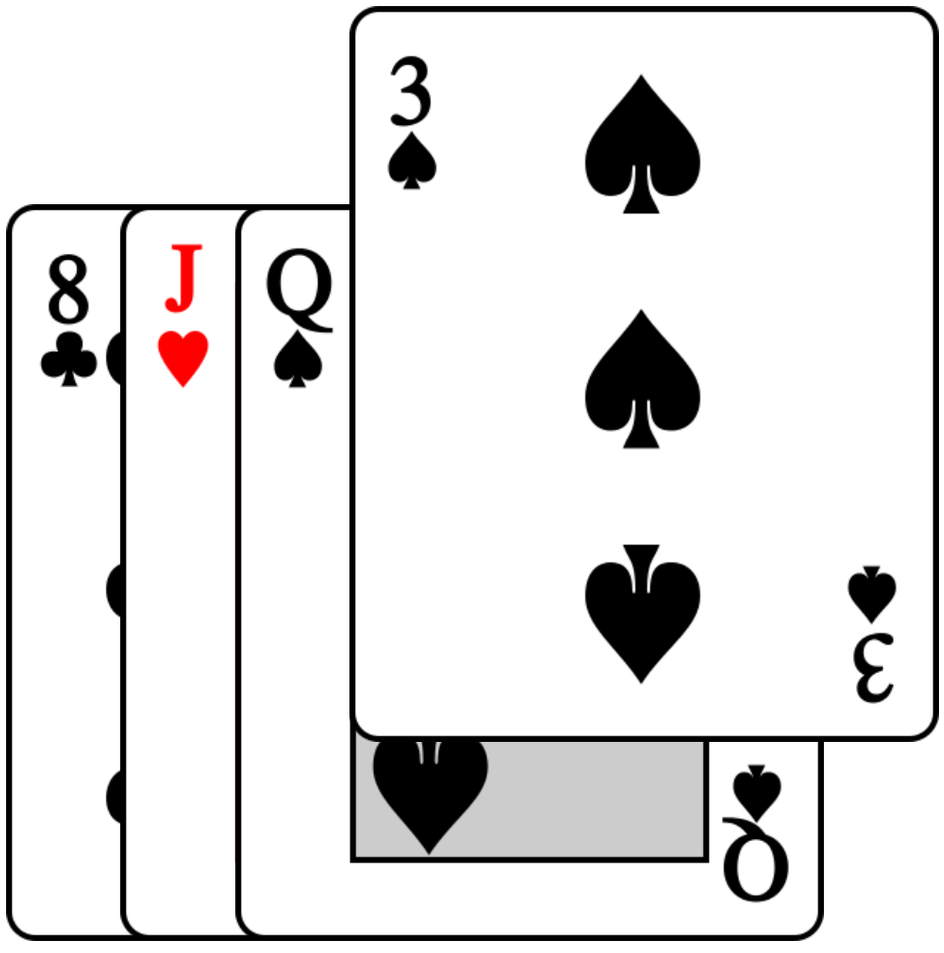
№ operations = 1 + 2 + 2

Le pire des cas



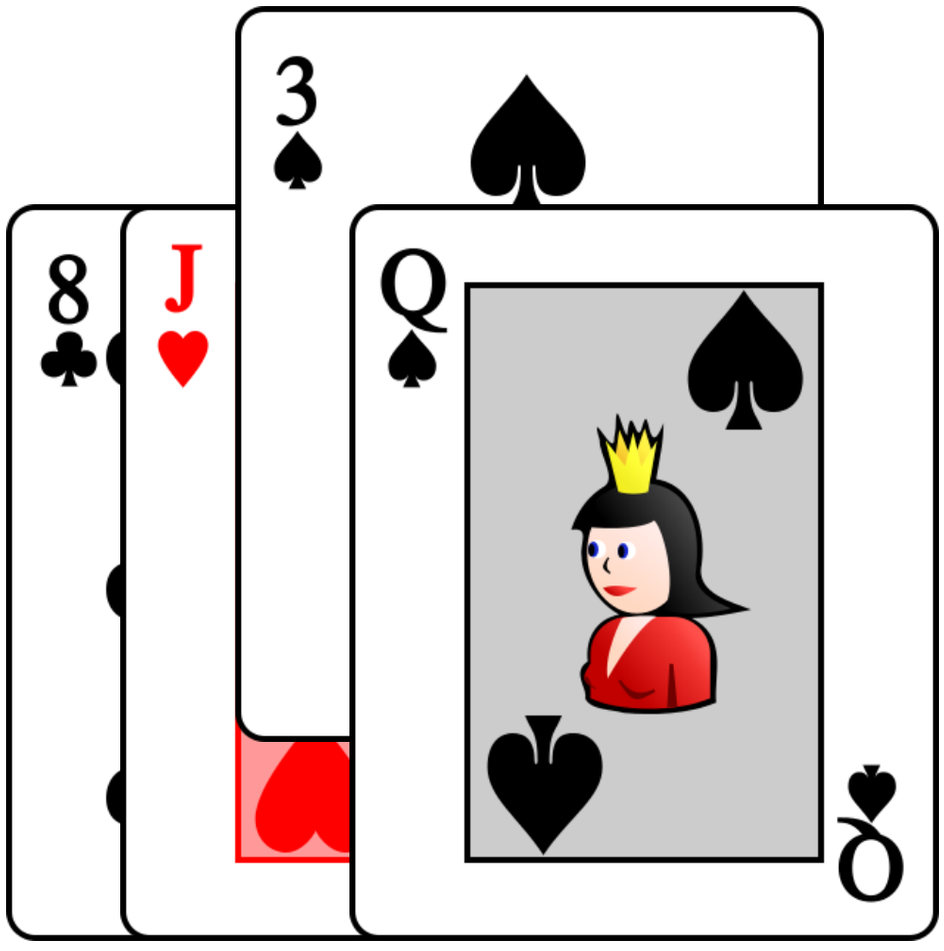
№ operations = 1 + 2 + 3

Le pire des cas



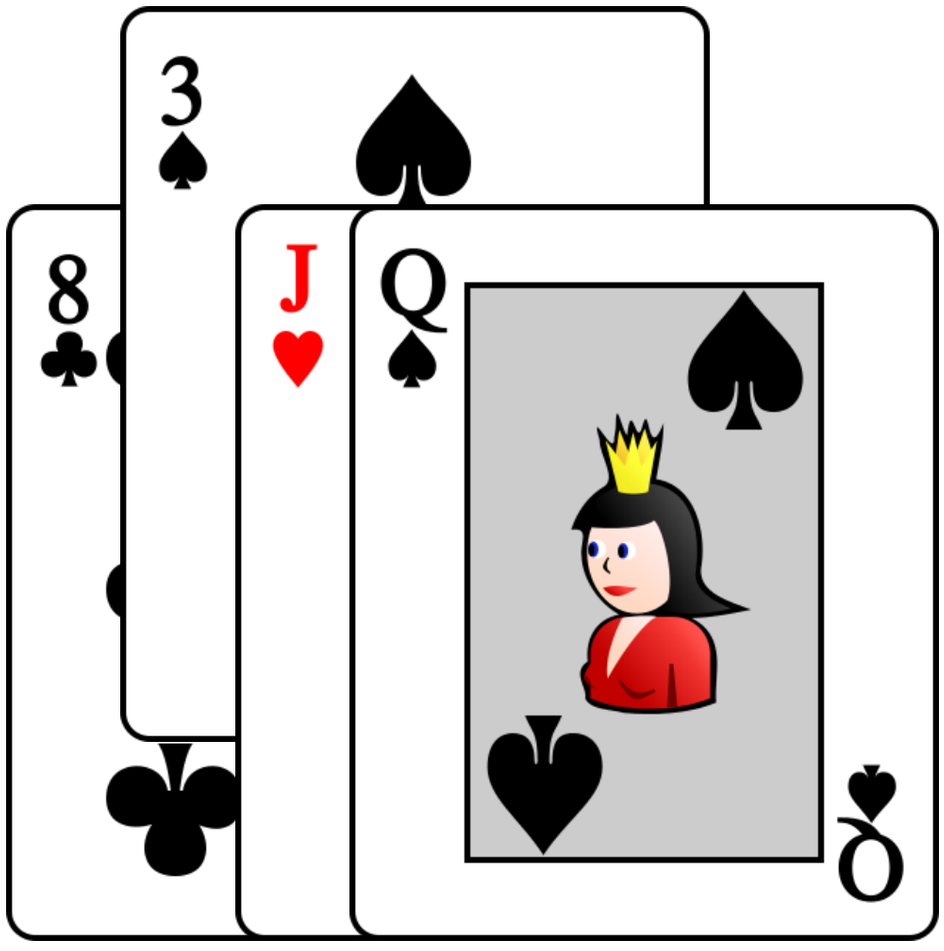
№ operations = 1 + 2 + 3 + 1

Le pire des cas



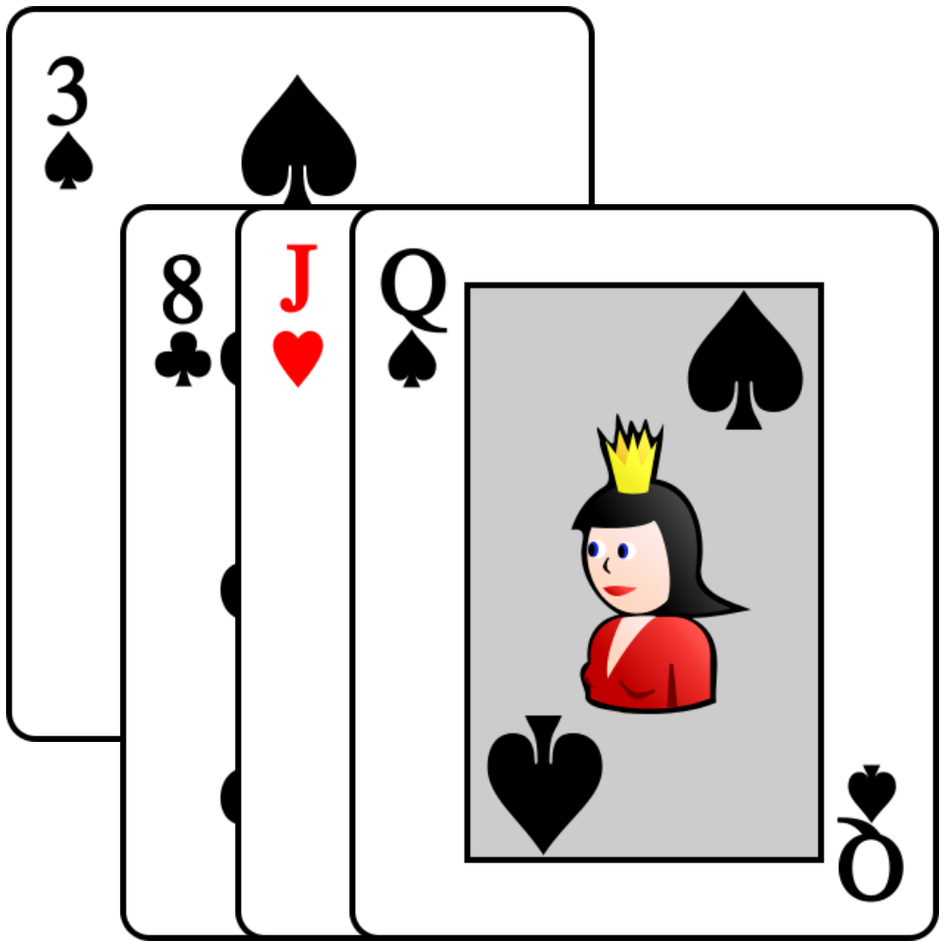
№ operations = 1 + 2 + 3 + 2

Le pire des cas



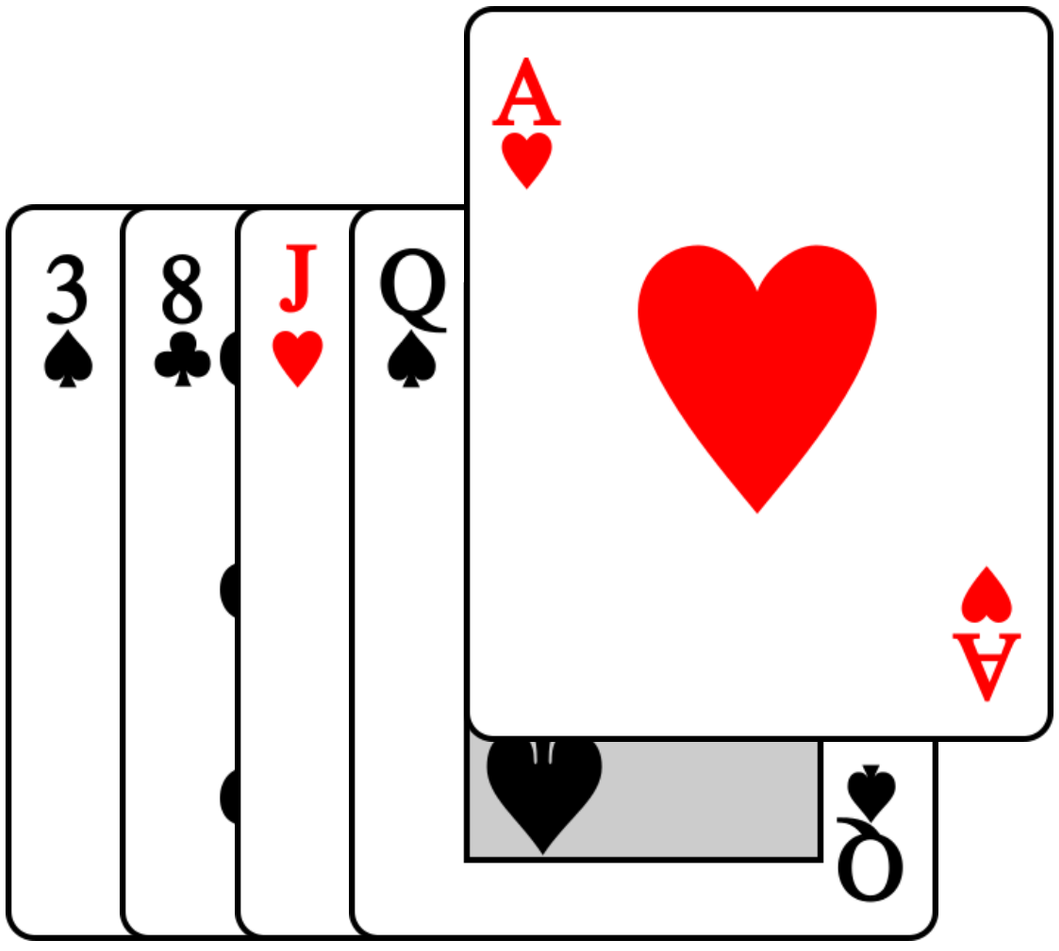
№ operations = 1 + 2 + 3 + 3

Le pire des cas



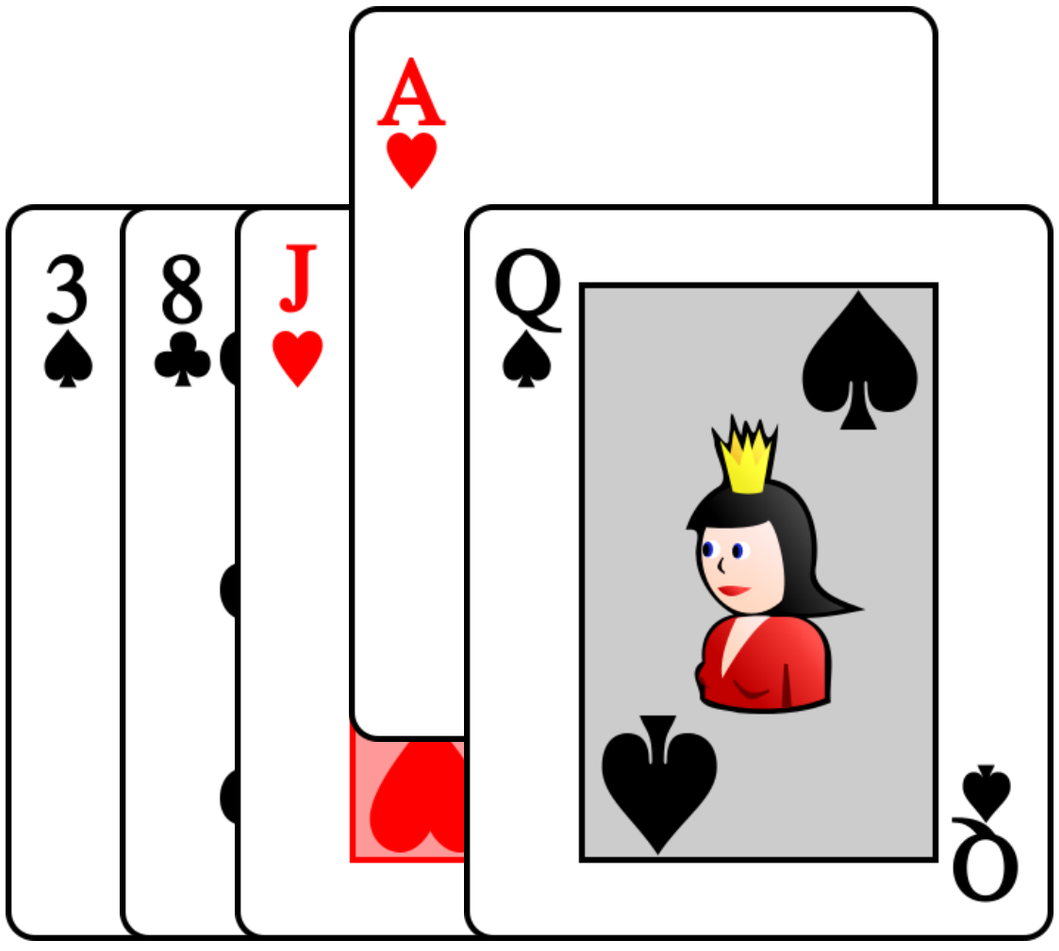
№ operations = 1 + 2 + 3 + 4

Le pire des cas



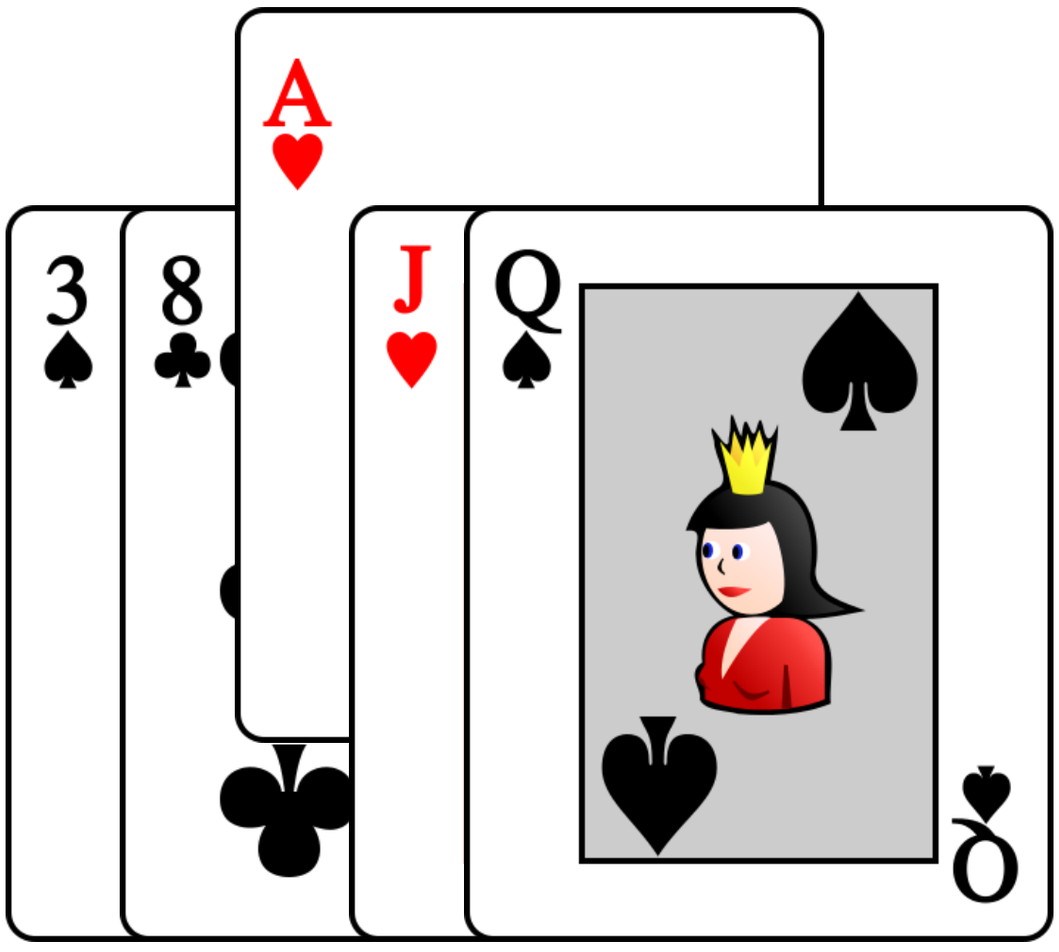
№ operations = 1 + 2 + 3 + 4 + 1

Le pire des cas



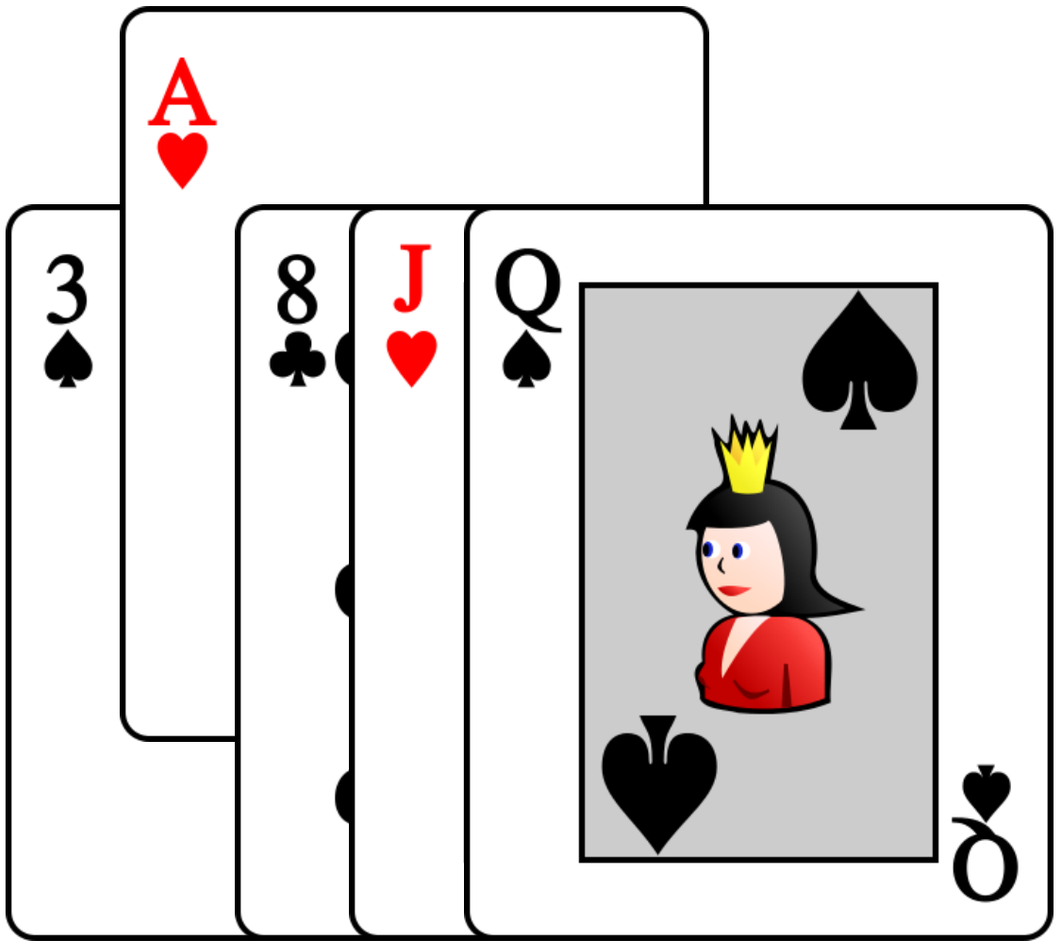
№ operations = 1 + 2 + 3 + 4 + 2

Le pire des cas



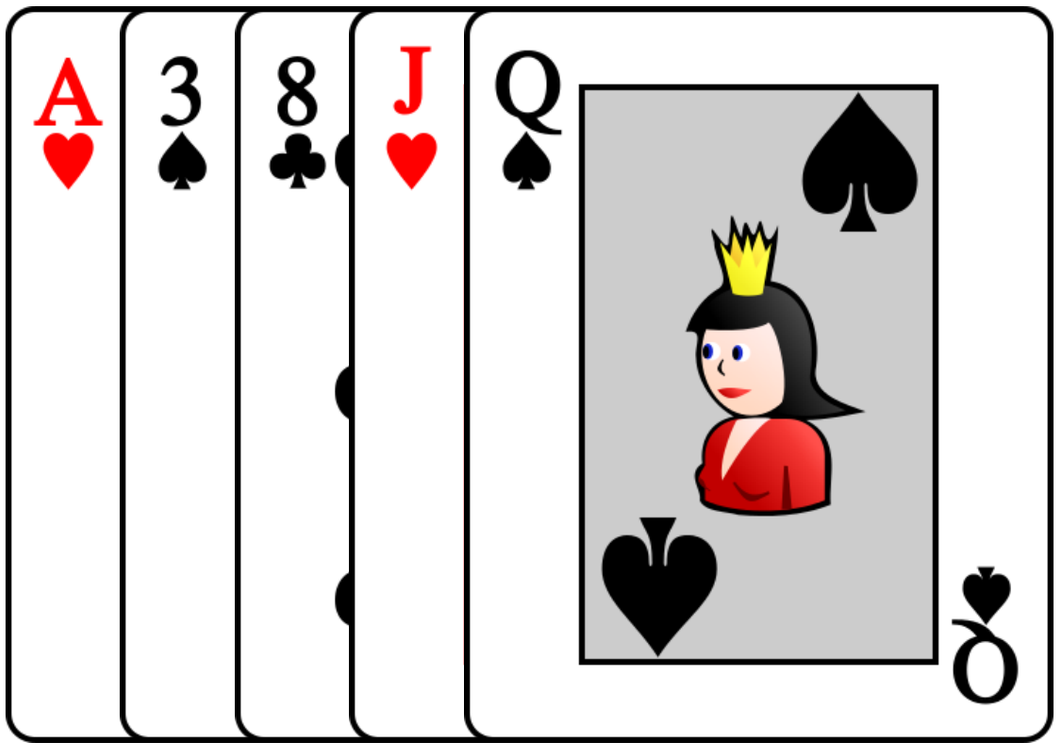
№ operations = 1 + 2 + 3 + 4 + 3

Le pire des cas



№ operations = 1 + 2 + 3 + 4 + 4

Le pire des cas



№ operations = 1 + 2 + 3 + 4 + 5

Le pire des cas

- Les cartes arrivent en ordre décroissant
- On fait i opérations pour la i-ème carte
- Le nombre totale est $1 + 2 + 3 + \cdots + n$

$$\sum_{i=1}^{n} i = \frac{1}{2}n(n+1) = \frac{1}{2}(n^2 + n) = \frac{1}{2}n^2 + \frac{1}{2}n \in O(n^2)$$

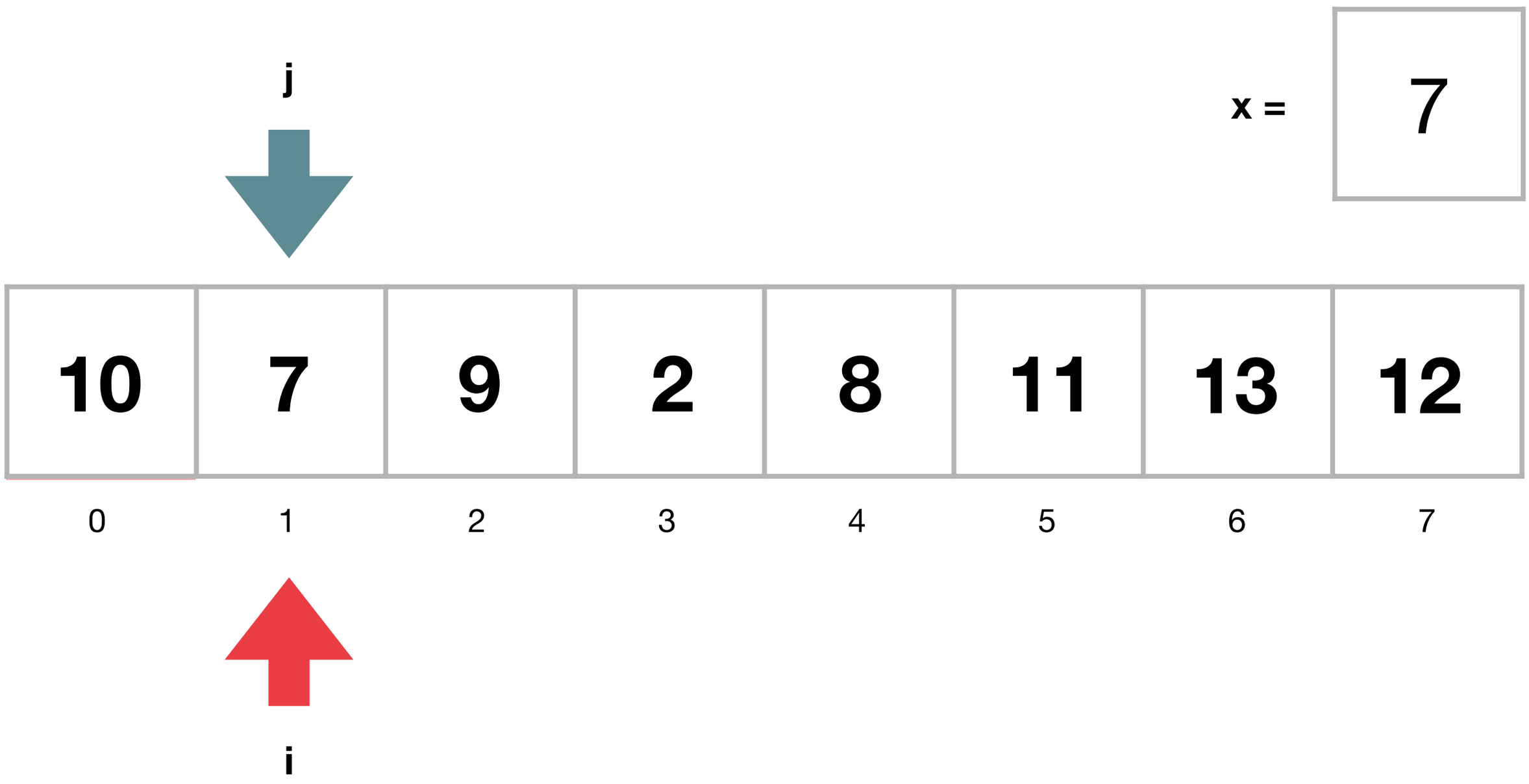
Tri par insertion

```
def trier_par_insertion(A) :
    n = len(A)
    for i in range(1, n):
        x = A[i]
        # insérer x parmi les i premiers éléments
        j = i
        while j > 0 and x < A[j - 1]:
            # décaler d'un élément
            A[j] = A[j - 1]
            j = j - 1
            # ici, x >= A[j - 1] ou bien j == 0
        A[j] = x
        # le tableau est trié !
```

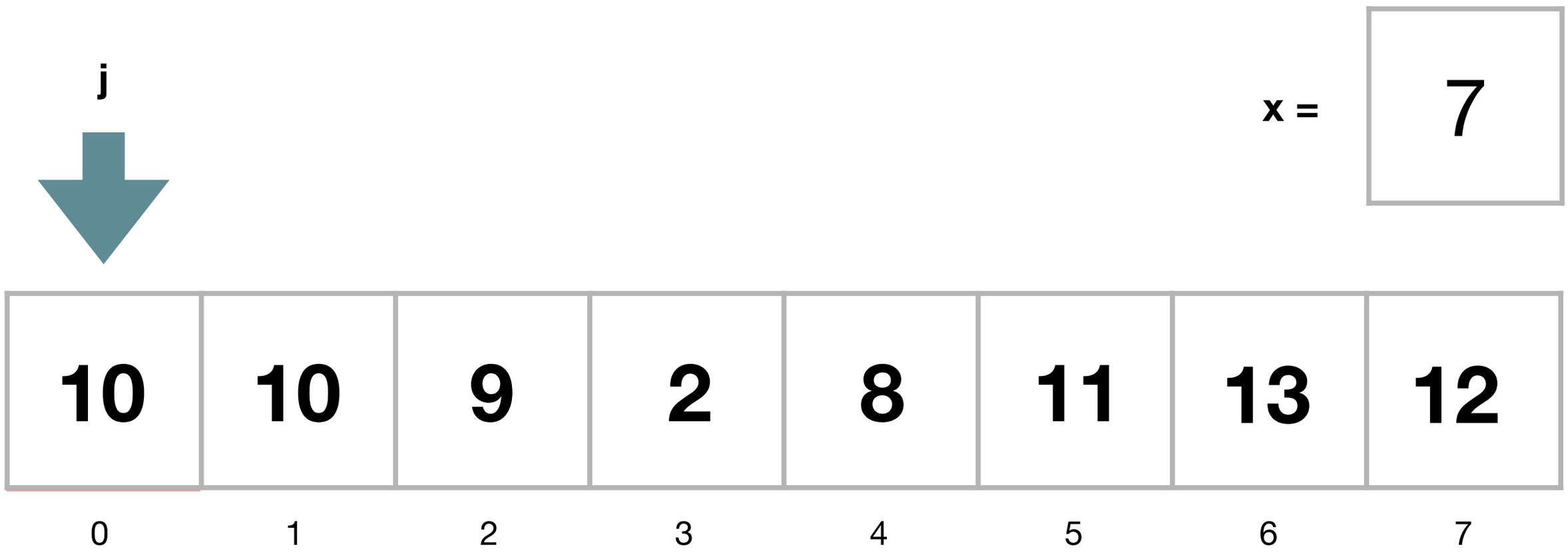
Tri par insertion

10	7	9	2	8	11	13	12
0	1	2	3	4	5	6	7

Tri par insertion



Tri par insertion



Tri par insertion

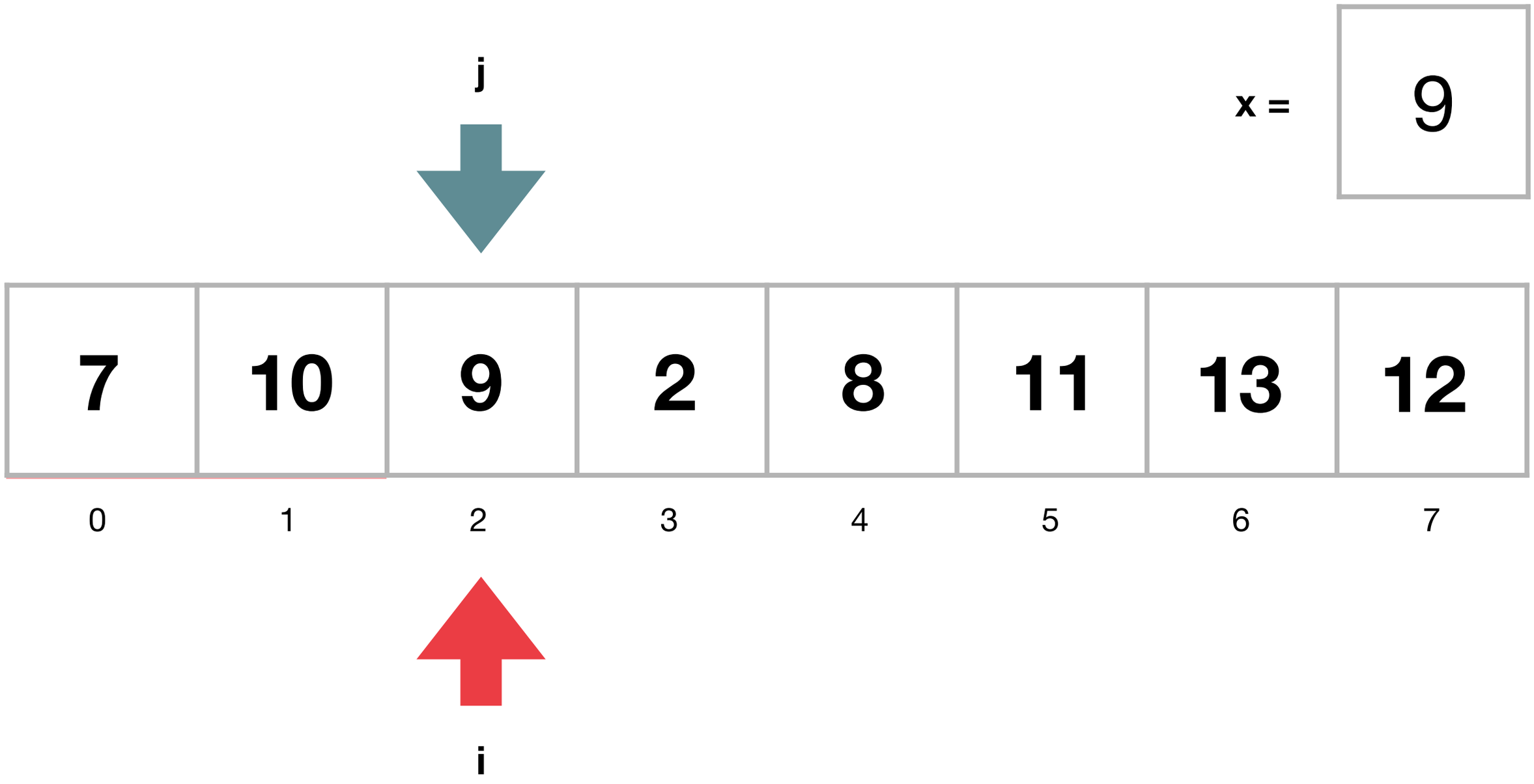
j

x = 7

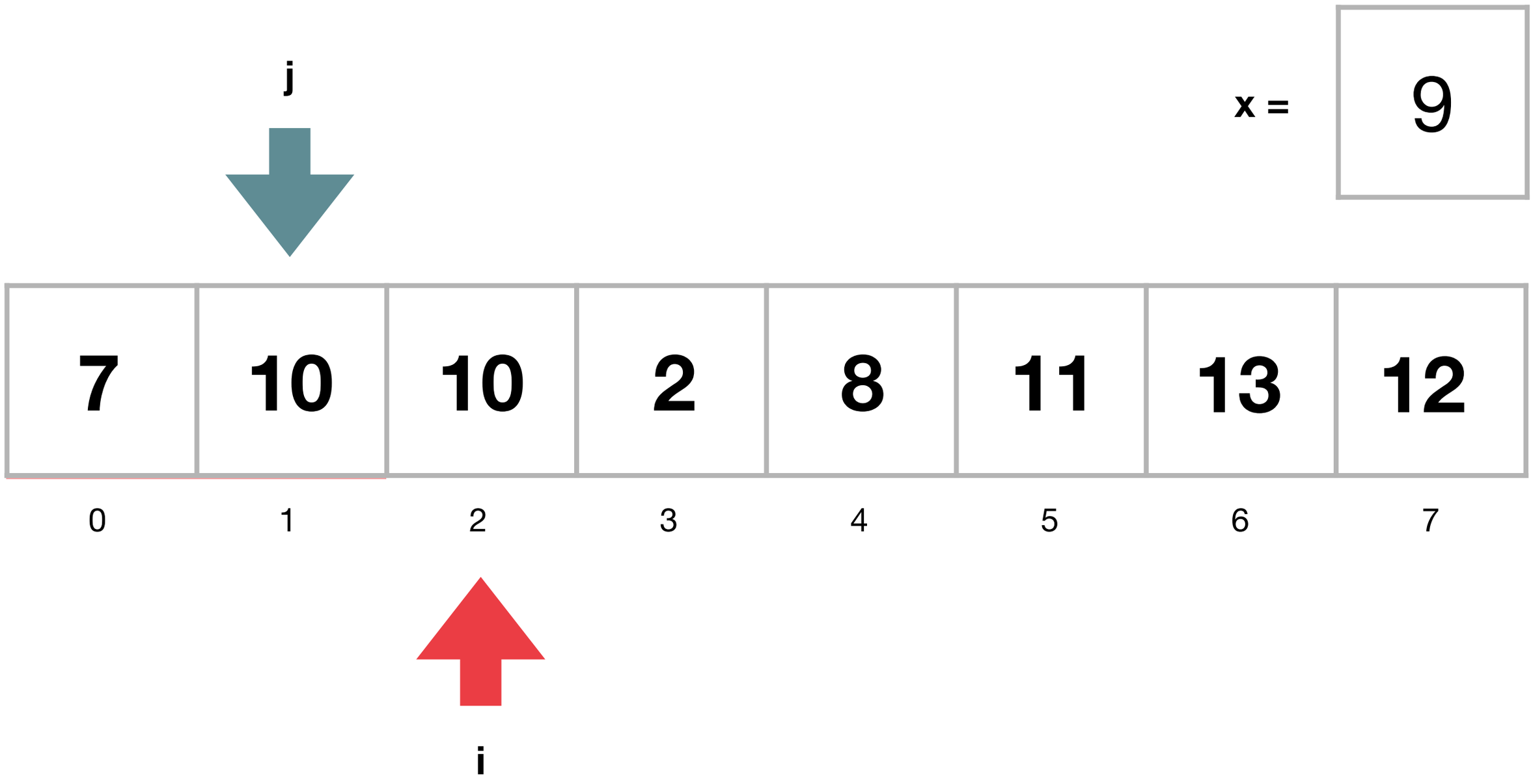
7	10	9	2	8	11	13	12
0	1	2	3	4	5	6	7

i

Tri par insertion



Tri par insertion



Tri par insertion

j

x = 9

7	9	10	2	8	11	13	12
0	1	2	3	4	5	6	7

i

Tri par insertion

j

x = 2

7	9	10	2	8	11	13	12
0	1	2	3	4	5	6	7

i

Tri par insertion

j

x = 2

7	9	10	10	8	11	13	12
0	1	2	3	4	5	6	7

i

Tri par insertion

j

x = 2

7	9	9	10	8	11	13	12
0	1	2	3	4	5	6	7

i

Tri par insertion

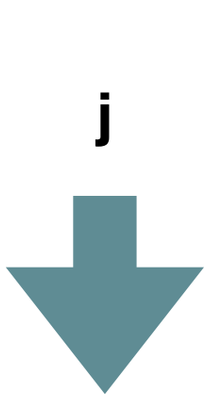
j

x = 2

7	7	9	10	8	11	13	12
0	1	2	3	4	5	6	7

i

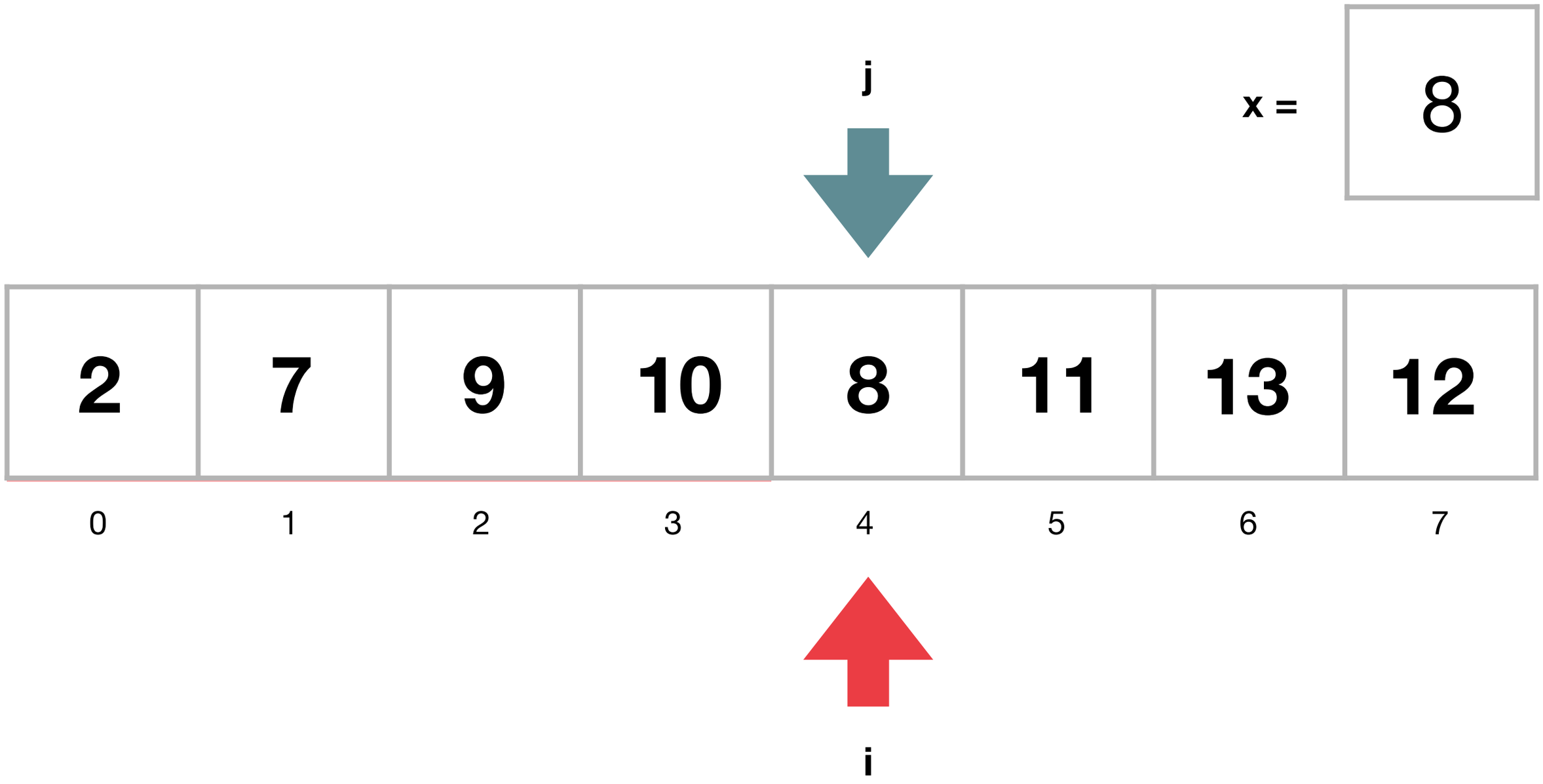
Tri par insertion



x = 2

2	7	9	10	8	11	13	12
0	1	2	3	4	5	6	7

Tri par insertion



Tri par insertion

j

x = 8

2	7	9	10	10	11	13	12
0	1	2	3	4	5	6	7

i

Tri par insertion

j

x = 8

2	7	9	9	10	11	13	12
0	1	2	3	4	5	6	7

i

Tri par insertion

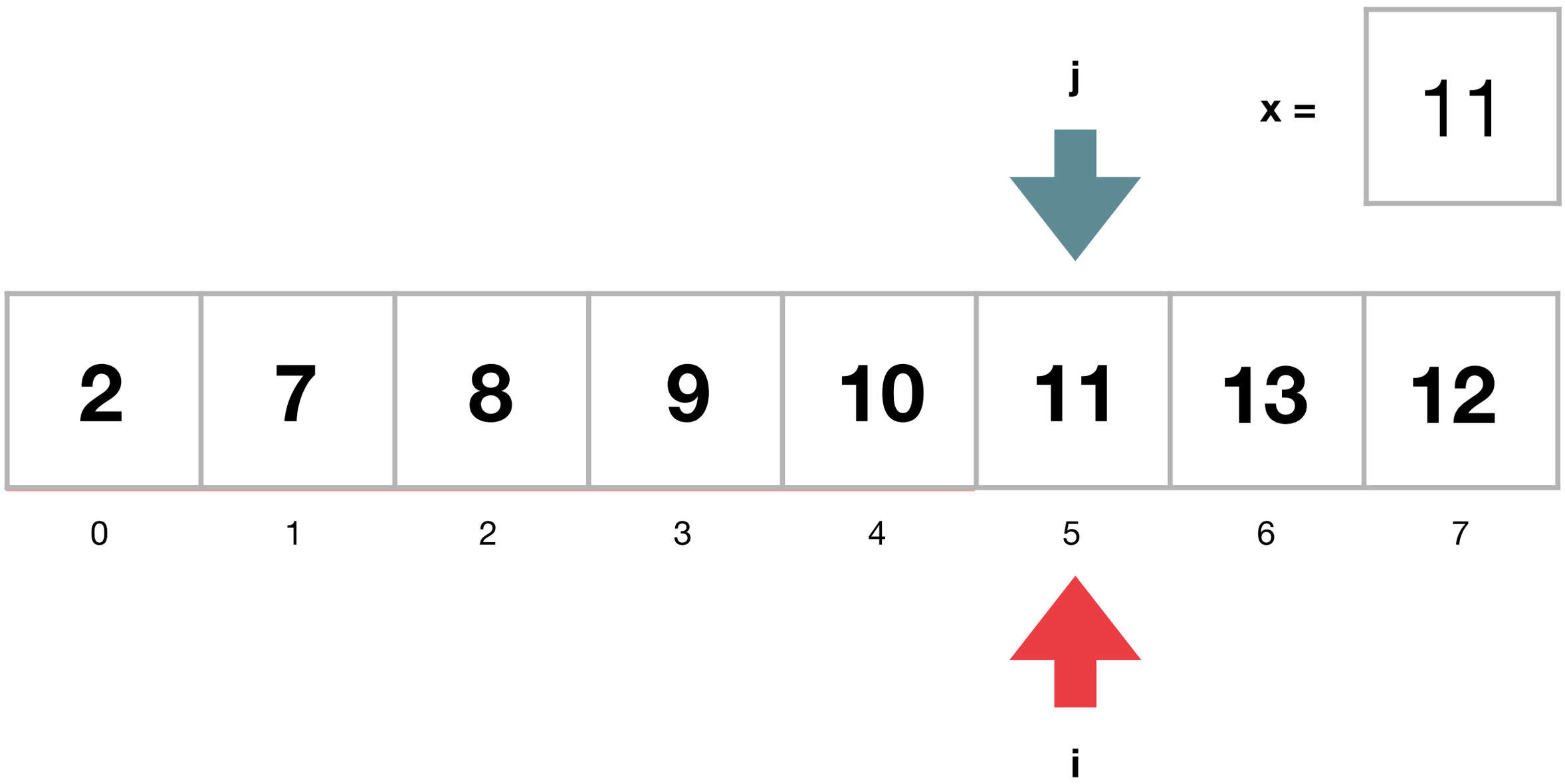
j

x = 8

2	7	8	9	10	11	13	12
0	1	2	3	4	5	6	7

i

Tri par insertion



Tri par insertion

x = 13

j

2	7	8	9	10	11	13	12
0	1	2	3	4	5	6	7

i

Tri par insertion

x = 12

j

2	7	8	9	10	11	13	12
0	1	2	3	4	5	6	7

i

Tri par insertion

x = 12

j

2	7	8	9	10	11	13	13
0	1	2	3	4	5	6	7

i

Tri par insertion

x = 12

j

2	7	8	9	10	11	12	13
0	1	2	3	4	5	6	7

i

Terminaison

```
def trier_par_insertion(A) :
    n = len(A)
    for i in range(1, n):
        x = A[i]
        j = i
        while j > 0 and x < A[j - 1]:
            A[j] = A[j - 1]
            j = j - 1
        A[j] = x
```

Terminaison

```
def trier_par_insertion(A) :
    n = len(A)
    for i in range(1, n):
        x = A[i]
        j = i
        while j > 0 and x < A[j - 1]:
            A[j] = A[j - 1]
            j = j - 1
        A[j] = x
```

- La boucle **for** termine toujours

Terminaison

```
def trier_par_insertion(A) :
    n = len(A)
    for i in range(1, n):
        x = A[i]
        j = i
        while j > 0 and x < A[j - 1]:
            A[j] = A[j - 1]
            j = j - 1
        A[j] = x
```

- La boucle **`for`** termine toujours
- La boucle **`while`** termine (au pire) quand $j = 0$

Correction

```
def trier_par_insertion(A) :
    n = len(A)
    for i in range(1, n):
        x = A[i]
        j = i
        while j > 0 and x < A[j - 1]:
            A[j] = A[j - 1]
            j = j - 1
            A[j] = x
```

Correction

```
def trier_par_insertion(A) :
    n = len(A)
    for i in range(1, n):
        x = A[i]
        j = i
        while j > 0 and x < A[j - 1]:
            A[j] = A[j - 1]
            j = j - 1
            A[j] = x
```

- Le sous-tableau $A[0,\ldots,i-1]$ est trié au début des instructions de la boucle **for**

Correction

```
def trier_par_insertion(A) :
    n = len(A)
    for i in range(1, n):
        x = A[i]
        j = i
        while j > 0 and x < A[j - 1]:
            A[j] = A[j - 1]
            j = j - 1
            A[j] = x
```

- Le sous-tableau $A[0,\ldots,i-1]$ est trié au début des instructions de la boucle **for**

- À la fin on a $i = n - 1$, donc le tableau $A[0,\ldots,n-2]$ est trié, et on déplace l'élément $A[n-1]$ à la bonne position, ce qui donne un tableau entier trié

Efficacité

```
def trier_par_insertion(A) :
    n = len(A)
    for i in range(1, n):
        x = A[i]
        j = i
        while j > 0 and x < A[j - 1]:
            A[j] = A[j - 1]
            j = j - 1
            A[j] = x
```

Efficacité

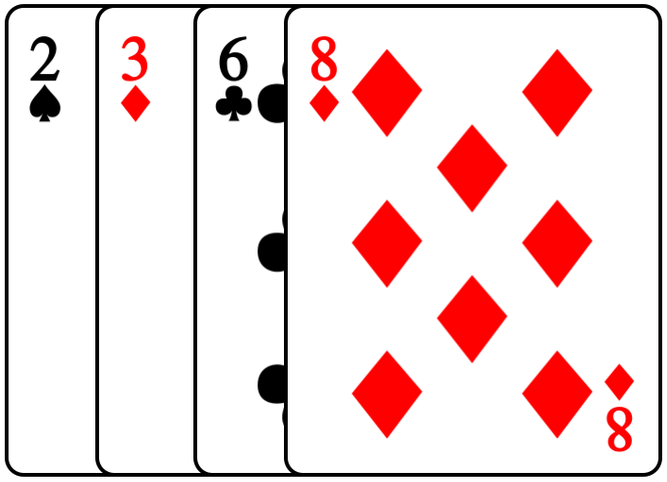
```
def trier_par_insertion(A) :
    n = len(A)
    for i in range(1, n):
        x = A[i]
        j = i
        while j > 0 and x < A[j - 1]:
            A[j] = A[j - 1]
            j = j - 1
            A[j] = x
```

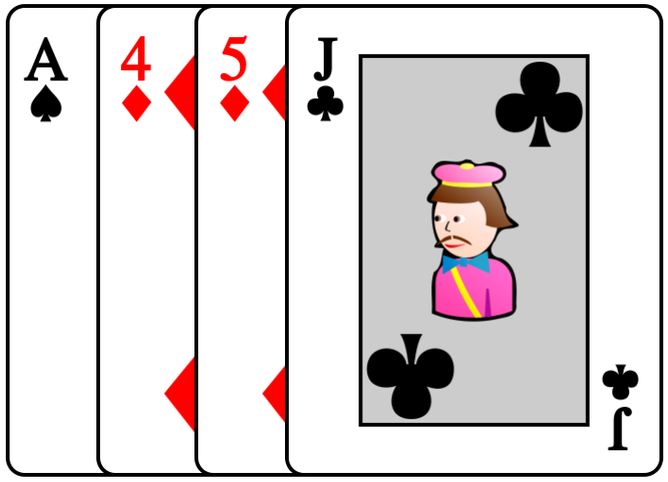
- $O(n)$ opérations dans le meilleur des cas

- $O(n^2)$ opérations dans le pire des cas

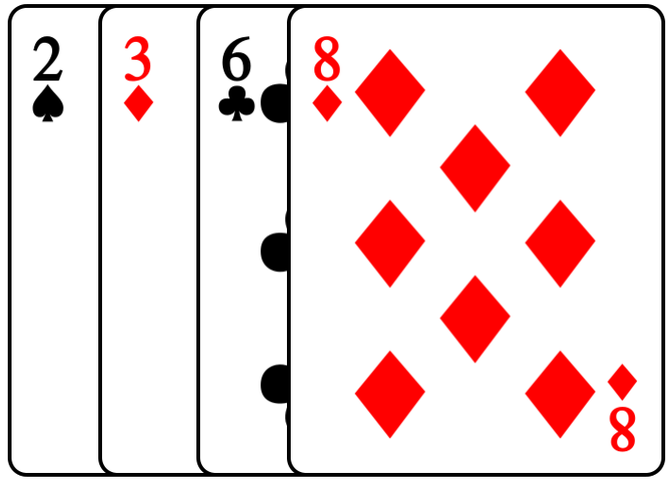
Peut-on faire mieux que $O(n^2)$ pour trier un tableau ?

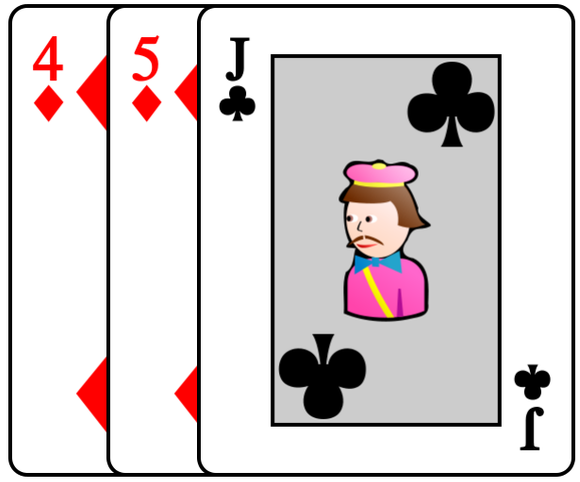
Fusion de tableaux triés

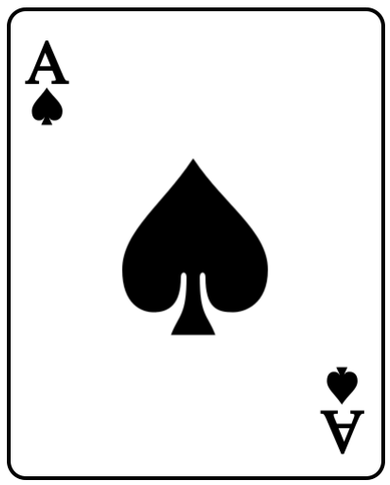




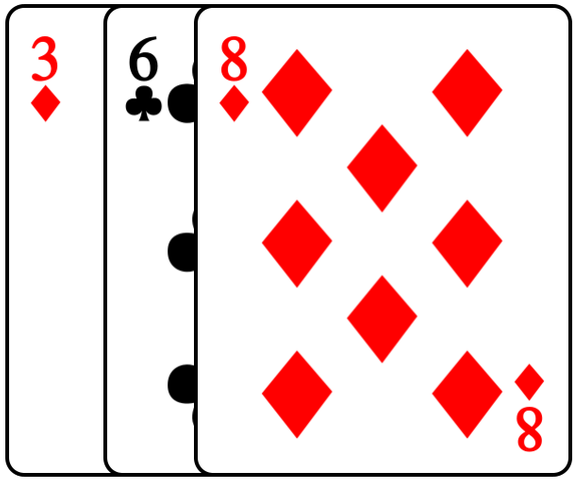
Fusion de tableaux triés

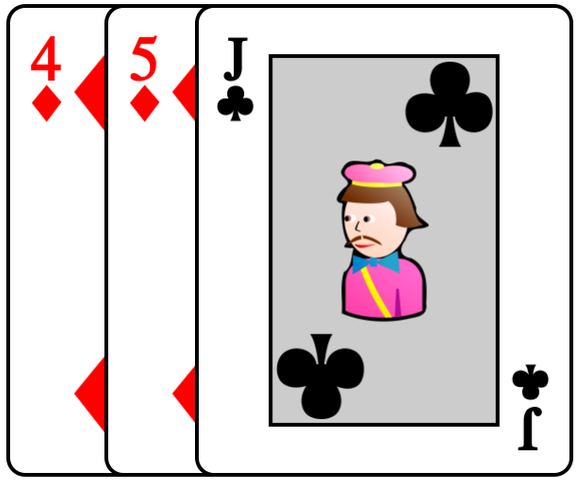


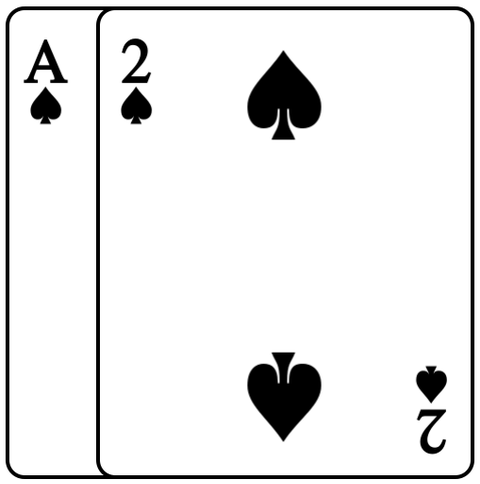




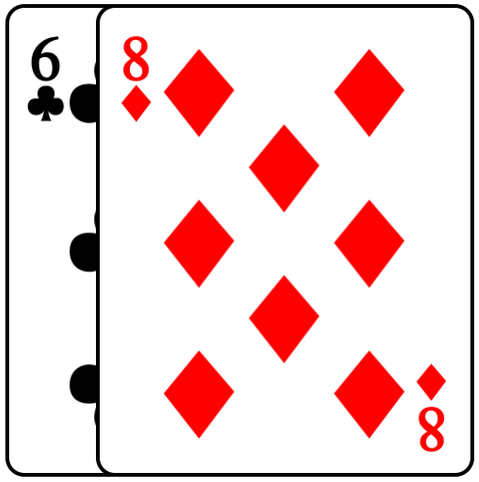
Fusion de tableaux triés

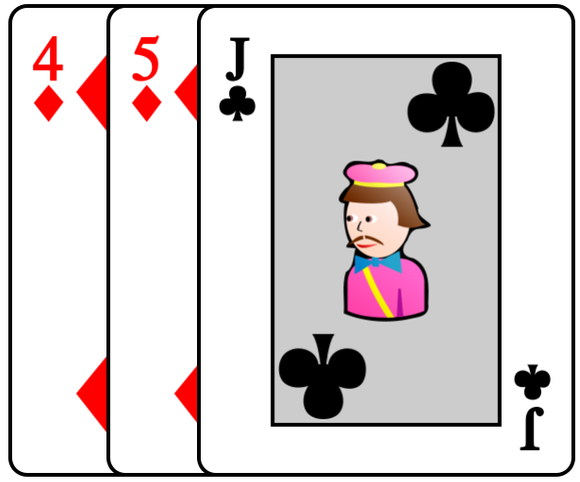


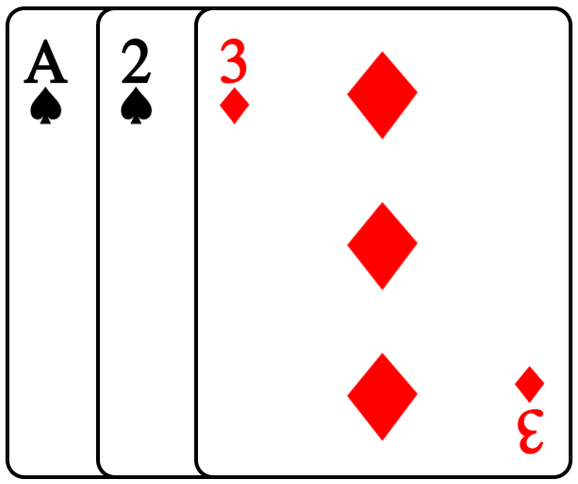




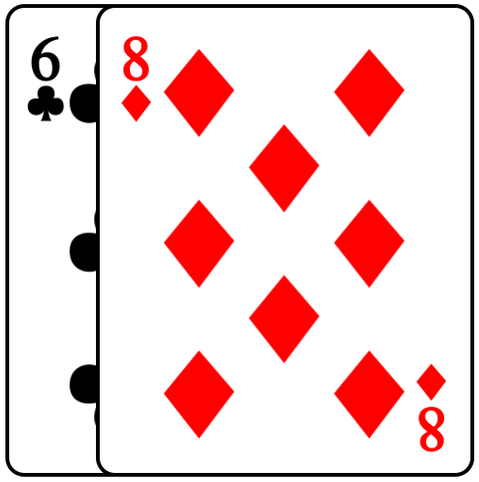
Fusion de tableaux triés

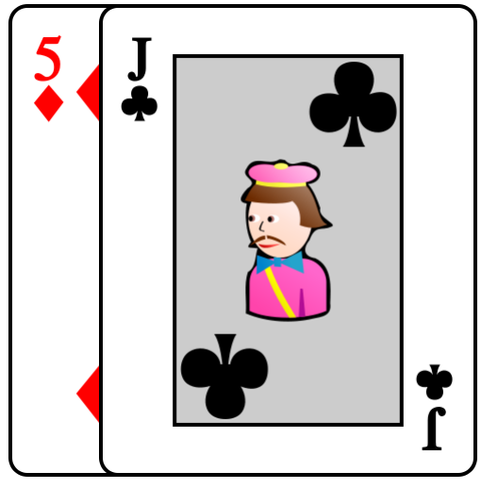


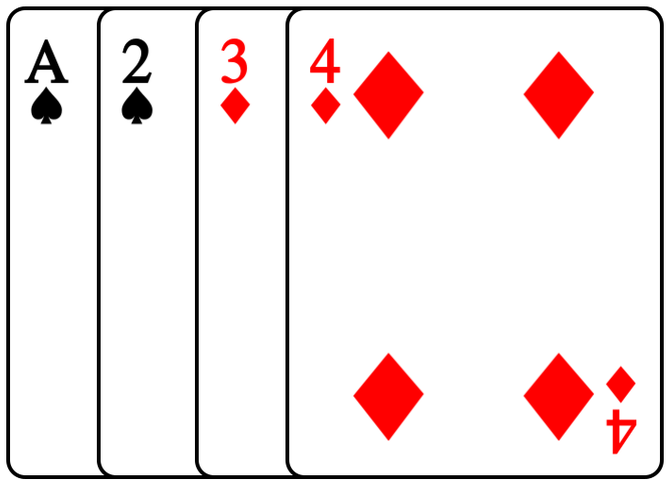




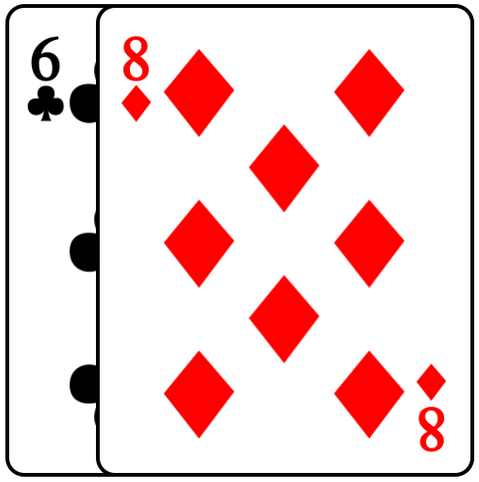
Fusion de tableaux triés

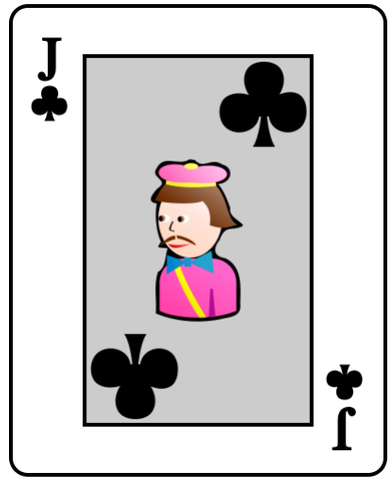


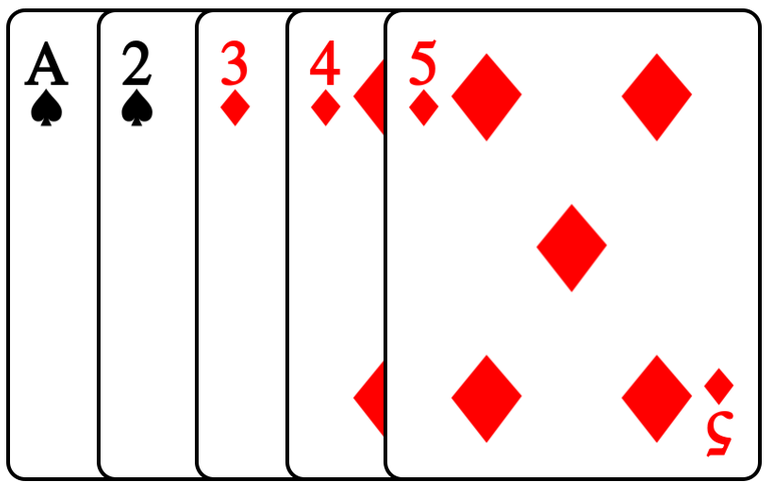




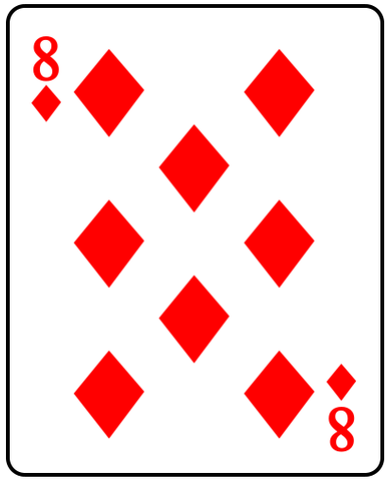
Fusion de tableaux triés

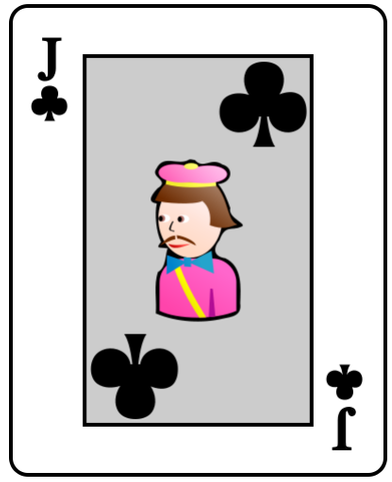


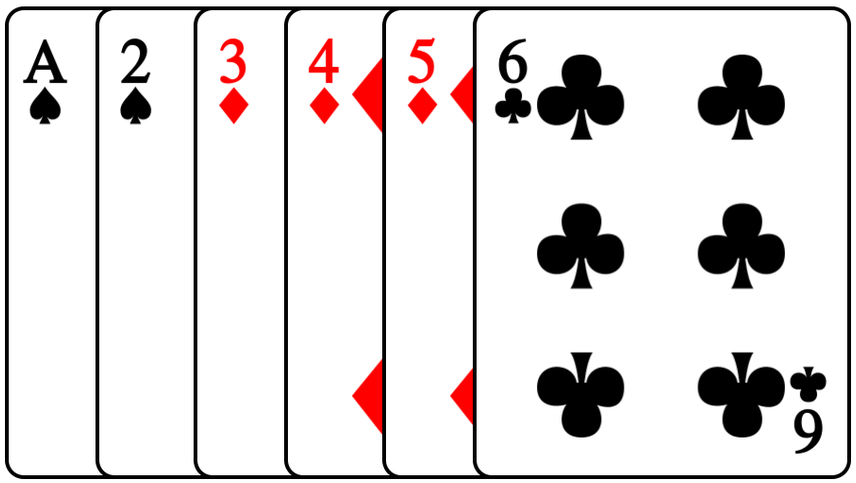




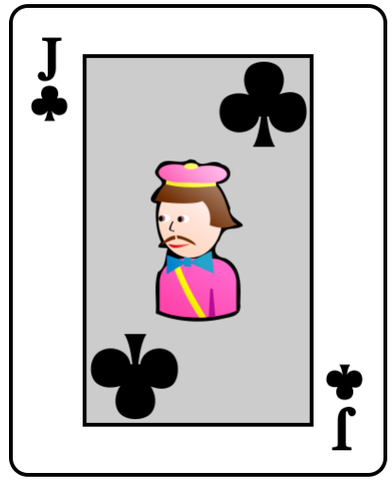
Fusion de tableaux triés

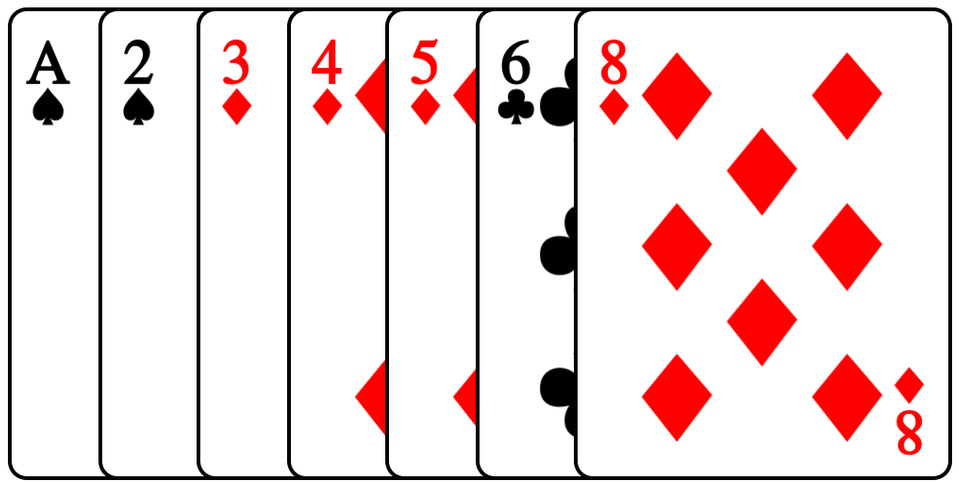




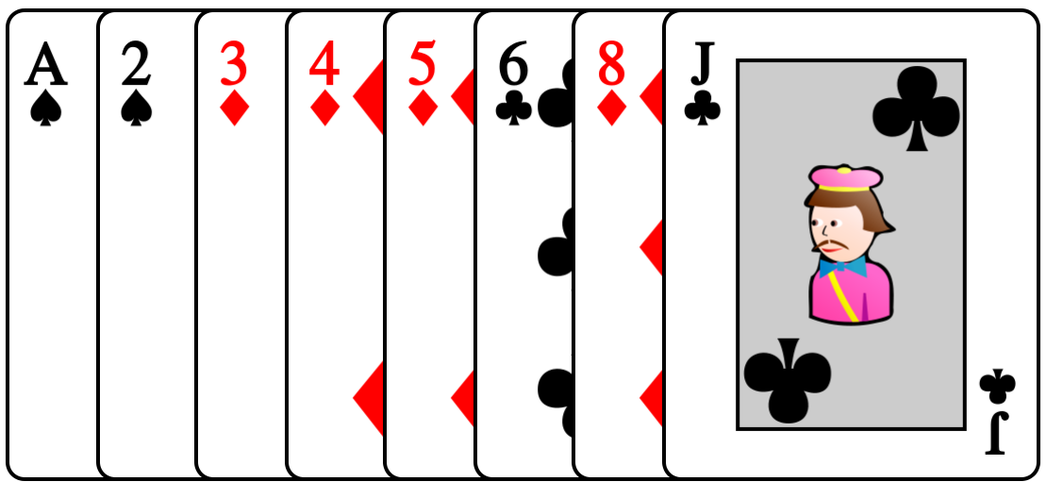


Fusion de tableaux triés



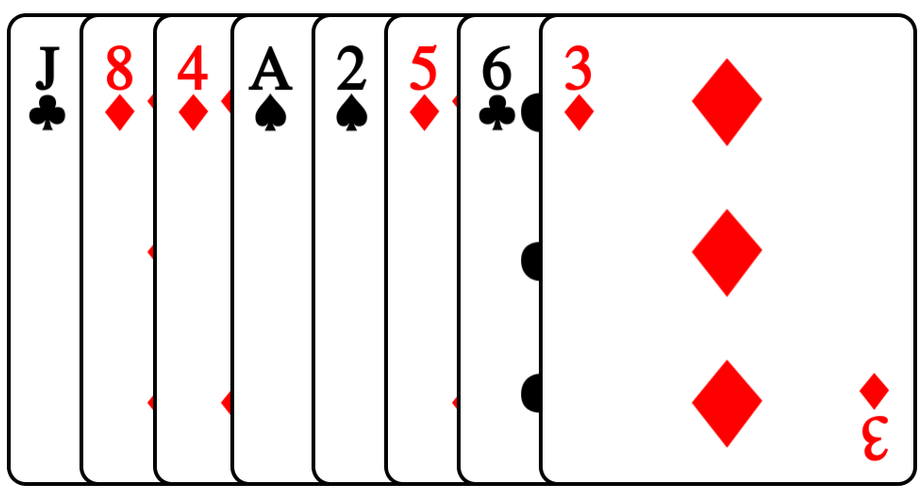


Fusion de tableaux triés

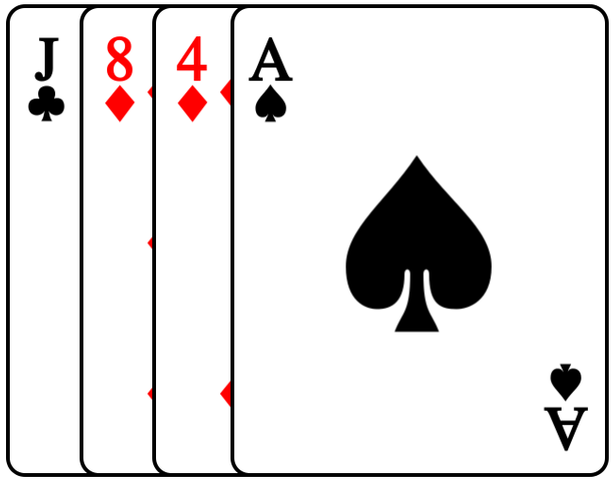


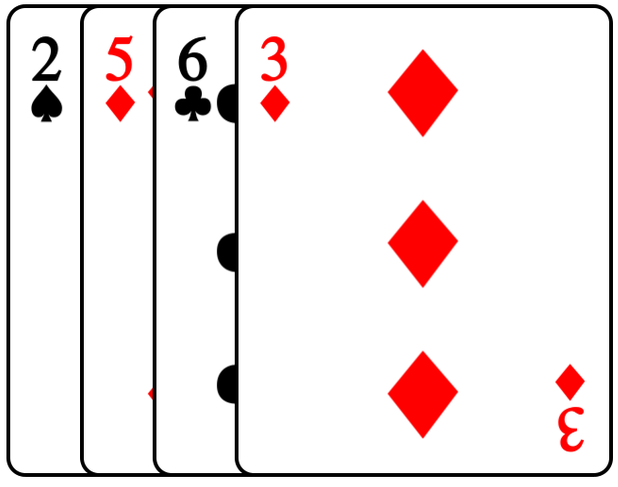
Diviser pour mieux régner !

Tri fusion

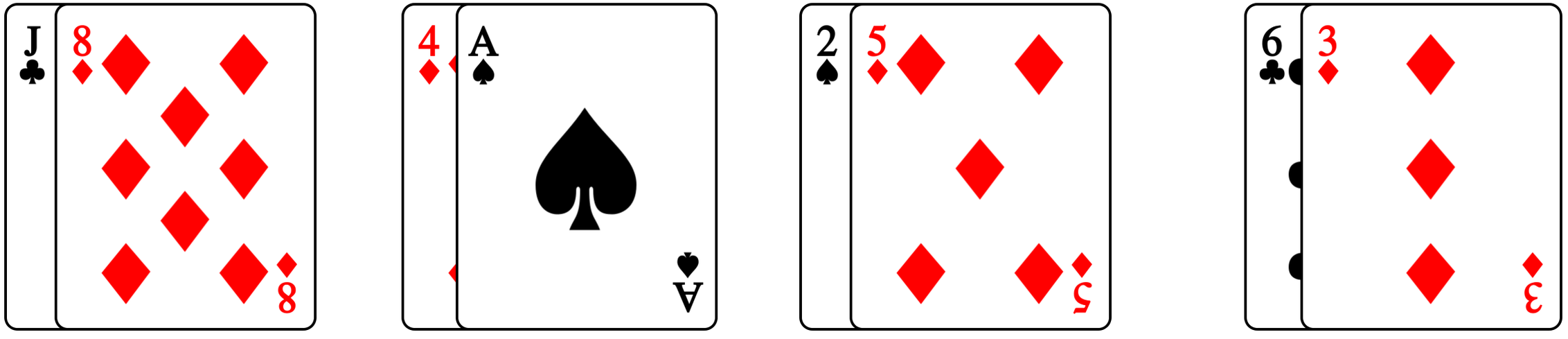


Diviser

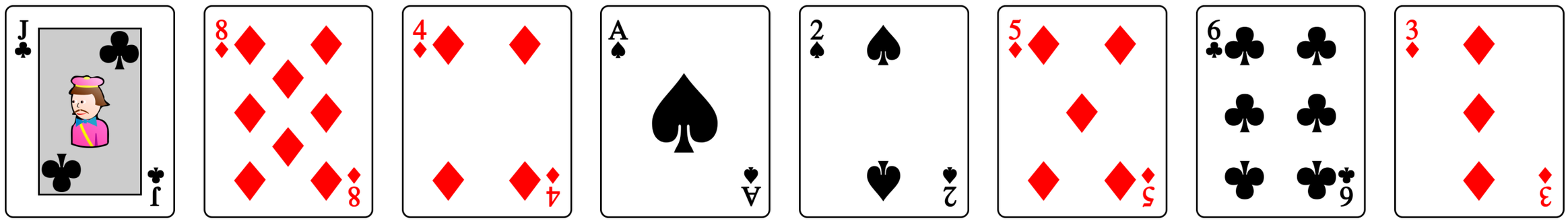




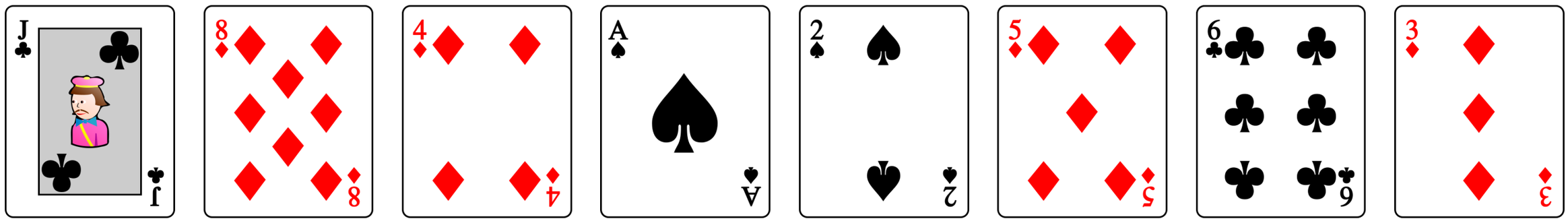
Diviser



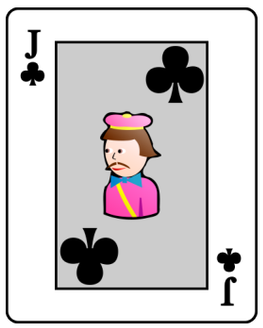
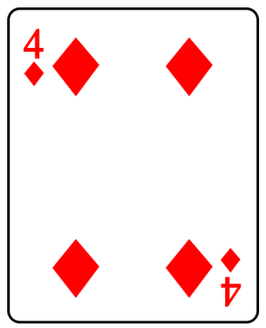
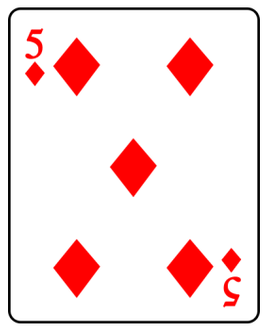
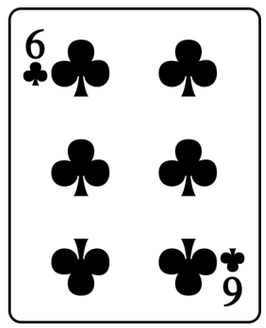
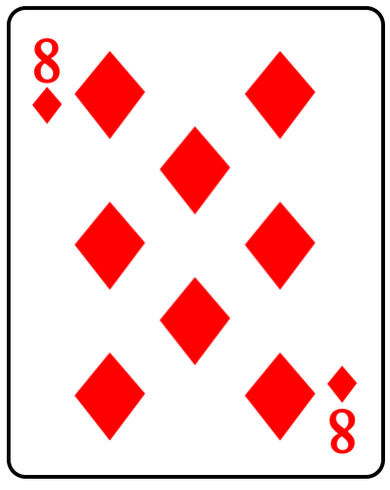
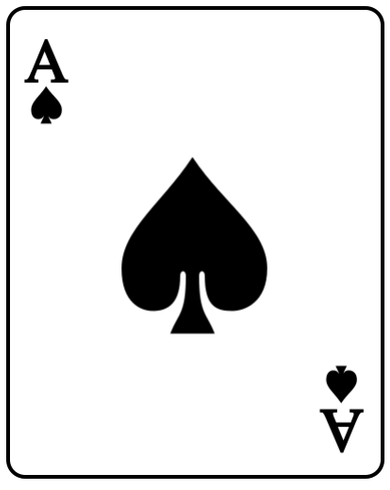
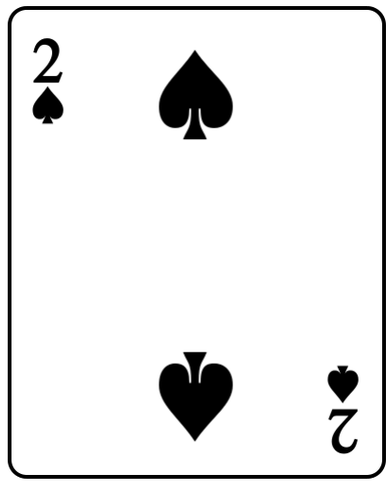
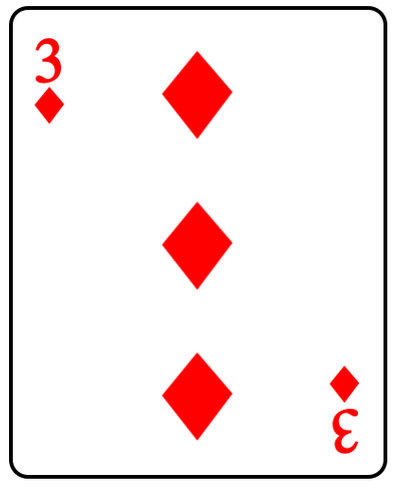
Diviser



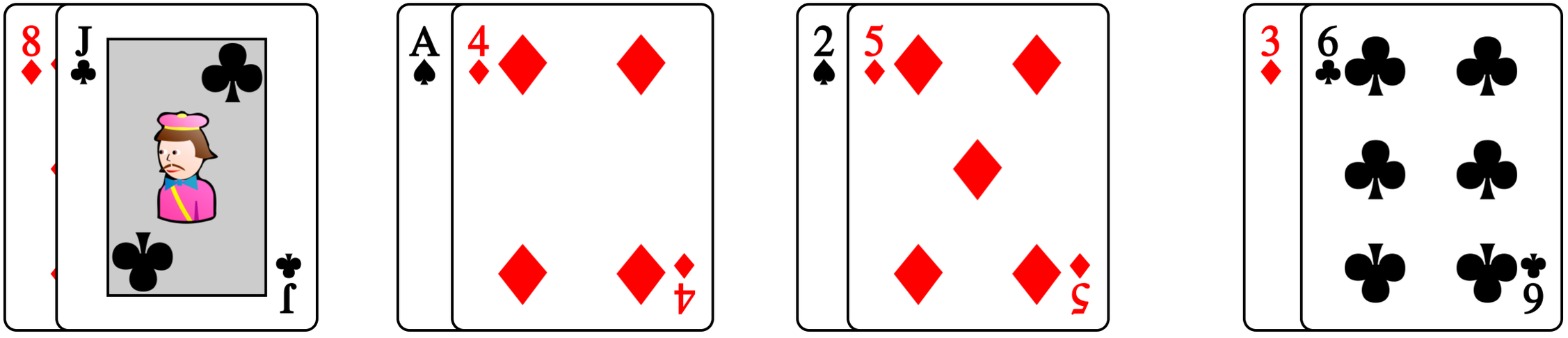
Tous les jeux sont triés !



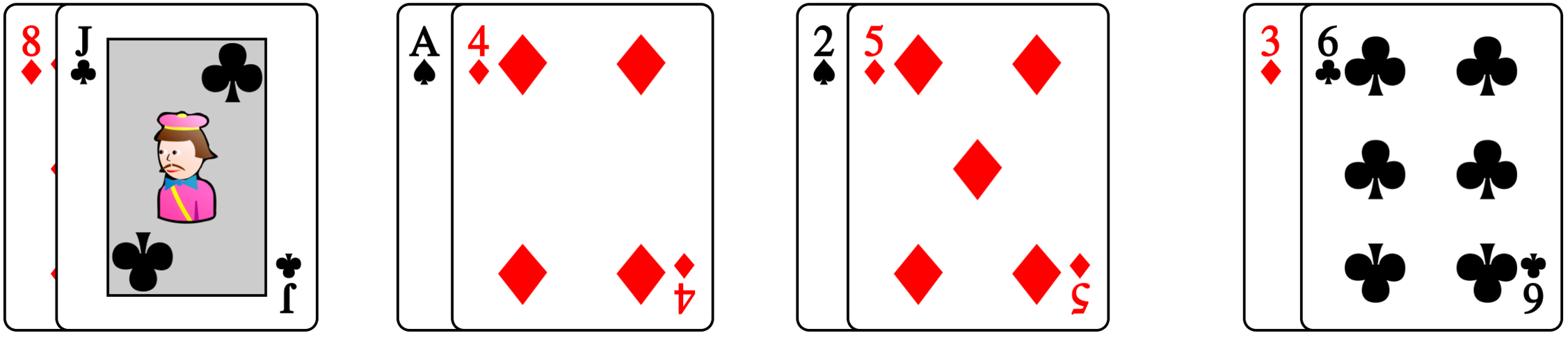
Fusionner

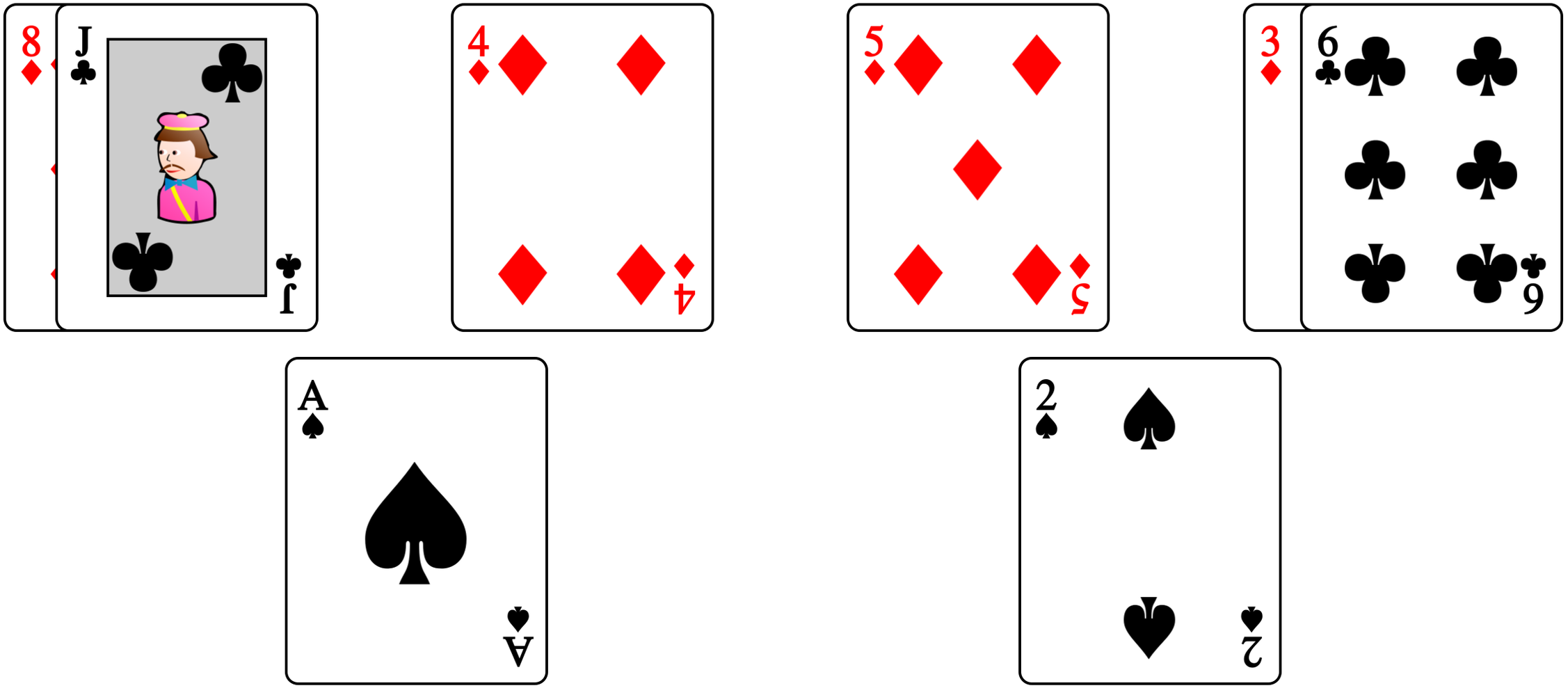
Fusionner



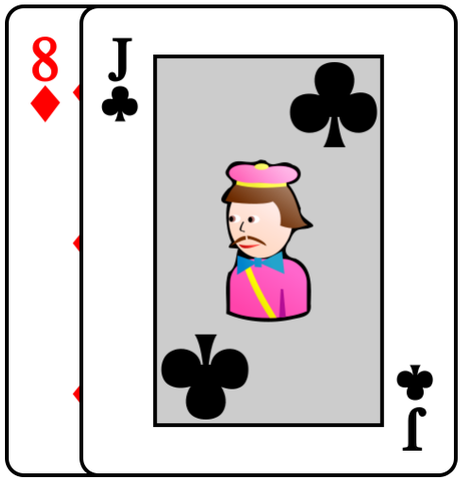
Tous les jeux sont triés !

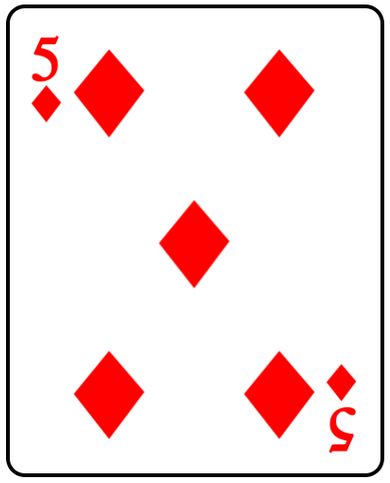


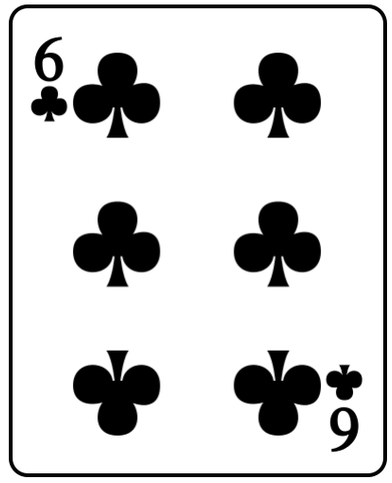
Fusionner

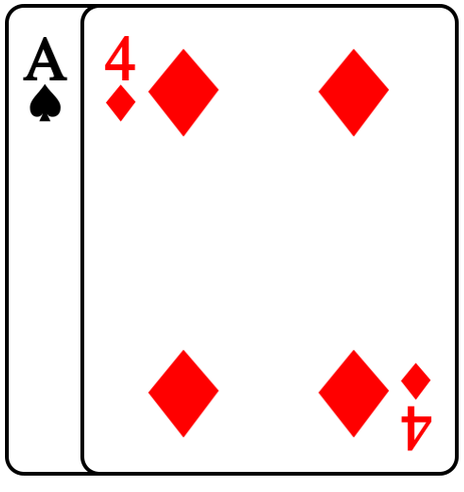


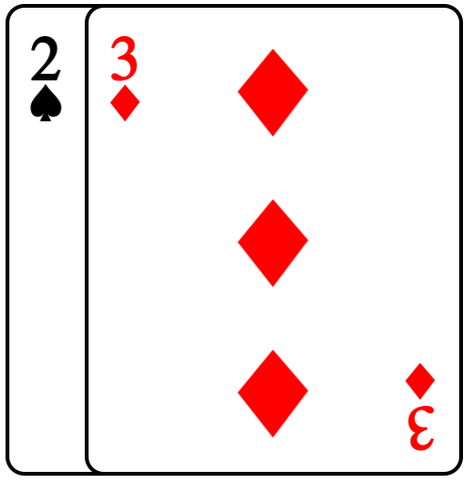
Fusionner



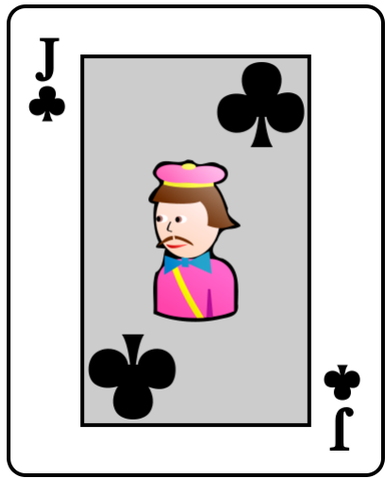


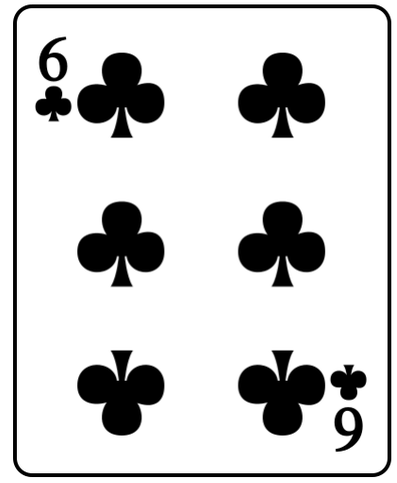


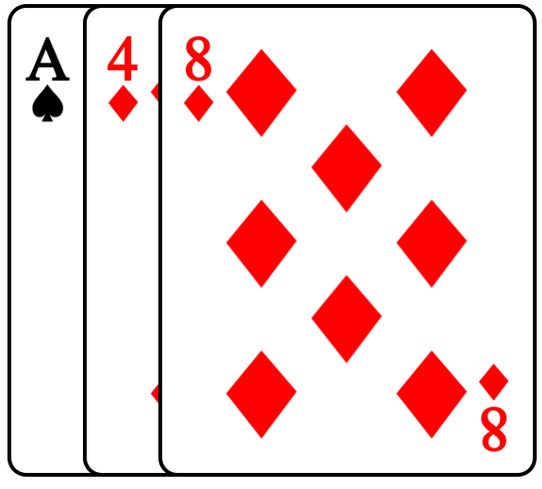


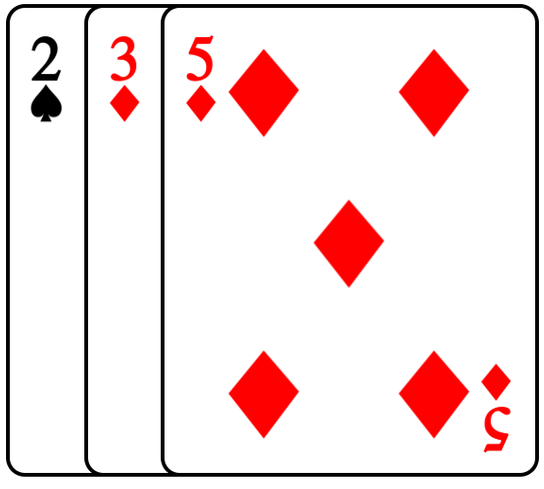


Fusionner

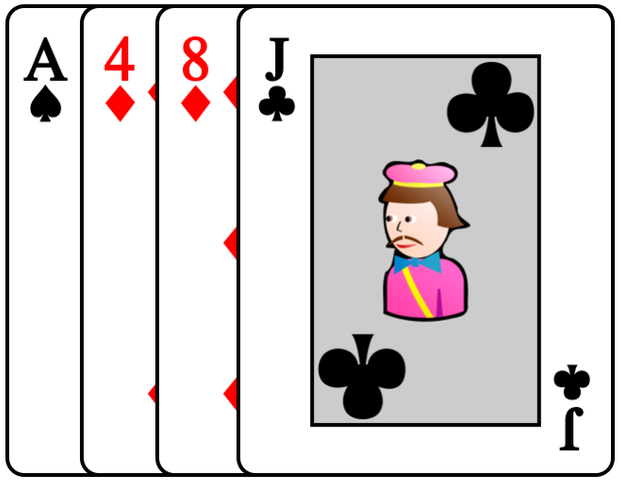


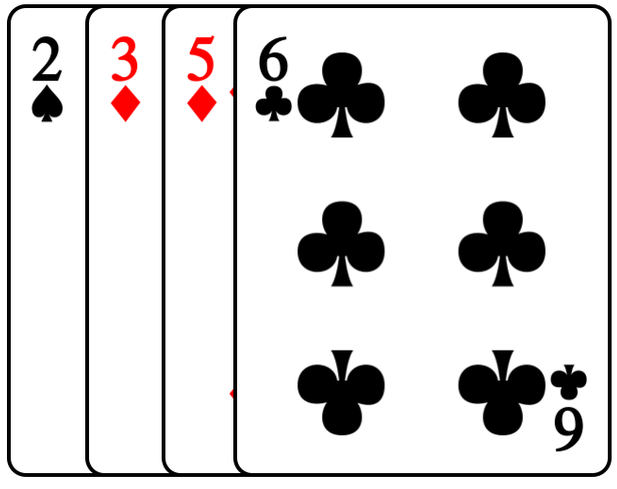




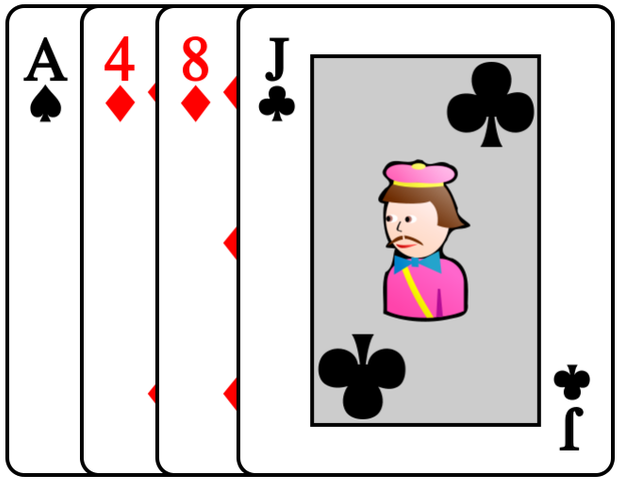


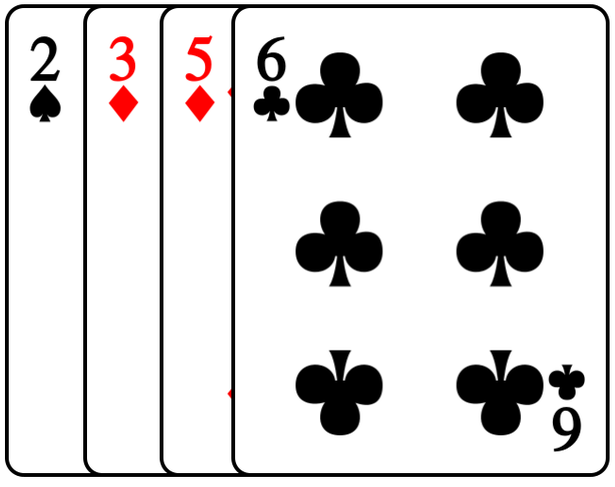
Fusionner



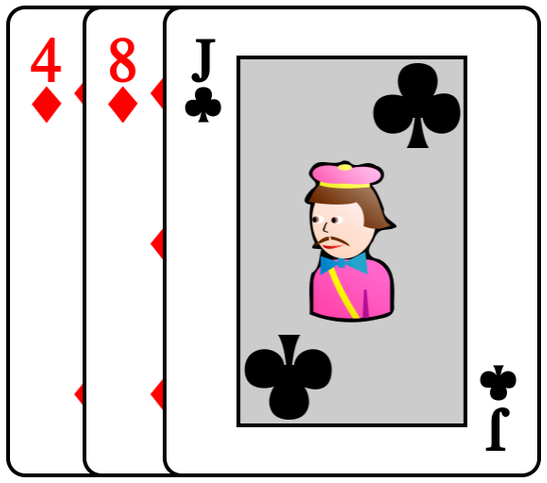


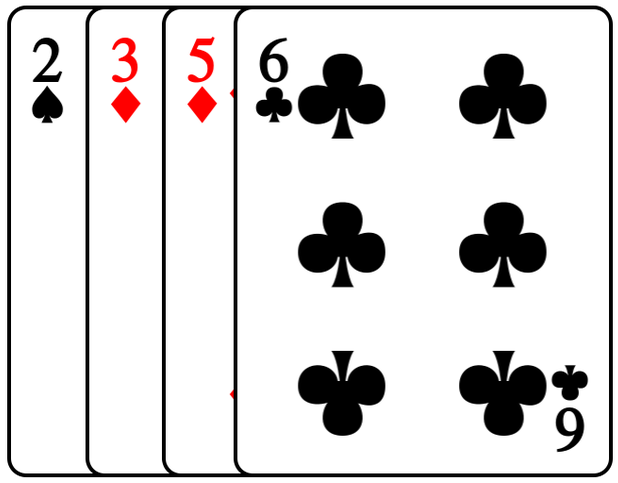
Tous les jeux sont triés !

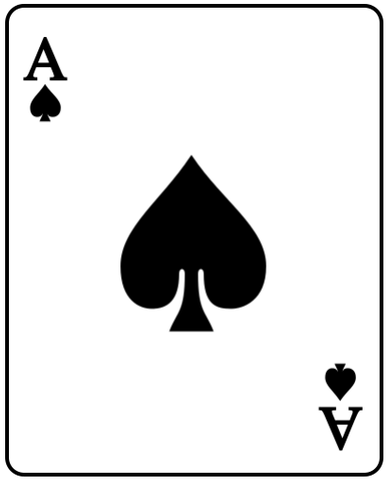




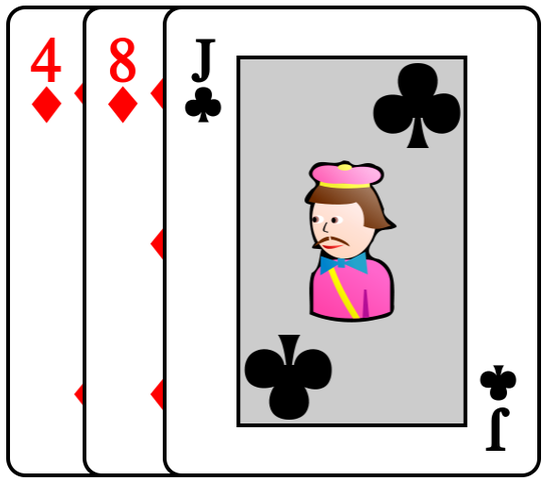
Fusionner

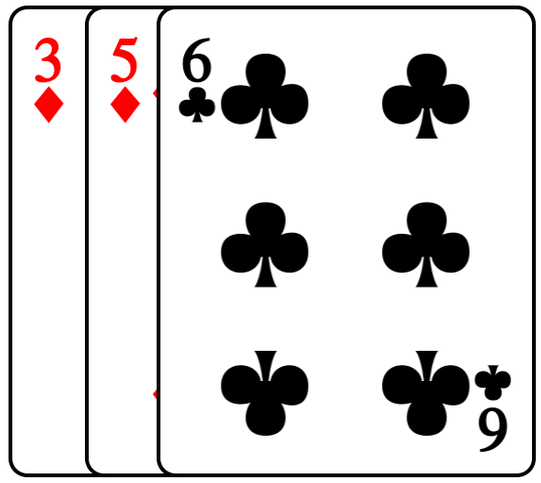


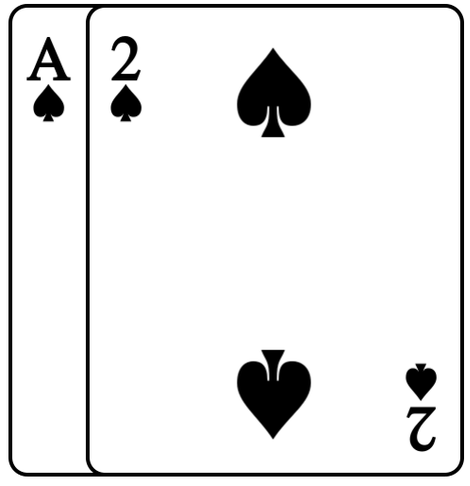




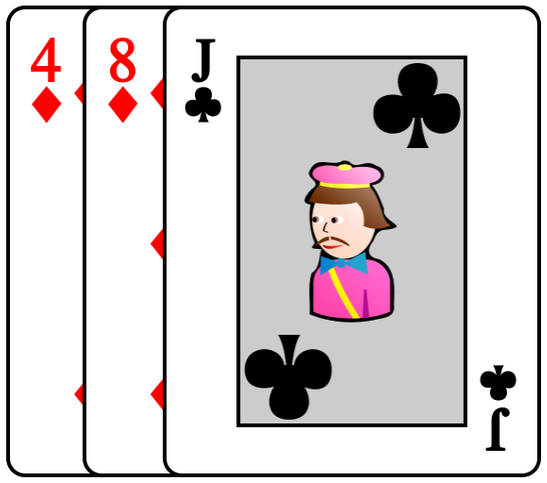
Fusionner

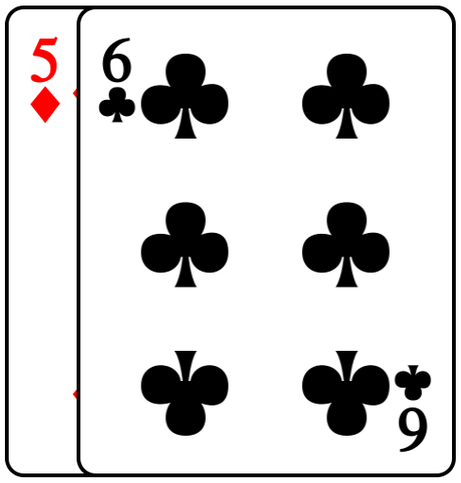


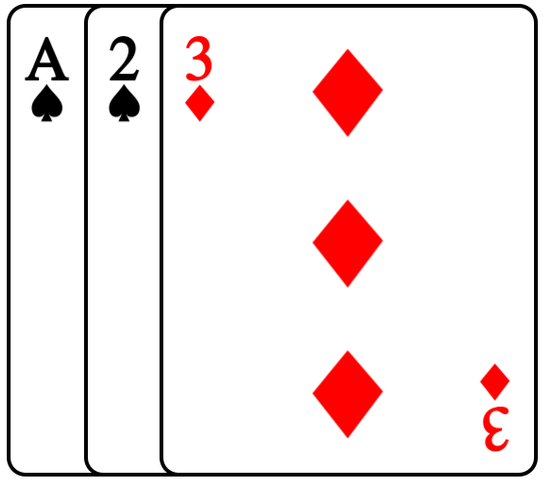




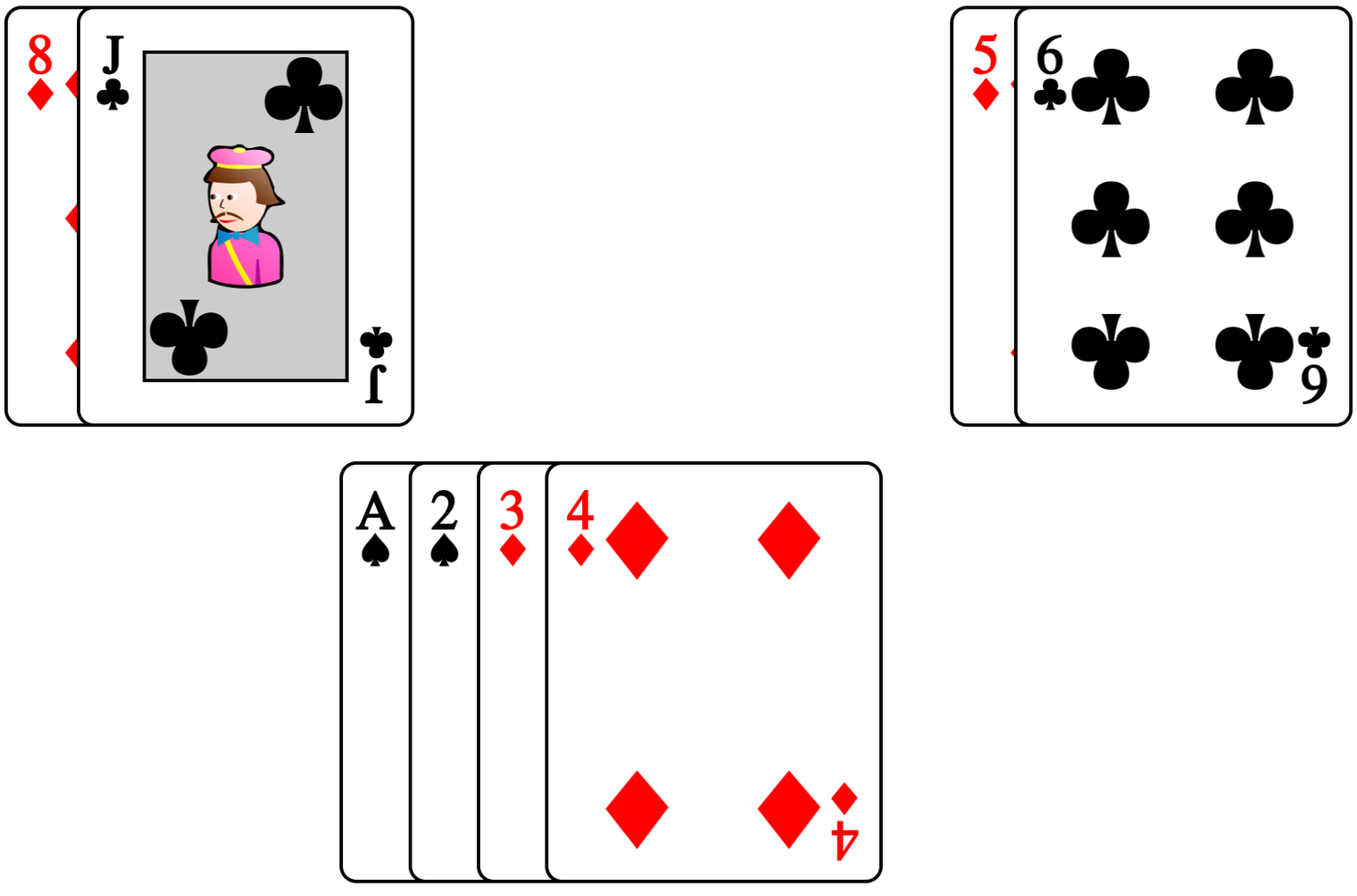
Fusionner



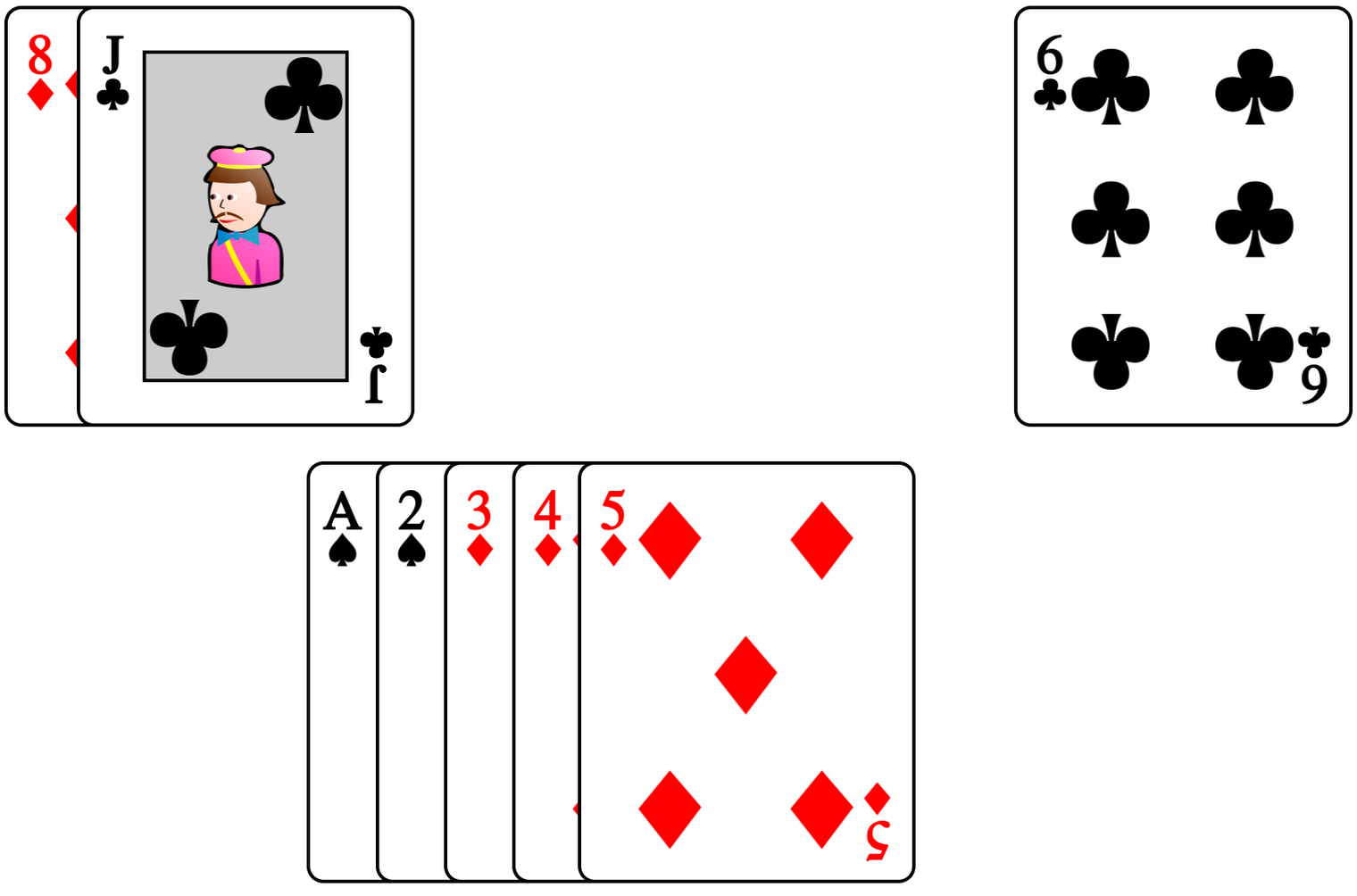




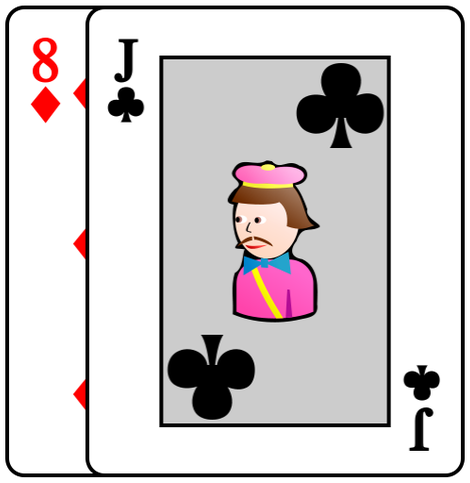
Fusionner

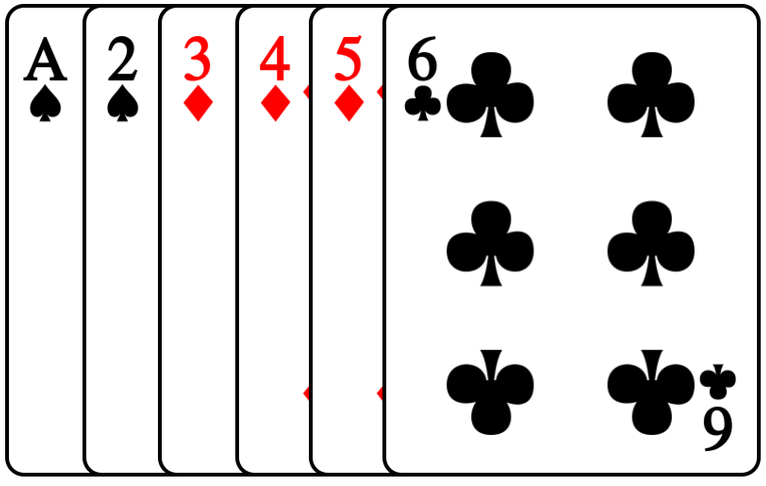


Fusionner

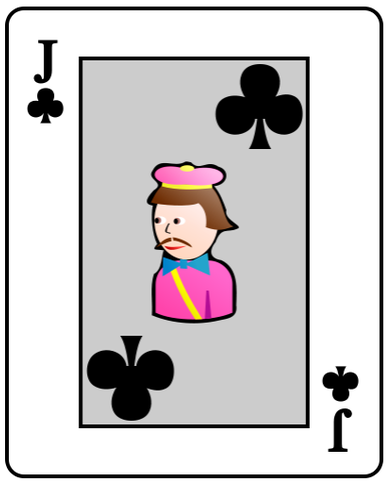


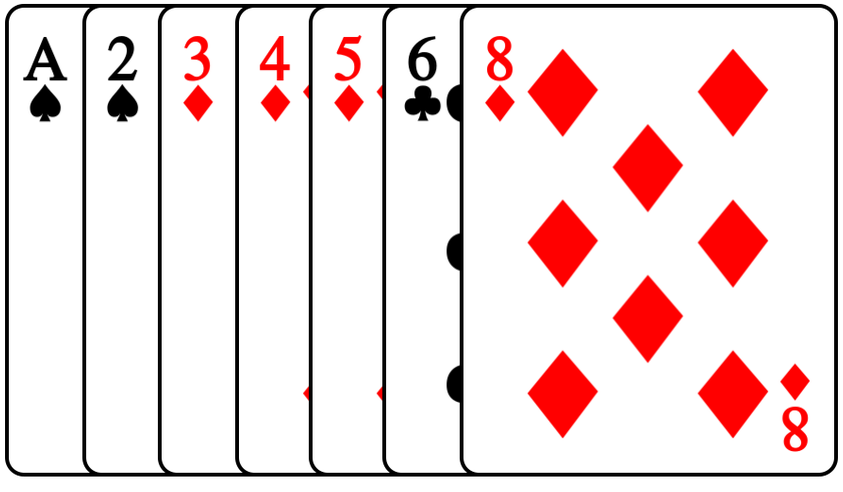
Fusionner



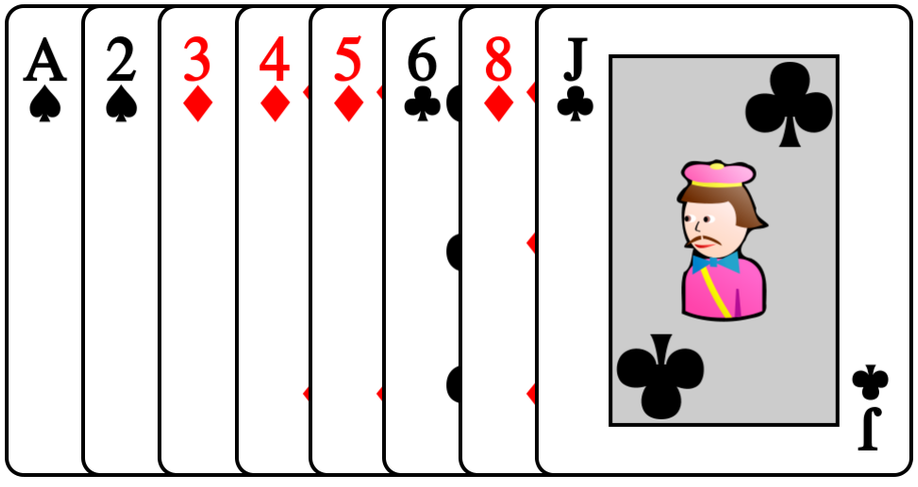


Fusionner

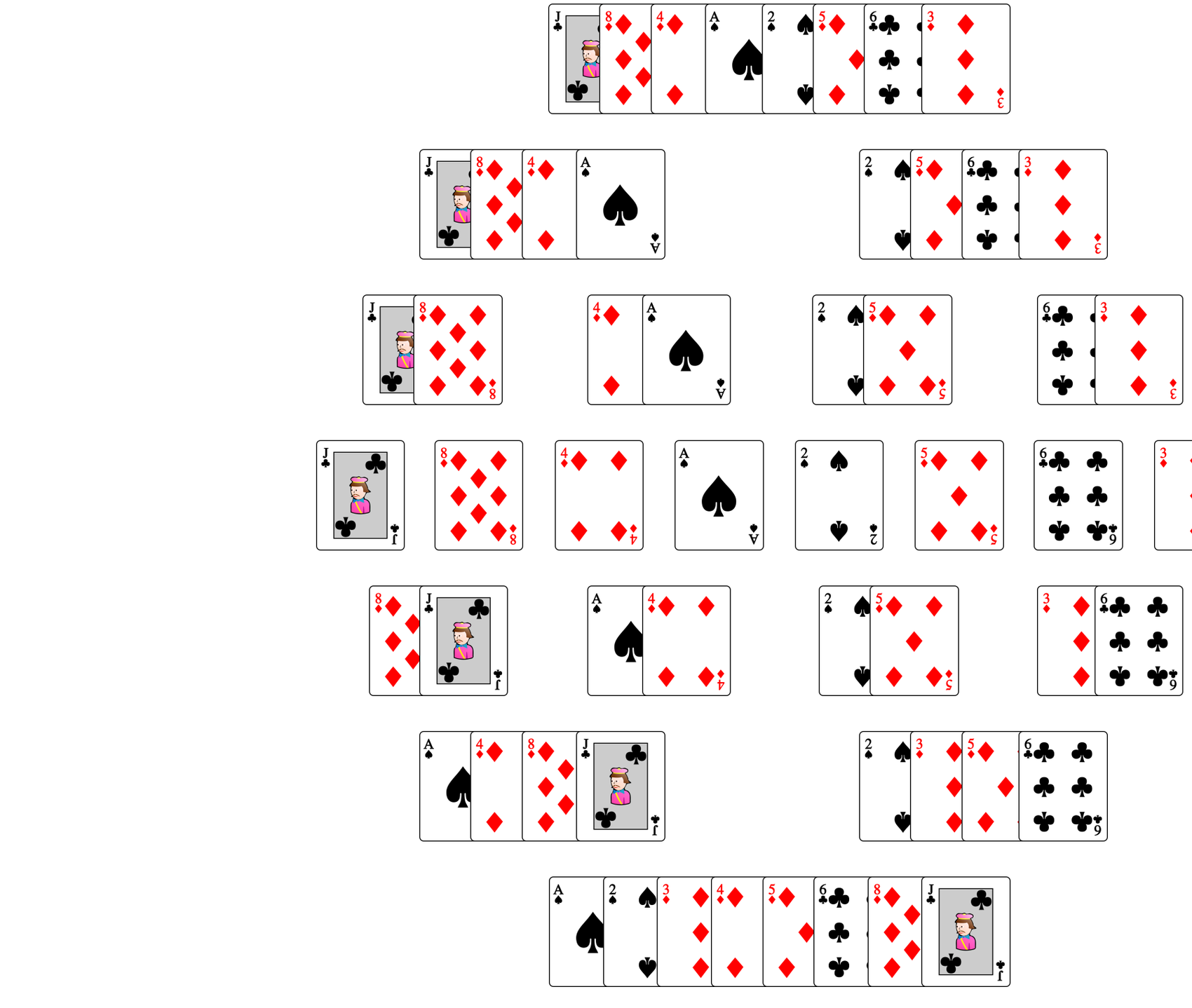




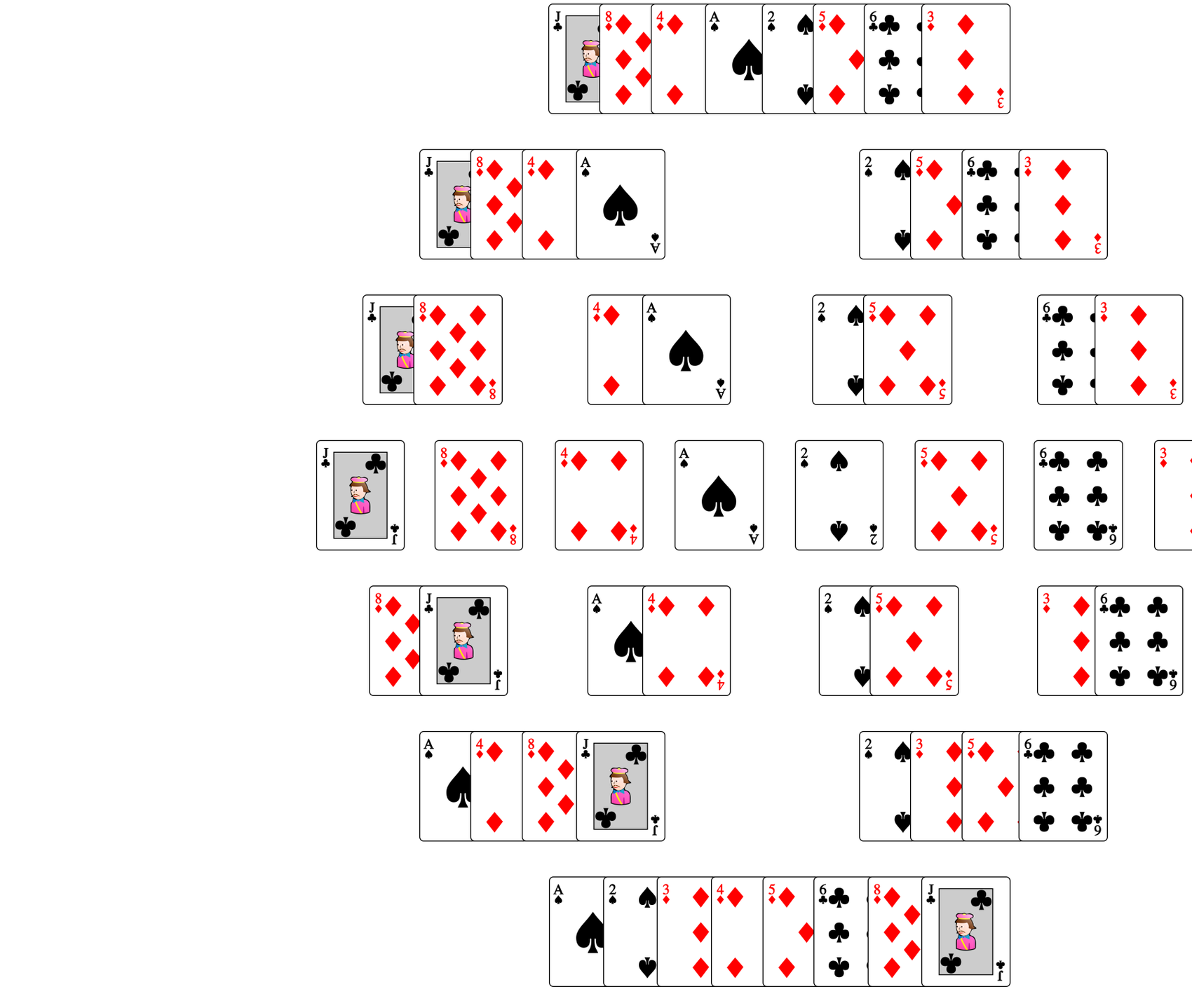
Le jeu est trié !



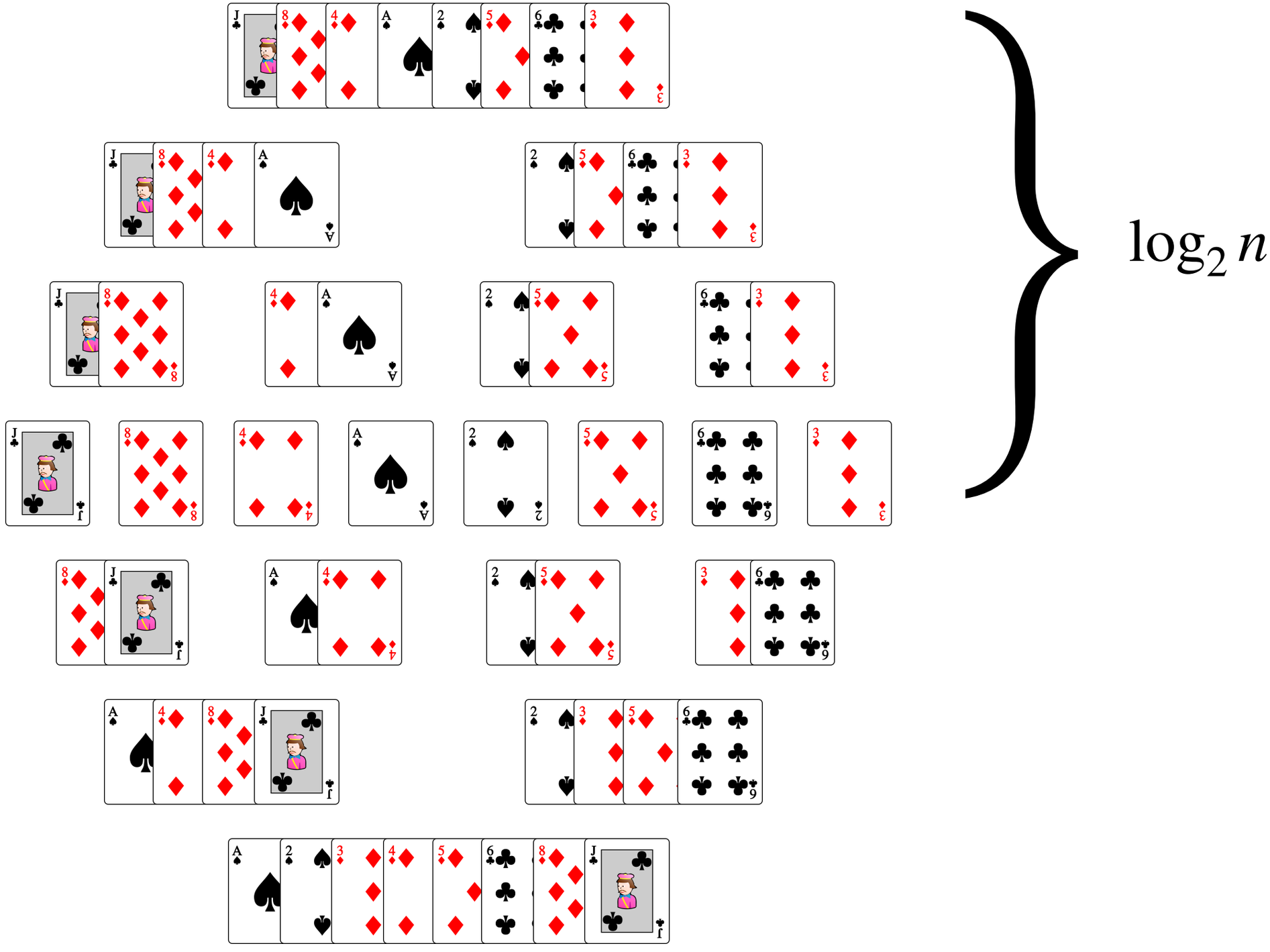
Efficacité du tri fusion



Efficacité du tri fusion

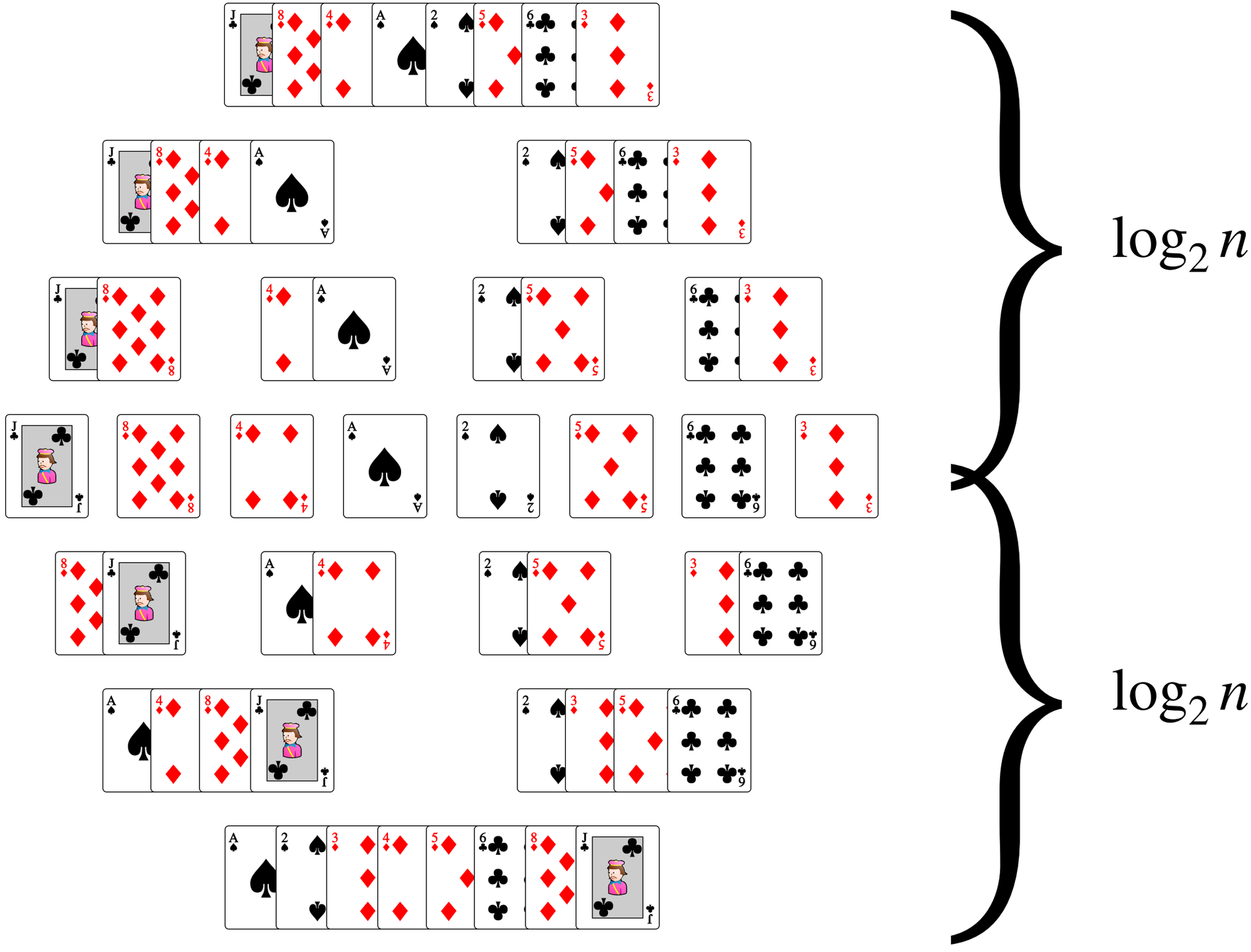


Efficacité du tri fusion



$\log_2 n$

Efficacité du tri fusion

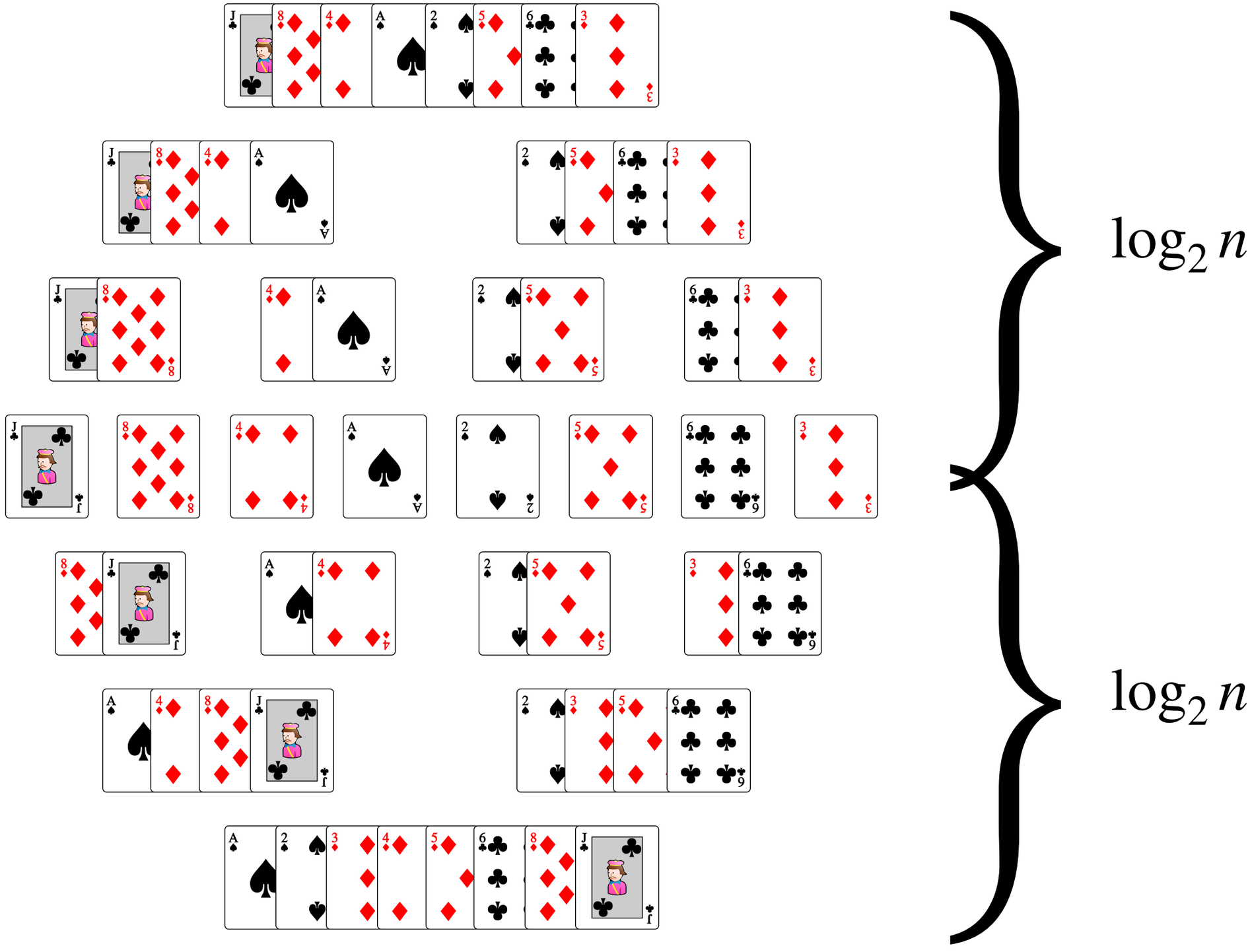


$\log_2 n$

$\log_2 n$

Efficacité du tri fusion

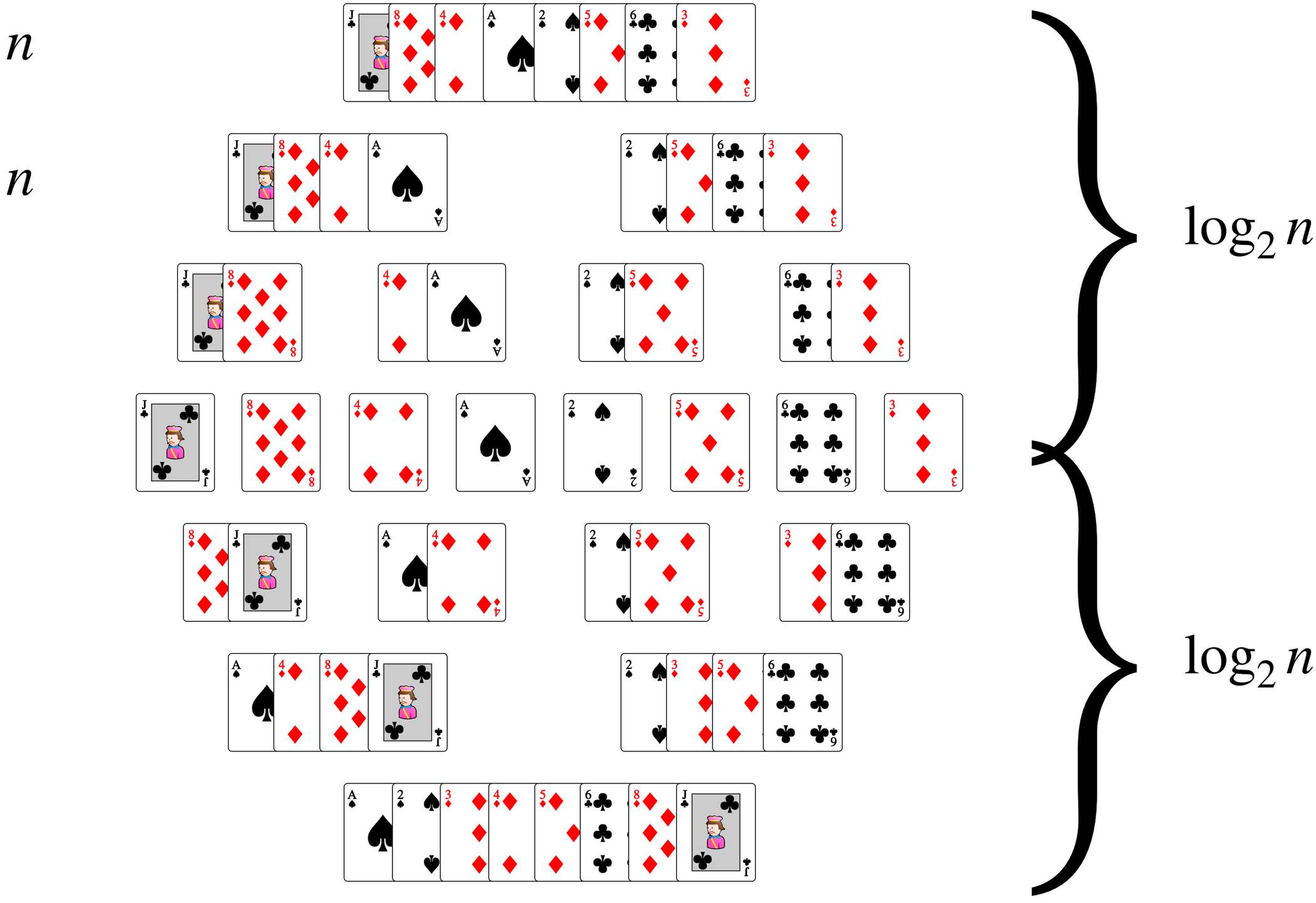
n



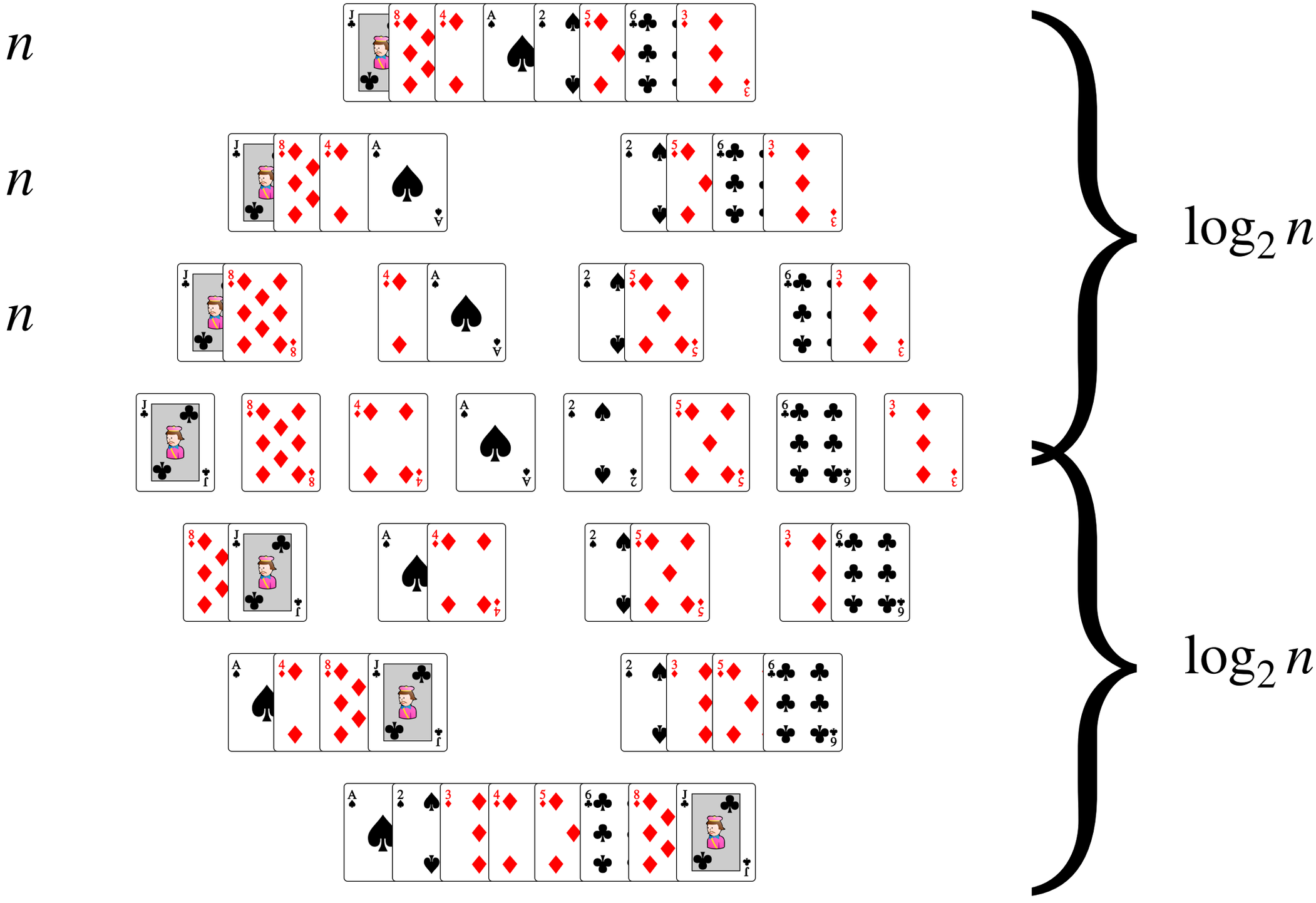
$\log_2 n$

$\log_2 n$

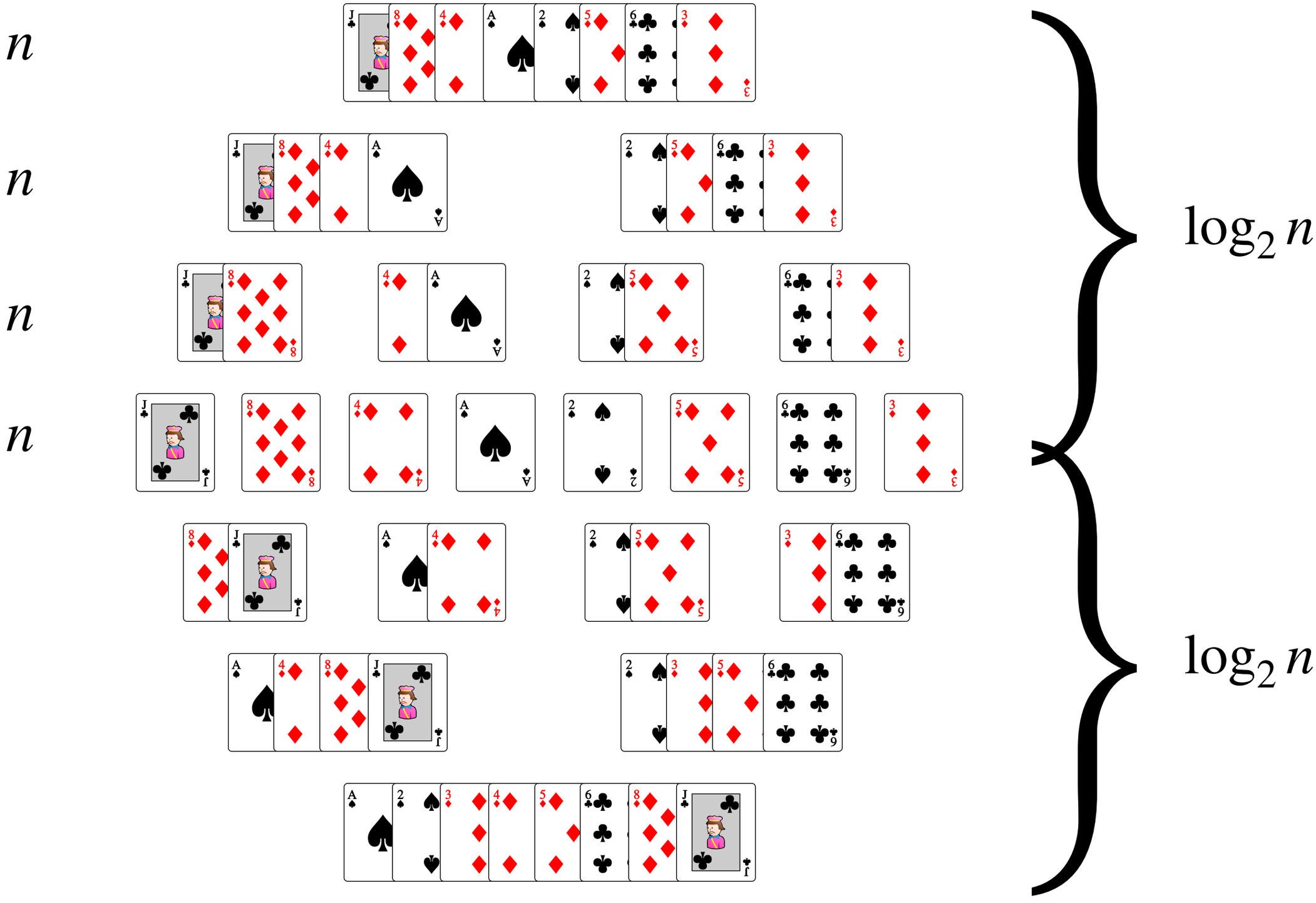
Efficacité du tri fusion



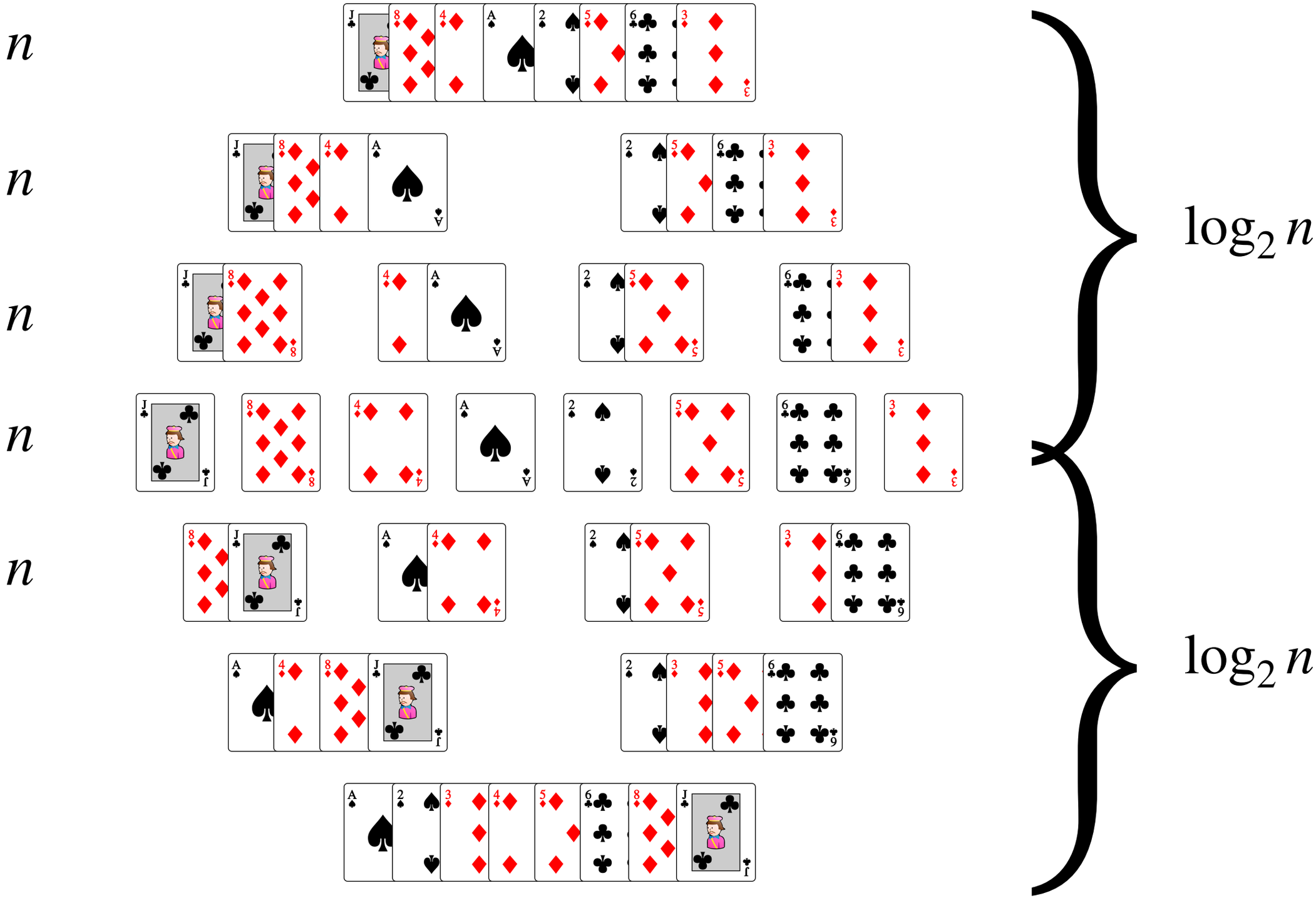
Efficacité du tri fusion



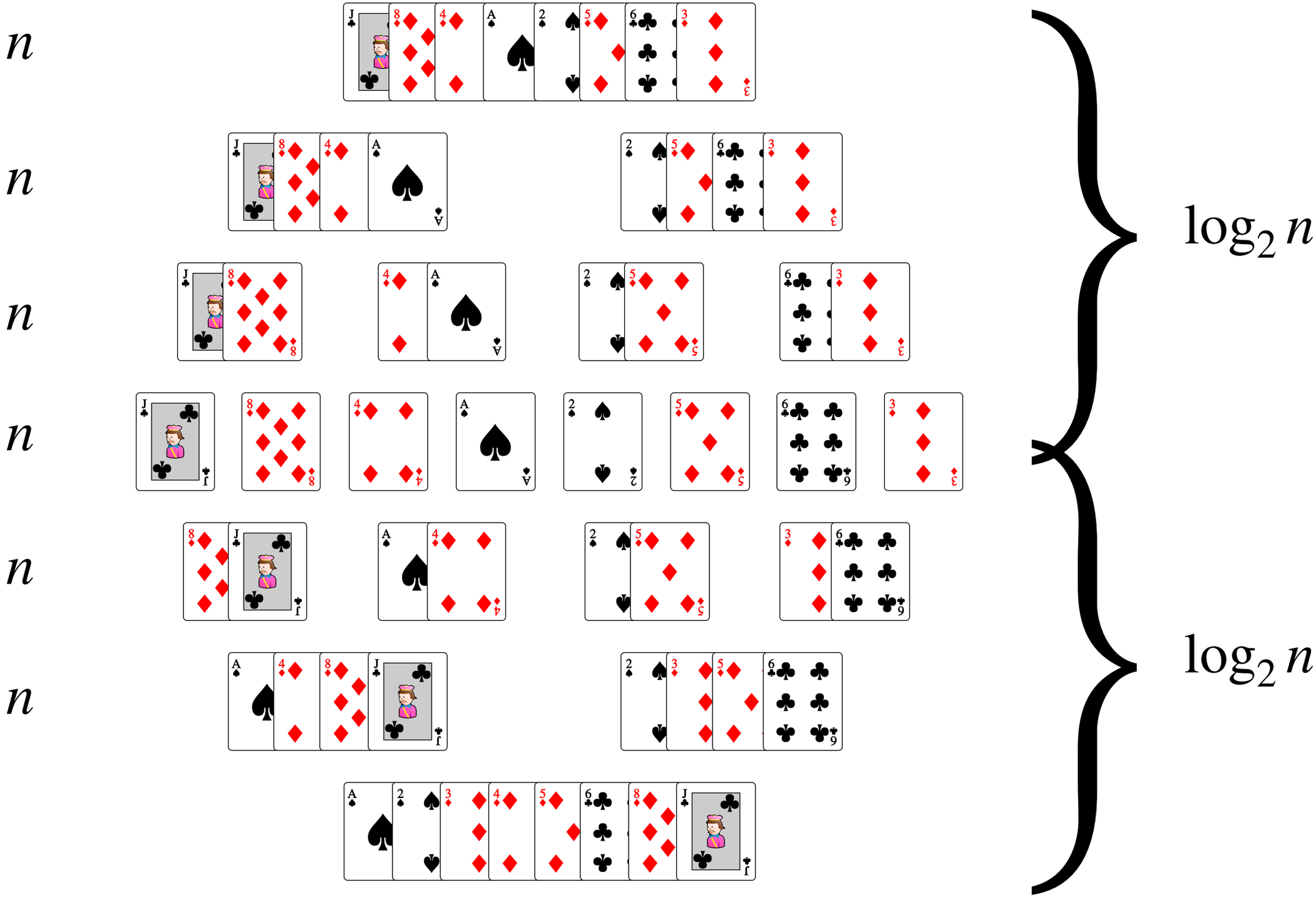
Efficacité du tri fusion



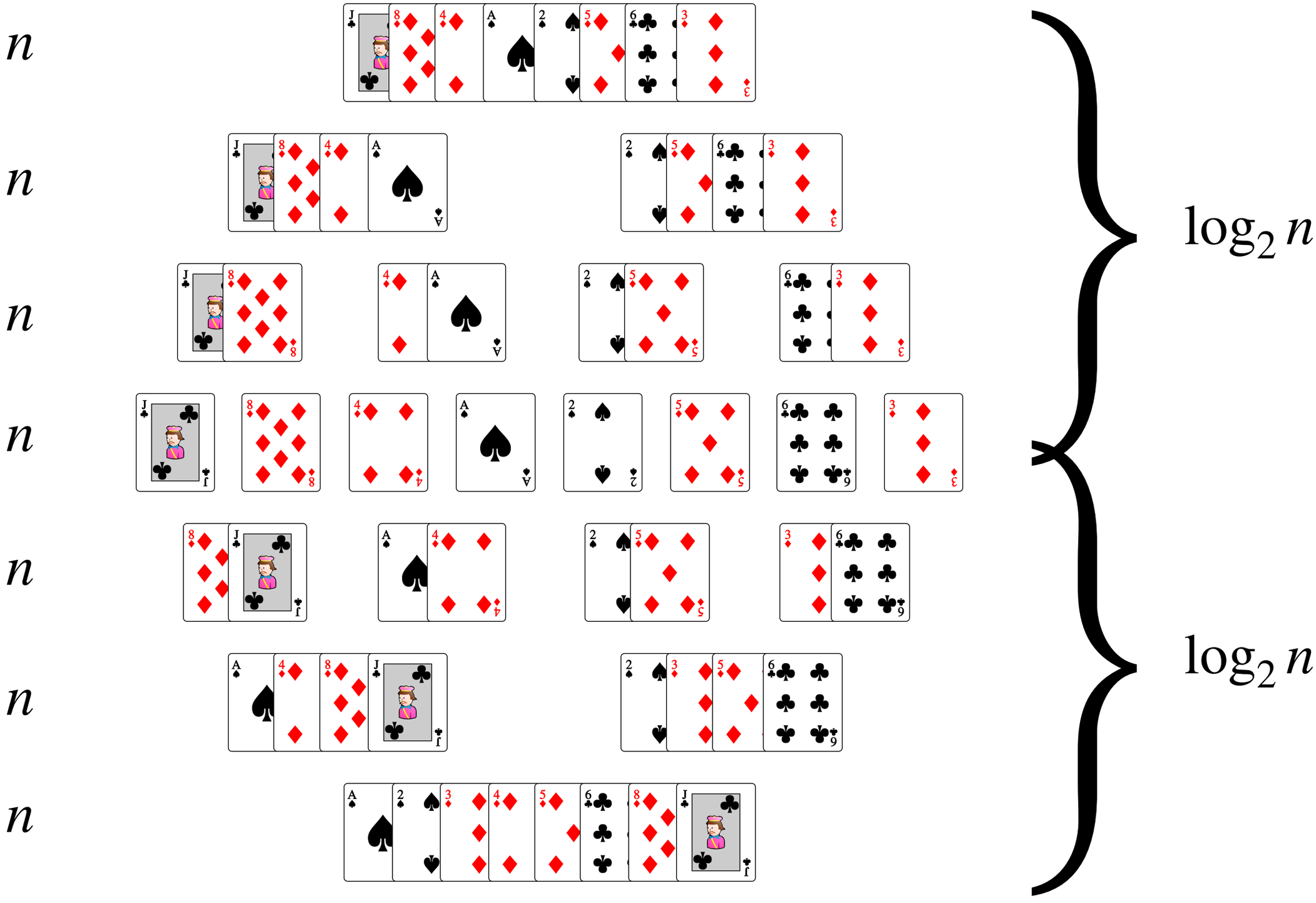
Efficacité du tri fusion



Efficacité du tri fusion



Efficacité du tri fusion



La complexité du tri fusion est

$$O(n \log_2 n)$$