Introduction à l’informatique CM5

Antonio E. Porreca
aeporreca.org/introinfo

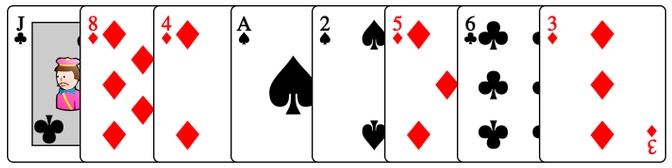
⚠️ Le partiel est le

27 octobre à 13h ⚠️

Tri fusion

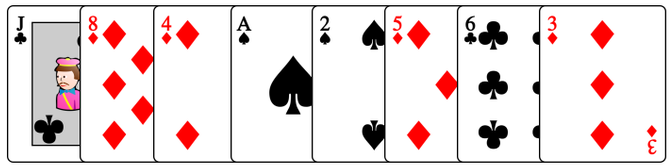
Efficacité du tri fusion

n

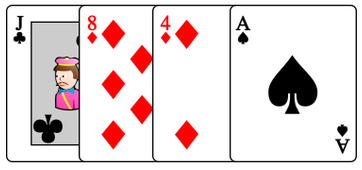


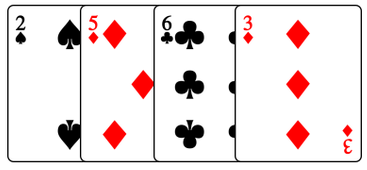
Efficacité du tri fusion

n



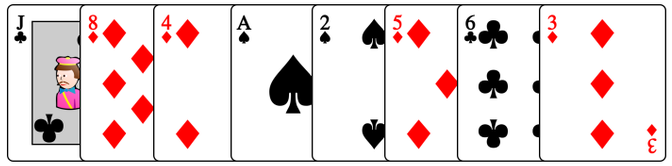
n





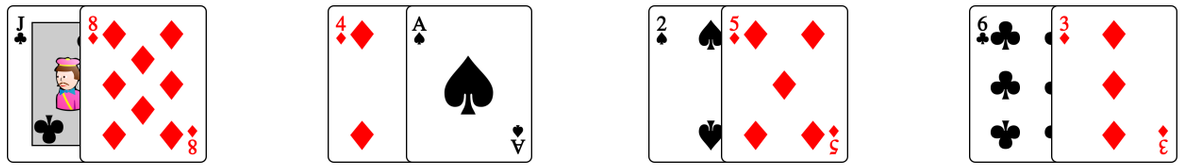
Efficacité du tri fusion

n



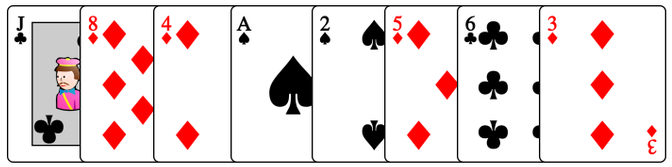
n

n



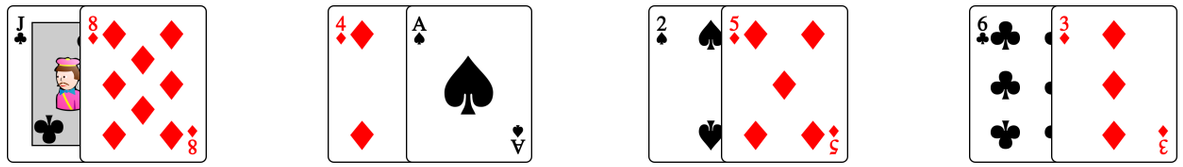
Efficacité du tri fusion

n

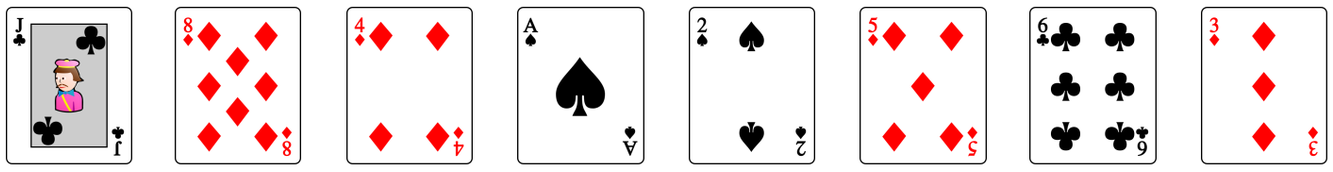


n

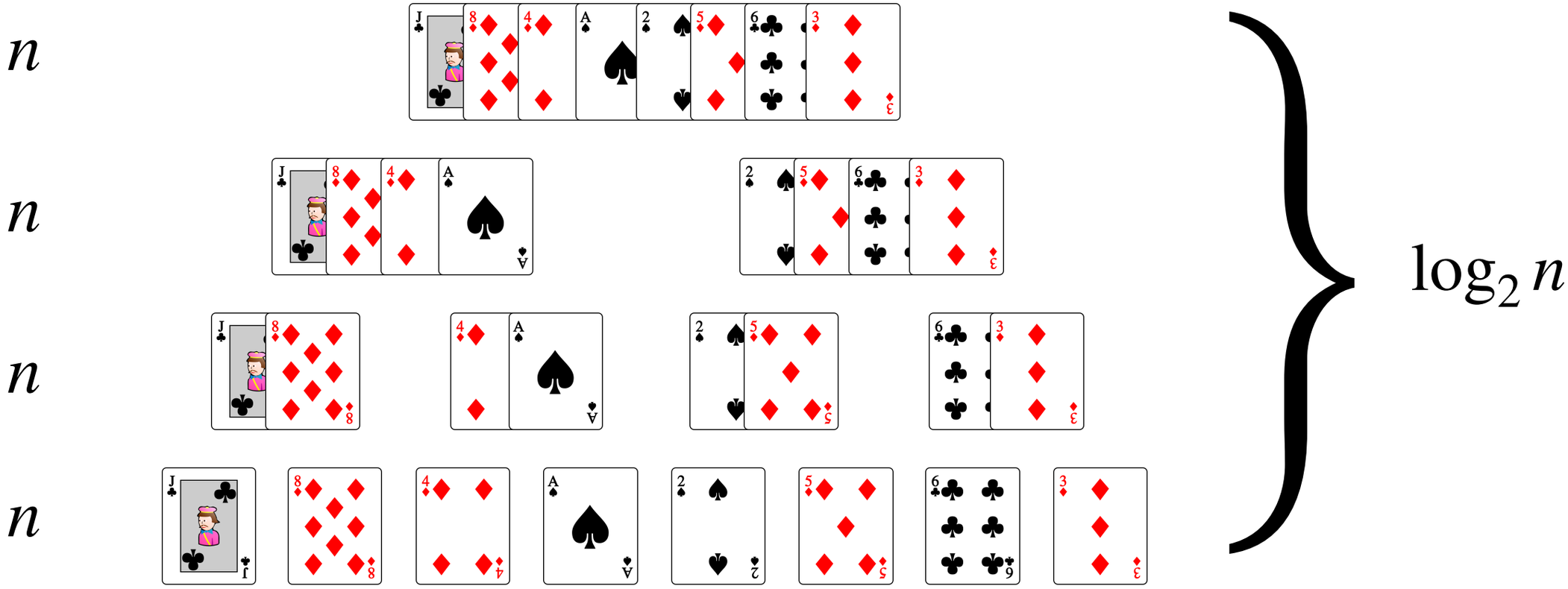
n



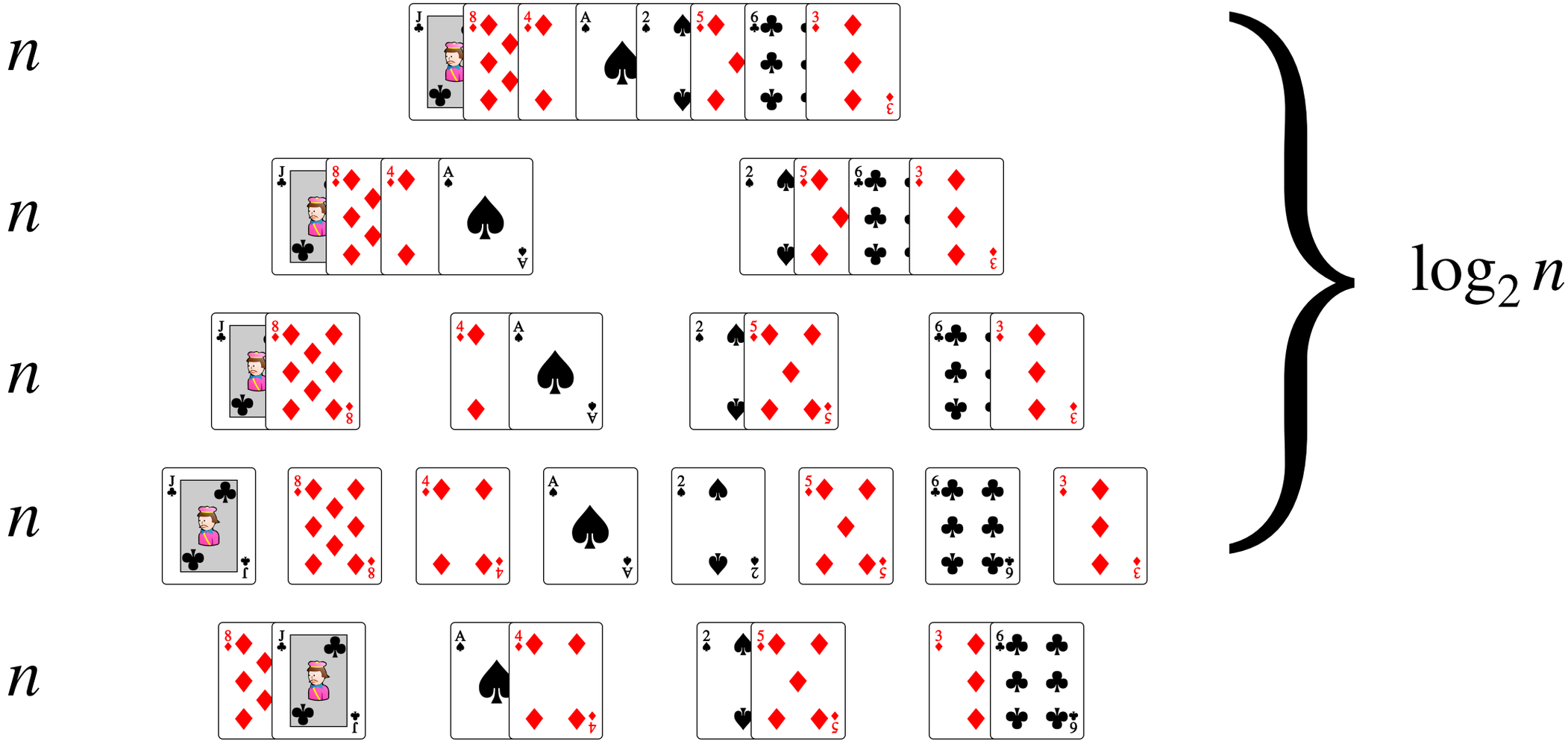
n



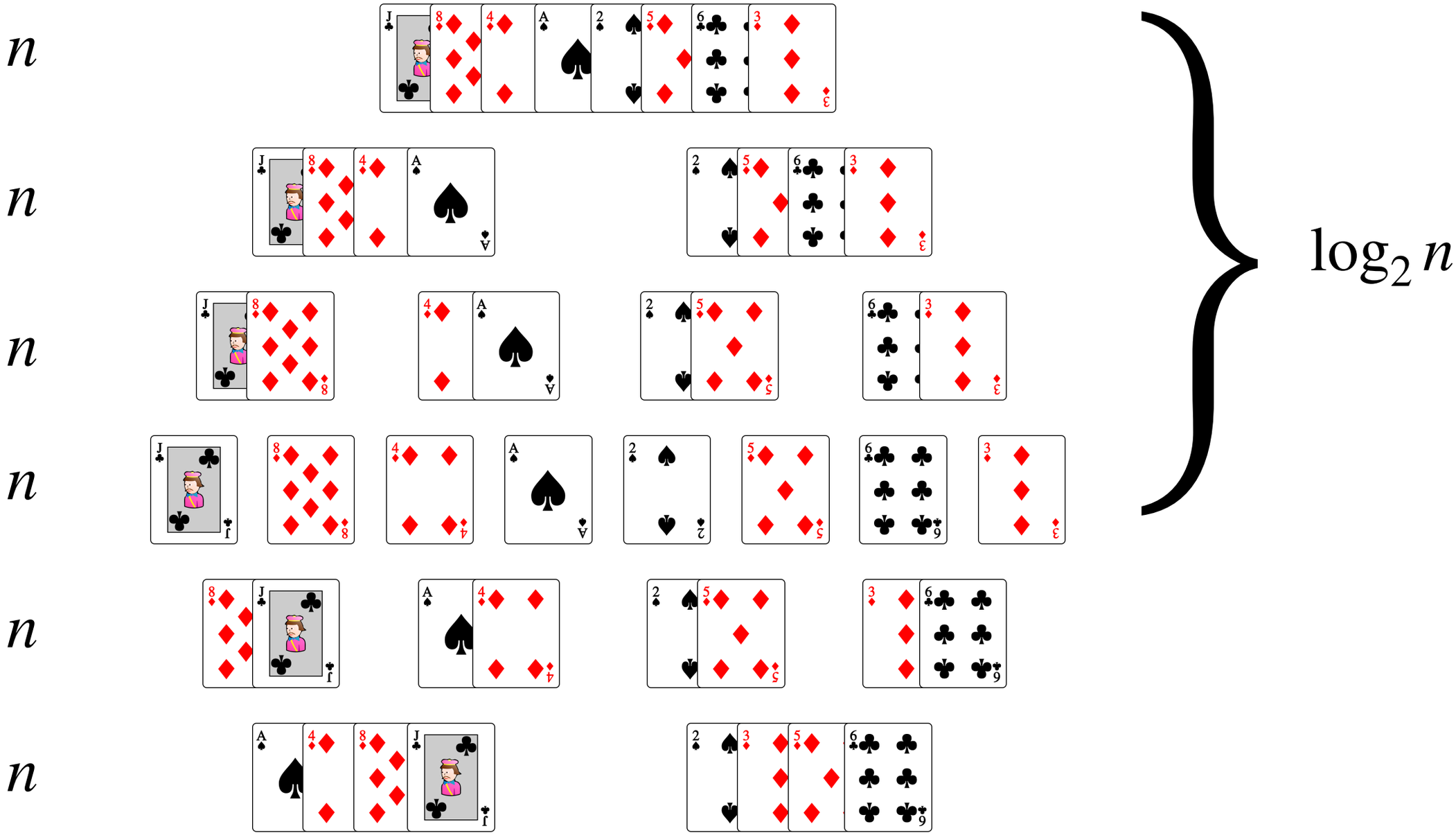
Efficacité du tri fusion



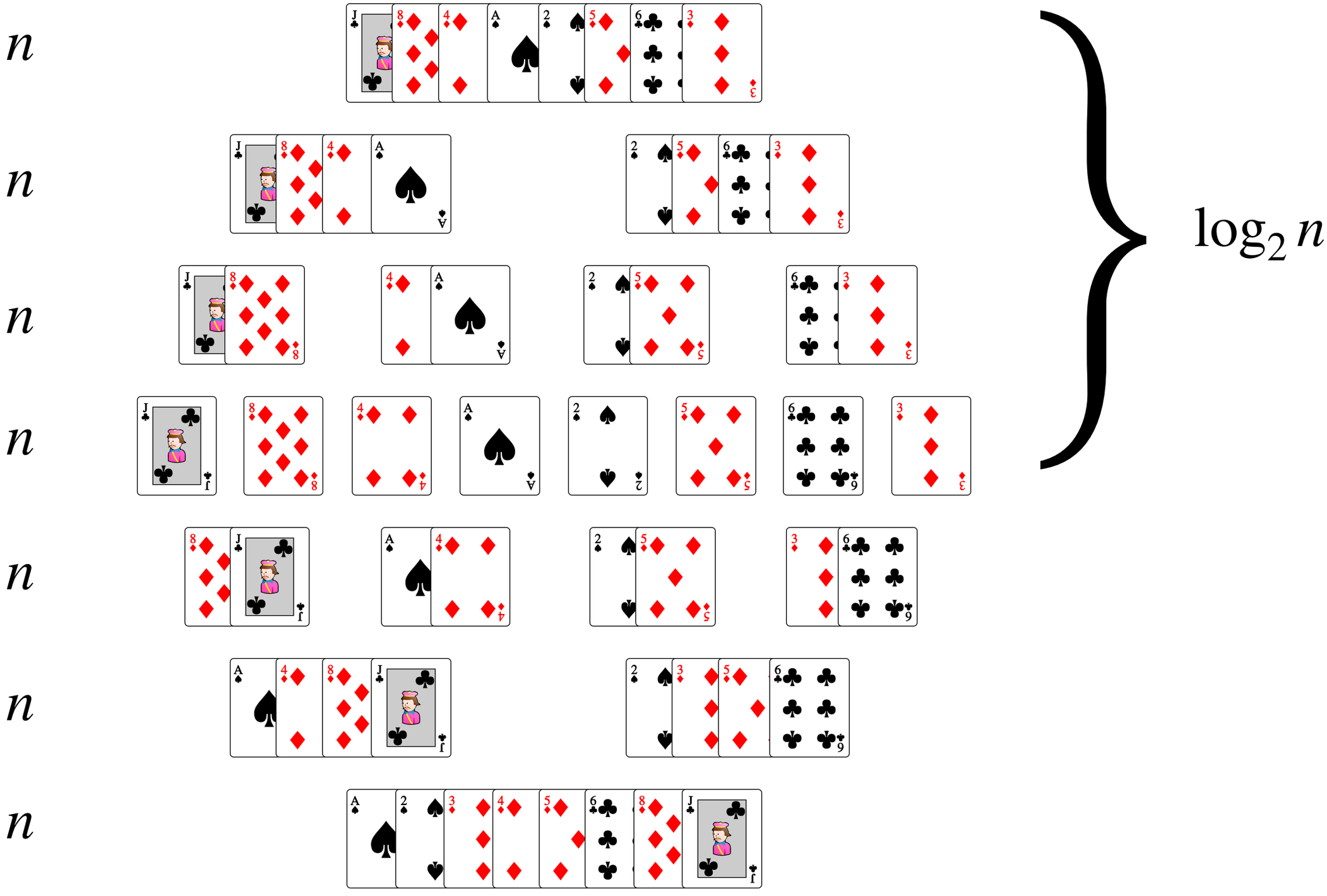
Efficacité du tri fusion



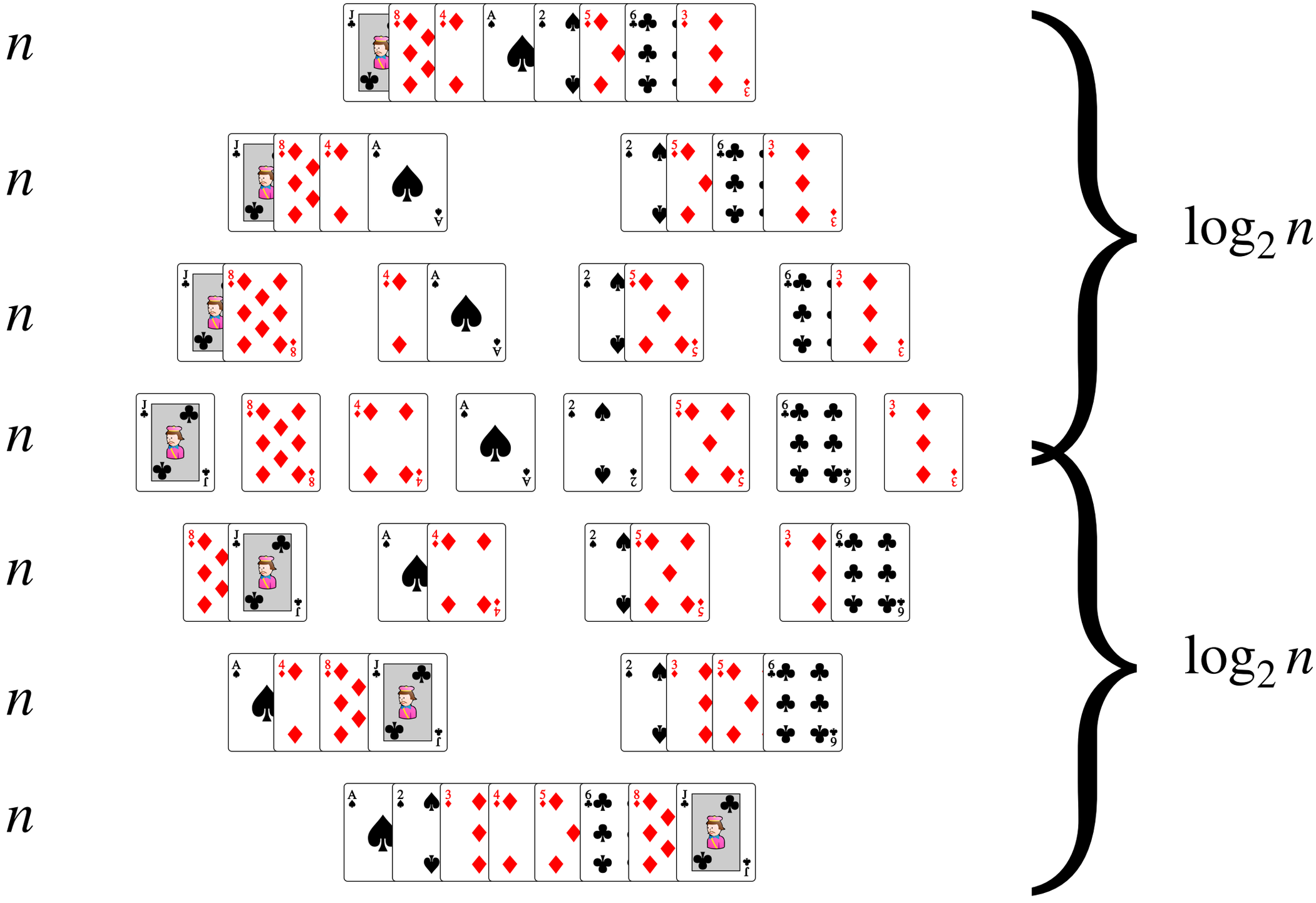
Efficacité du tri fusion



Efficacité du tri fusion



Efficacité du tri fusion



La complexité
du tri fusion est

$$O(n \log_2 n)$$

Fusionner

```
def fusionner(A, B):
    n = len(A)
    m = len(B)
    C = []
    i = j = 0
    while i < n and j < m:
        if A[i] < B[j]:
            C.append(A[i])
            i = i + 1
        else:
            C.append(B[j])
            j = j + 1
    if i < n:
        return C + A[i:n]
    else:
        return C + B[j:m]
```

Tri fusion

```
def tri_fusion(A):
    n = len(A)
    if n > 1:
        m = n // 2
        B = tri_fusion(A[0:m])
        C = tri_fusion(A[m:n])
        return fusionner(B, C)
    else:
        return A
```

Algorithmes sur les entiers

Incrémentation

1	2	1	3	9	9	9	9

Incrémentation

1	2	1	3	9	9	9	9

+1

Incrémentation

1	2	1	3	9	9	9	9

+1

Incrémentation

1	2	1	3	9	9	9	9

+1

							0

Incrémentation

1	2	1	3	9	9	9	9

+1

							0

Incrémentation

1	2	1	3	9	9	9	9

+1

						0	0

Incrémentation

1	2	1	3	9	9	9	9

+1

						0	0

Incrémentation

1	2	1	3	9	9	9	9

+1

					0	0	0

Incrémentation

1	2	1	3	9	9	9	9

+1

					0	0	0

Incrémentation

1	2	1	3	9	9	9	9

+1

				0	0	0	0

Incrémentation

1	2	1	3	9	9	9	9

+1

				0	0	0	0

Incrémentation

1	2	1	3	9	9	9	9

+1

			4	0	0	0	0

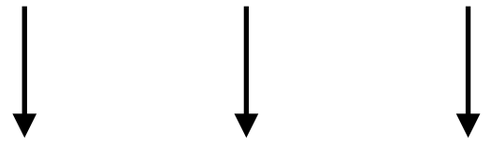
Incrémentation

1	2	1	3	9	9	9	9

			4	0	0	0	0

Incrémentation

1	2	1	3	9	9	9	9



1	2	1	4	0	0	0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

							0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

						0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

						0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

					0	0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

					0	0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

				0	0	0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

				0	0	0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

			0	0	0	0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

			0	0	0	0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

		0	0	0	0	0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

		0	0	0	0	0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

	0	0	0	0	0	0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

	0	0	0	0	0	0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

0	0	0	0	0	0	0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

+1

0	0	0	0	0	0	0	0

Dépassement d'entier (overflow)

9	9	9	9	9	9	9	9

0	0	0	0	0	0	0	0

Algo pour l'incrémentation

```
def incrémenter(N):
    n = len(N)
    R = [0] * n          # tableau de long. n
    i = n - 1
    while i >= 0 and N[i] == 9:
        R[i] = 0
        i = i - 1
    if i >= 0:
        R[i] = N[i] + 1
        i = i - 1
    while i >= 0:
        R[i] = N[i]
        i = i - 1
    return R
```

Addition

9	3	2	3	6	4	5	3	+
0	6	8	1	6	5	9	7	=

Addition

						+1		
9	3	2	3	6	4	5	3	+
0	6	8	1	6	5	9	7	=
							0	

Addition

+1

9	3	2	3	6	4	5	3	+
0	6	8	1	6	5	9	7	=
						5	0	

Addition

+1

9	3	2	3	6	4	5	3	+
0	6	8	1	6	5	9	7	=
					0	5	0	

Addition

+1

9	3	2	3	6	4	5	3	+
0	6	8	1	6	5	9	7	=
				3	0	5	0	

Addition

9	3	2	3	6	4	5	3	+
0	6	8	1	6	5	9	7	=
			5	3	0	5	0	

Addition

	+1							
9	3	2	3	6	4	5	3	+
0	6	8	1	6	5	9	7	=
		0	5	3	0	5	0	

Addition

+1								
9	3	2	3	6	4	5	3	+
0	6	8	1	6	5	9	7	=
	0	0	5	3	0	5	0	

Addition

+1

	9	3	2	3	6	4	5	3	+
0	6	8	1	6	5	9	7	=
0	0	0	5	3	0	5	0	

Addition

9	3	2	3	6	4	5	3	+
0	6	8	1	6	5	9	7	=
0	0	0	5	3	0	5	0	

Algo pour l'addition

```
def additionner(M, N):
    n = len(M)
    R = [0] * n
    retenue = 0
    for i in reversed(range(n)):
        R[i] = (M[i] + N[i] + retenue) % 10
        retenue = (M[i] + N[i] + retenue) // 10
    return R
```

Division euclidienne de a par b

Division euclidienne de a par b

$$a = q \times b + r \qquad \textbf{avec } 0 \leq r < b$$

Division euclidienne de *a* par *b*

$$a = q \times b + r$$

quotient reste

avec $0 \leq r < b$

Division euclidienne

```
def division_euclidienne(a, b):
    q = 0
    r = a
    while r >= b:
        q = q + 1
        r = r - b
    return (q, r)
```

Plus grand commun diviseur

Plus grand commun diviseur de 21 et 14

$$21 = 14 \times 1 + 7$$

Plus grand commun diviseur de 21 et 14

$$21 = 14 \times 1 + 7$$

$$14 = 7 \times 2 + 0$$

Plus grand commun diviseur de 21 et 14

$$21 = 14 \times 1 + 7$$

$$14 = 7 \times 2 + 0$$

$$\text{pgcd}(21, 14) = 7$$

Plus grand commun diviseur de 799 et 345

$$799 = 345 \times 2 + 109$$

Plus grand commun diviseur de 799 et 345

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

Plus grand commun diviseur de 799 et 345

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

$$109 = 18 \times 6 + 1$$

Plus grand commun diviseur de 799 et 345

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

$$109 = 18 \times 6 + 1$$

$$18 = 1 \times 18 + 0$$

Plus grand commun diviseur de 799 et 345

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

$$109 = 18 \times 6 + 1$$

$$18 = 1 \times 18 + 0$$

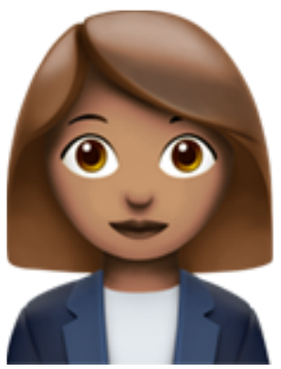
$$\text{pgcd}(799, 345) = 1$$

Algorithme d'Euclide

```
def pgcd(a, b):
    # a et b entiers != 0
    r = a % b
    while r > 0:
        a = b
        b = r
        r = a % b
    return b
```

Ça sert à quoi ?

Communications sécurisées



Alice

Bob

Communications sécurisées

Alice

Bob

Communications sécurisées

Alice

Bob

Eve

Communications sécurisées

Alice

Bob

Eve

Communications sécurisées

Alice

Bob

Eve

Communications sécurisées

Alice

Bob

Eve

Communications sécurisées

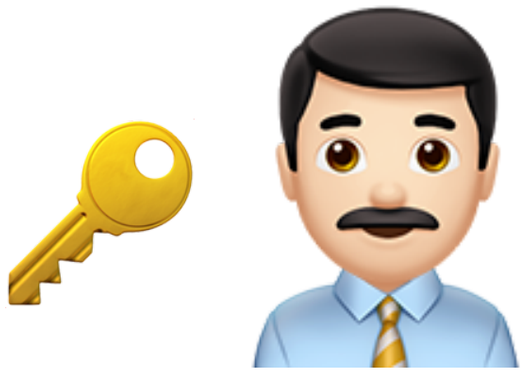
Alice

Bob

Eve

Communications sécurisées

Alice



Bob

Eve

Communications sécurisées

Alice

Bob

Eve

Communications sécurisées

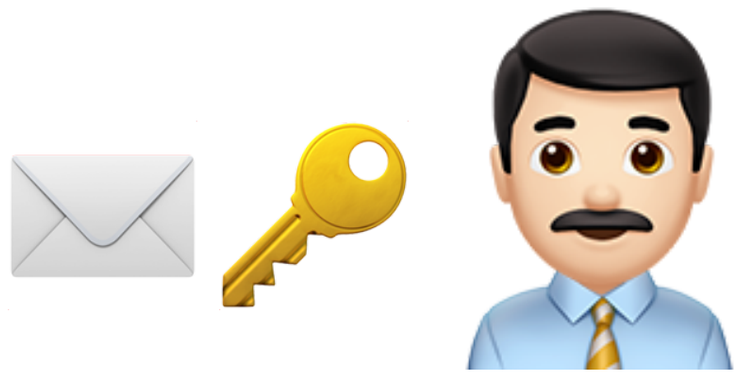
Alice

Bob

Eve

Communications sécurisées

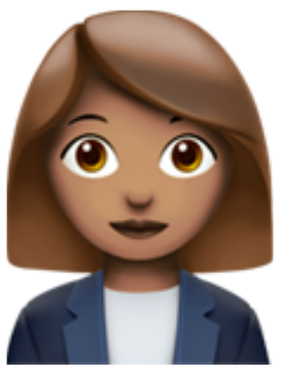
Alice



Bob

Eve

Cryptosystème RSA



🔑 (e, n)

(d, n) 🔑

Alice

Bob

✉️ $\in \{0, \ldots, n-1\}$

Chiffrement et déchiffrement RSA

Alice

(e, n)

$$M \in \{0, \ldots, n-1\}$$

$$C = M^e \bmod n$$

Bob

(d, n)

$$C^d \bmod n = M$$

Calculer les puissances

$$x \quad x^2 \quad x^3 \quad \dots \quad x^n$$

Calculer les puissances

$$x \quad x^2 \quad x^3 \quad \dots \quad x^n$$

```
def puissance(x, n):
    p = 1
    for i in range(n):
        p = p * x
    return p
```

Calculer les puissances

$$x \quad x^2 \quad x^3 \quad \dots \quad x^n$$

```
def puissance(x, n):
    p = 1
    for i in range(n):
        p = p * x
    return p
```

n multiplications

Exponentiation rapide

$$x^{16}$$

Exponentiation rapide

$$x^{16} = (x^8)^2$$

Exponentiation rapide

$$x^{16} = (x^8)^2$$

$$= ((x^4)^2)^2$$

Exponentiation rapide

$$x^{16} = (x^8)^2$$

$$= ((x^4)^2)^2$$

$$= (((x^2)^2)^2)^2$$

Exponentiation rapide

$$x^{16} = (x^8)^2$$

$$= ((x^4)^2)^2$$

$$= (((x^2)^2)^2)^2$$

4 multiplications

Exponentiation rapide

$$x^{13}$$

Exponentiation rapide

$$x^{13} = (x^6)^2 \times x$$

Exponentiation rapide

$$x^{13} = (x^6)^2 \times x$$
$$= ((x \times x \times x)^2)^2 \times x$$

Exponentiation rapide

$$x^{13} = (x^6)^2 \times x$$

$$= ((x \times x \times x)^2)^2 \times x$$

5 multiplications

Exponentiation rapide

```
def puissance_rapide(x, n):
    a = 1
    b = x
    m = n
    while m > 0:
        if m % 2 == 0:
            m = m // 2
        else:
            m = (m - 1) // 2
            a = a * b
        b = b * b
    return a
```