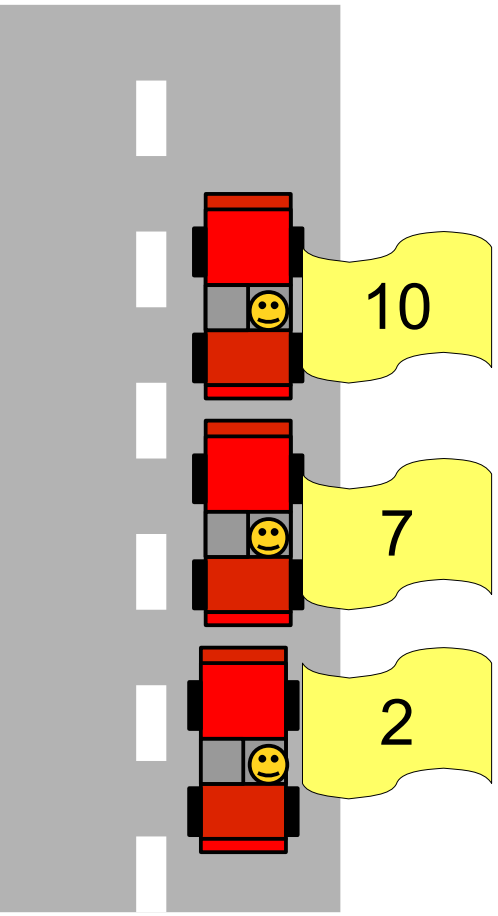
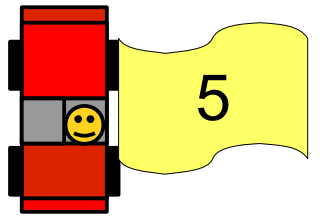
# Coût amorti et Tas de Fibonacci

Chapitre 17et 19, Cormen

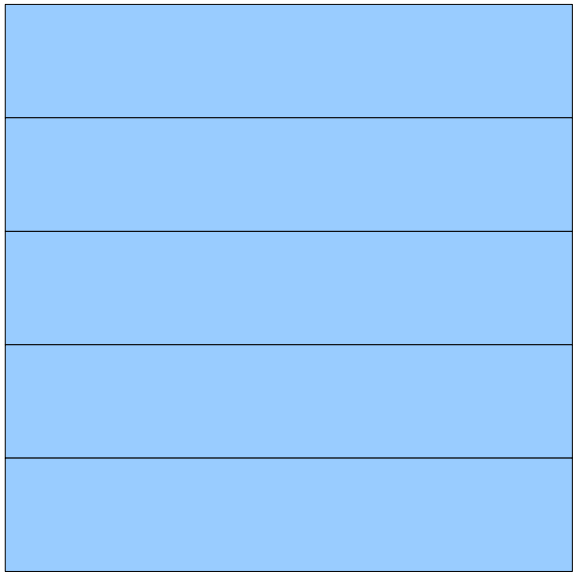
Fredman, M. L.; Tarjan (1987). "Fibonacci heaps and their uses in improved network optimization algorithms". Journal of the ACM 34 (3): 596–615

# Structure de données abstraite « File de priorité »



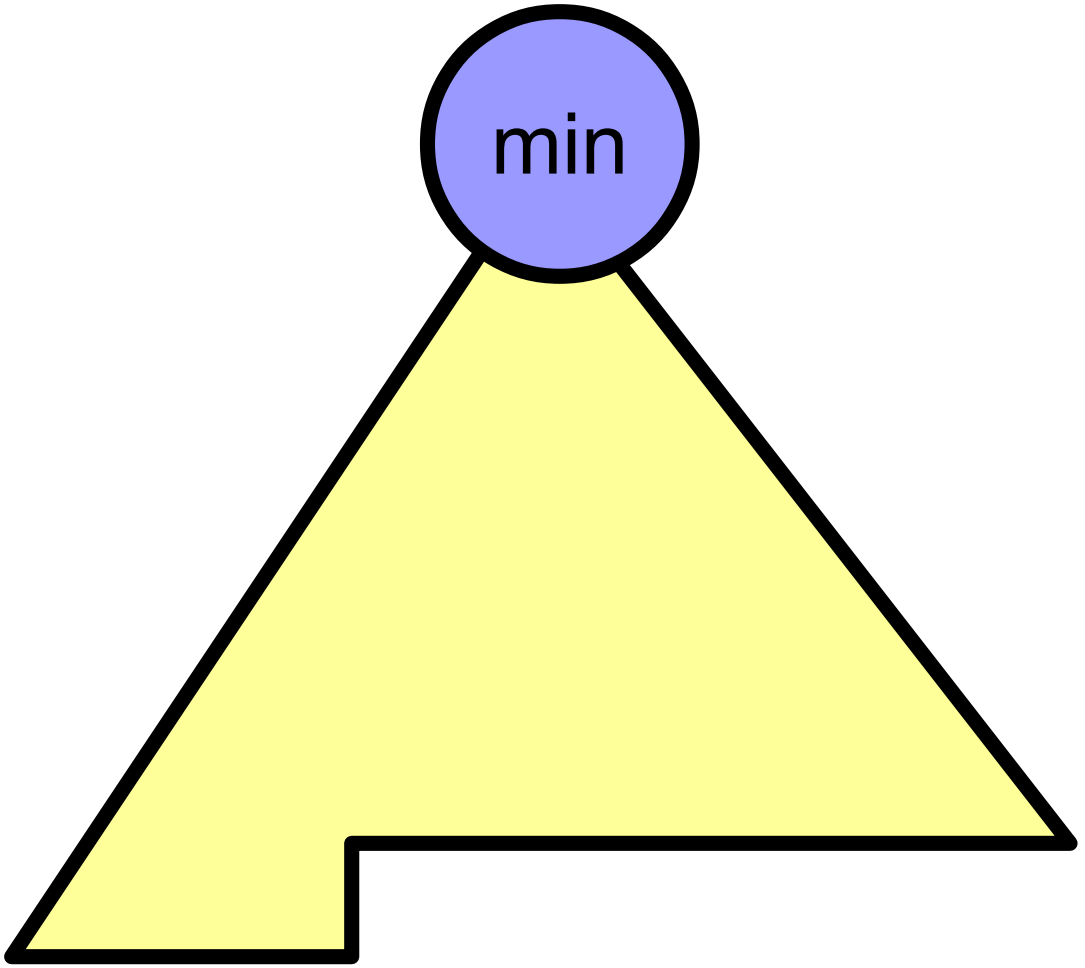

enfiler



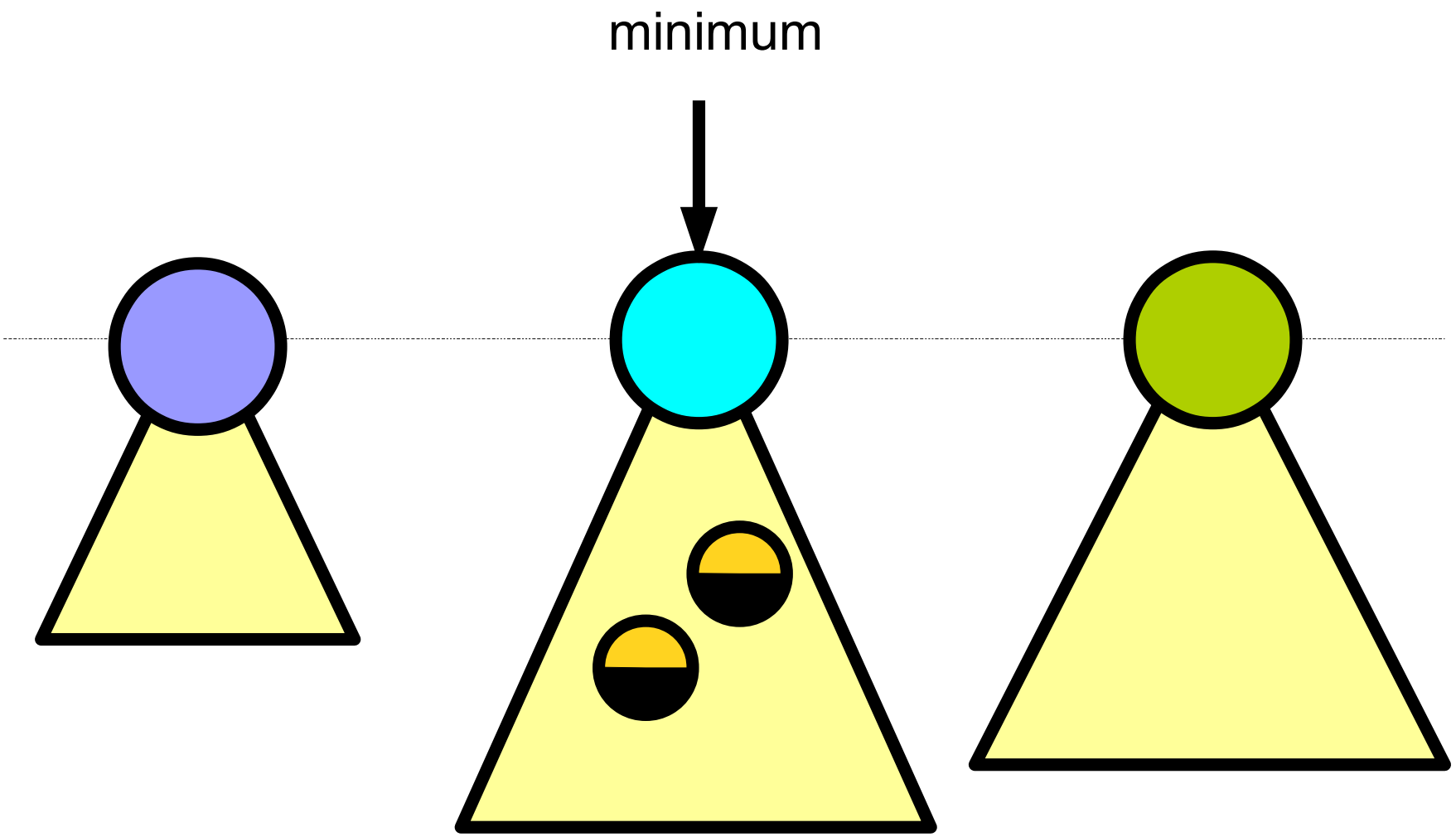
défiler min

| | créer | enfiler | Défiler min | Union | Mettre à jour diminution |
|---|---|---|---|---|---|
| Tableau | O(n) | O(1) | O(n) | O(n) | O(1) |
| Tas | O(n) | O(n) | O(log n) | O(n) | O(log n) |
| Tas de Fibonacci | O(n) | O(n) | O(log n) | O(1) | O(1) |

# Tas = arbre binaire



# Tas de Fibonacci



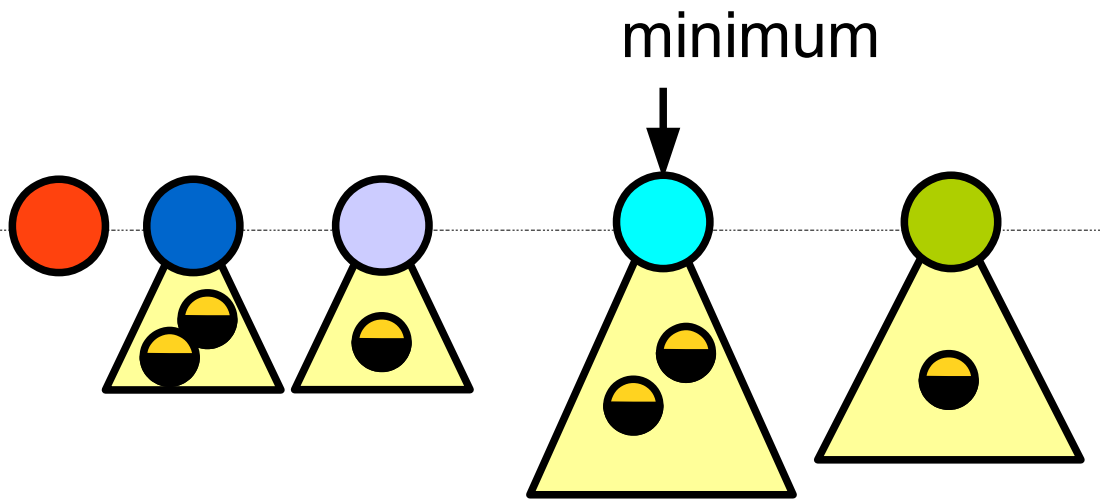
# Coût amorti

Une opération peut :

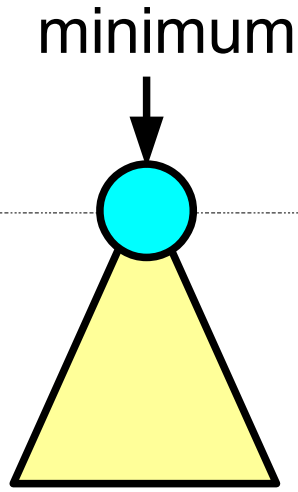
- être « rapide » mais désordonner la structure
- être « lente » car elle réordonne la structure

# Énergie potentielle [Cormen]

Énergie haute




Énergie faible




E(T) = nombre d'arbres + 2 × nombre de sommets marqués

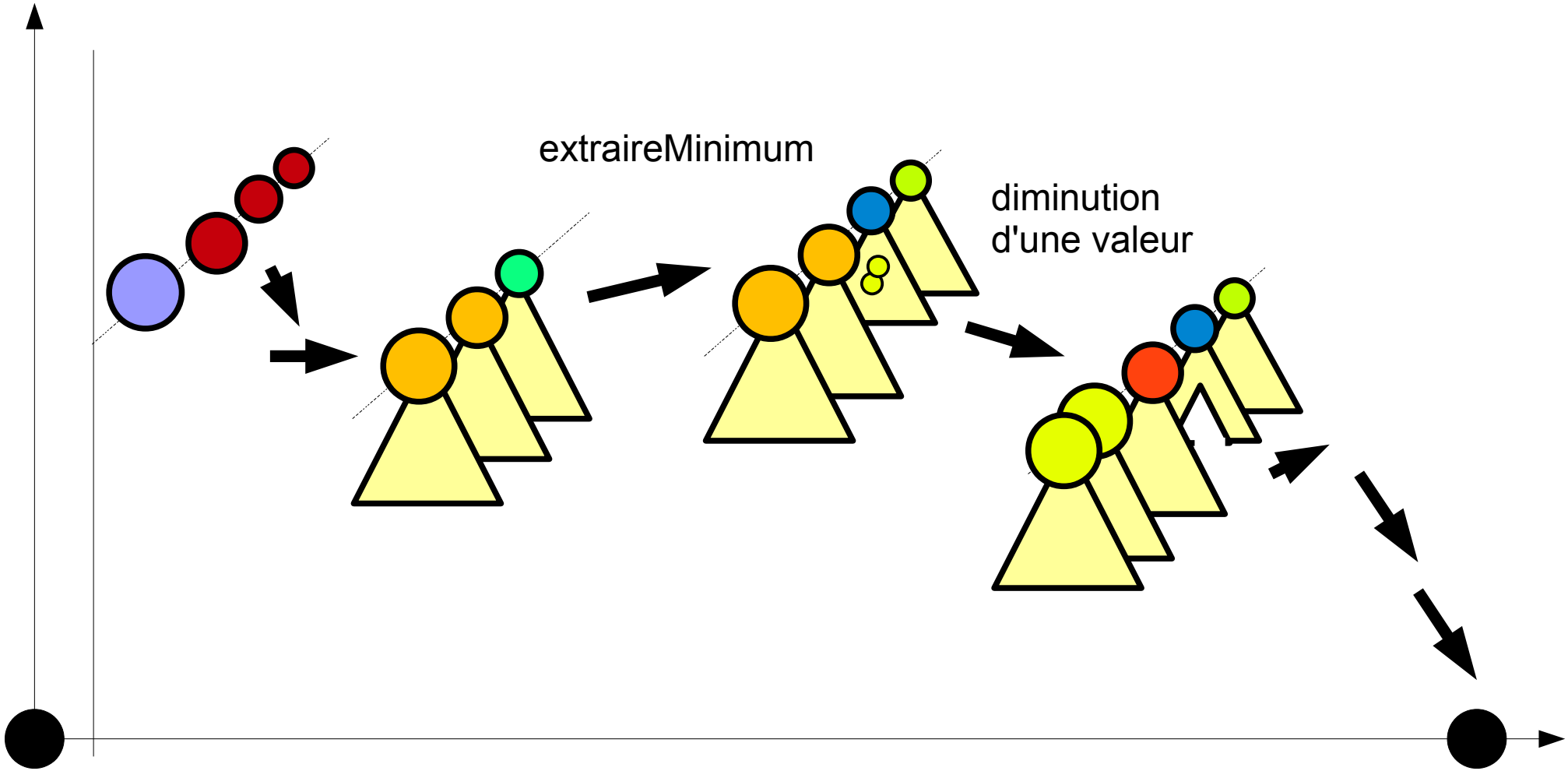
# Coût amorti

Somme du

et de la

| | Coût réel | Différence d'énergie |
|---|---|---|
| Opérations rapides qui désordonnent la structure | petit | positive |
| Opérations lentes qui rangent la structure | grand | négative |

# Évolution du tas de Fibonacci lors de l'algorithme de Dijkstra



- Somme des coût réels

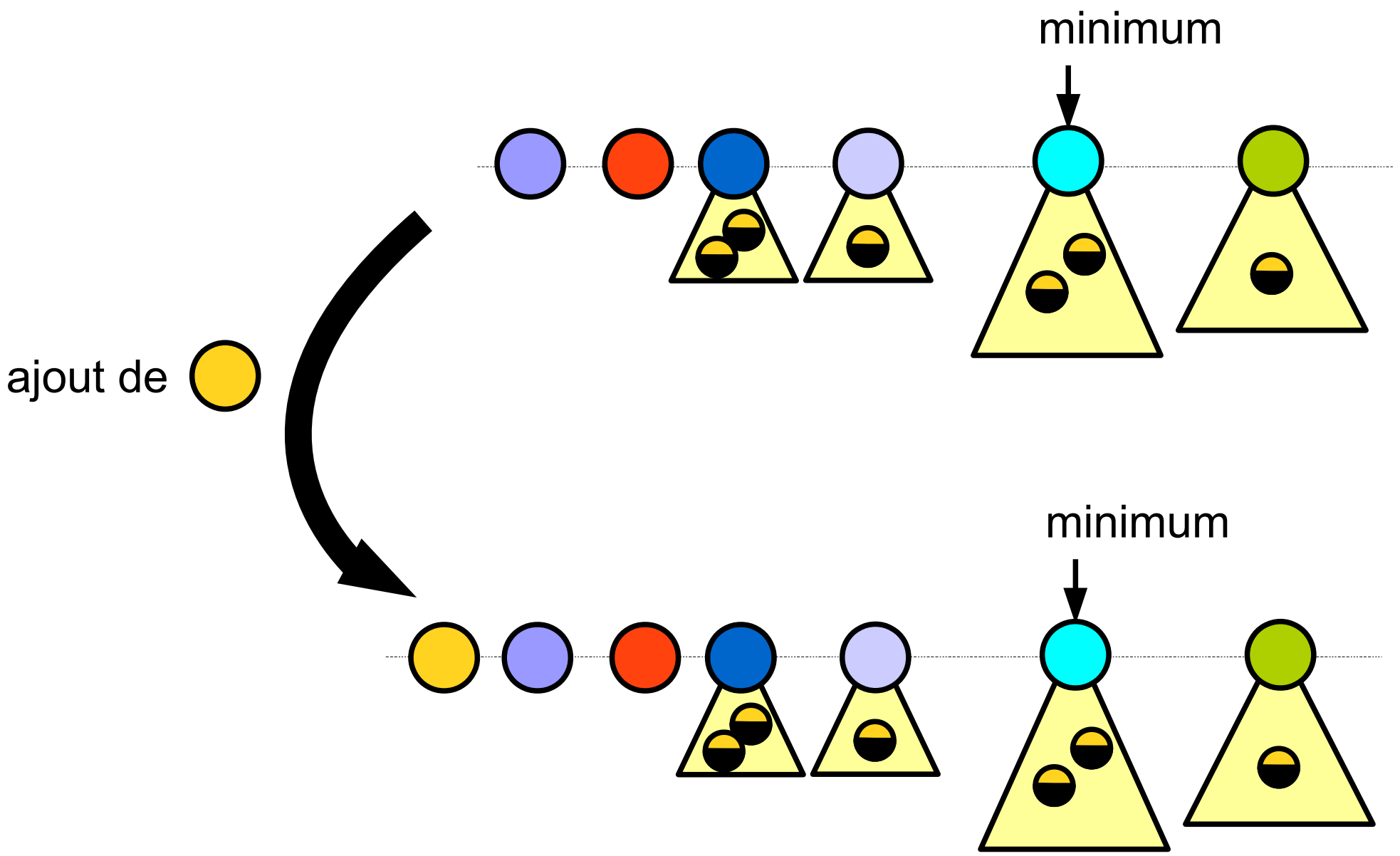
$$\sum_{i=1}^{i \leq N} c_i$$

- = somme des coûts amortis

$$\sum_{i=1}^{i \leq N} c_i + (E_{i+1} - E_i)$$

téléscopique

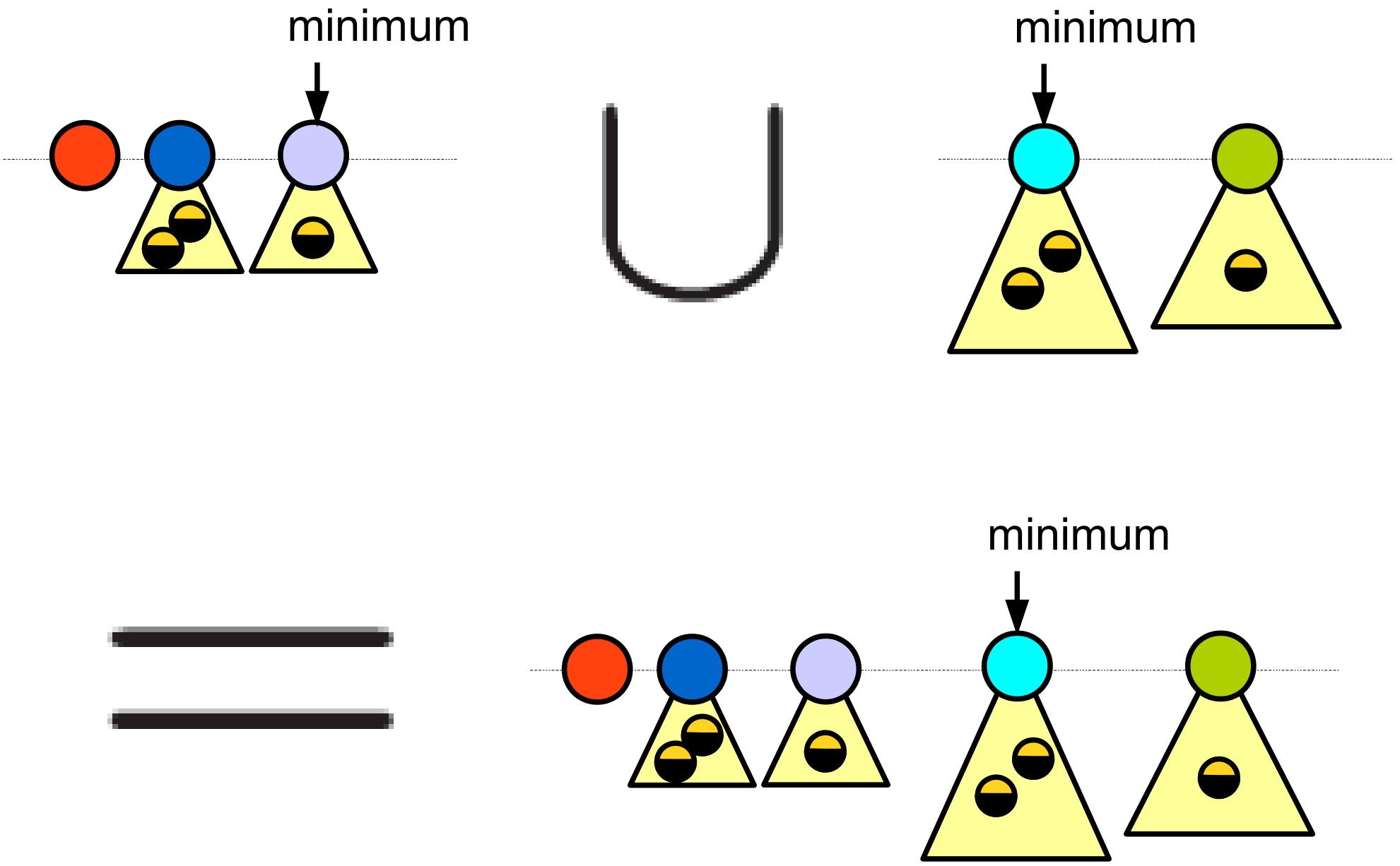
# Ajout



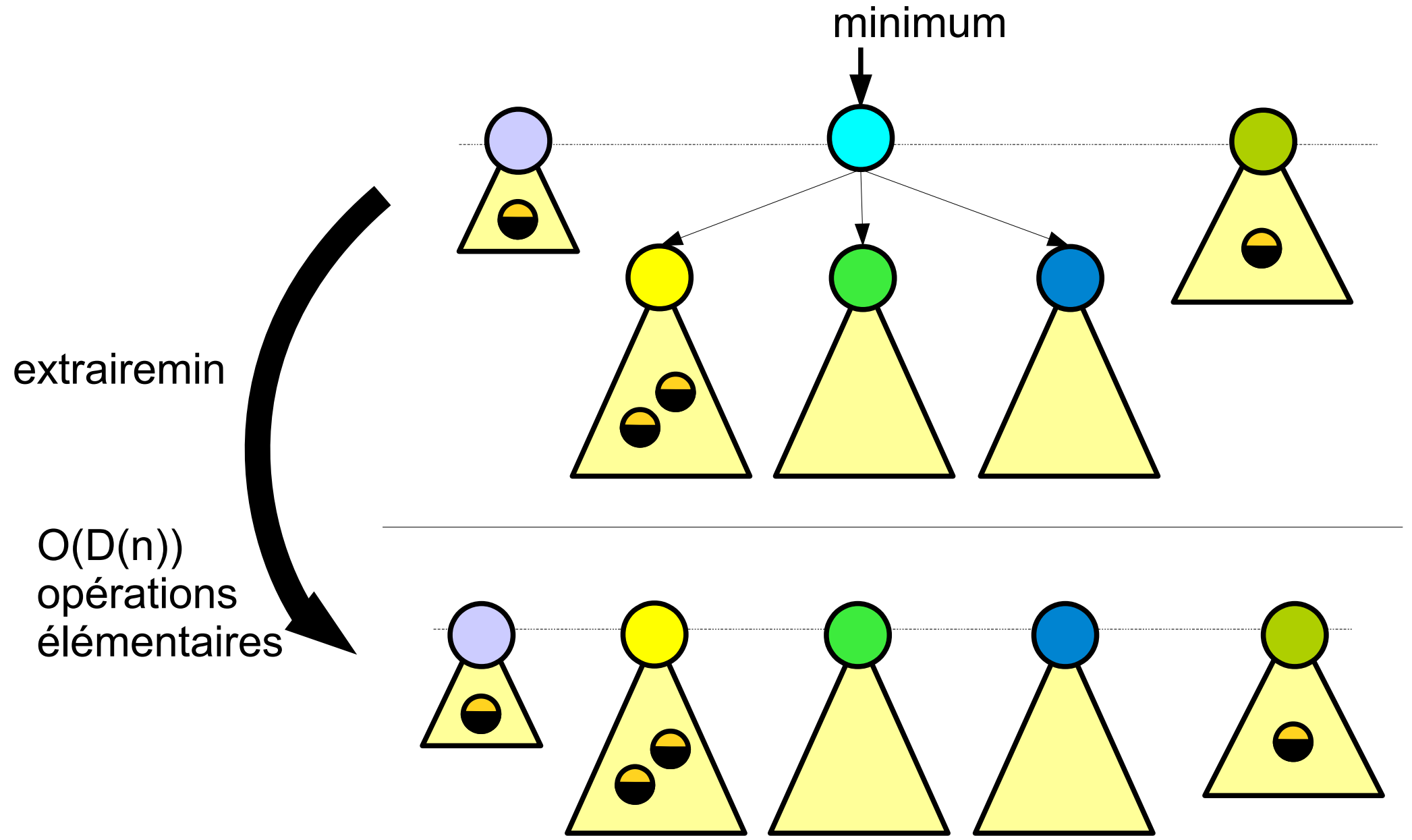
# Complexité de l'ajout

- O(1) opérations réelles
- Énergie augmente de 1
- Coût amorti : O(1) + 1 = O(1)

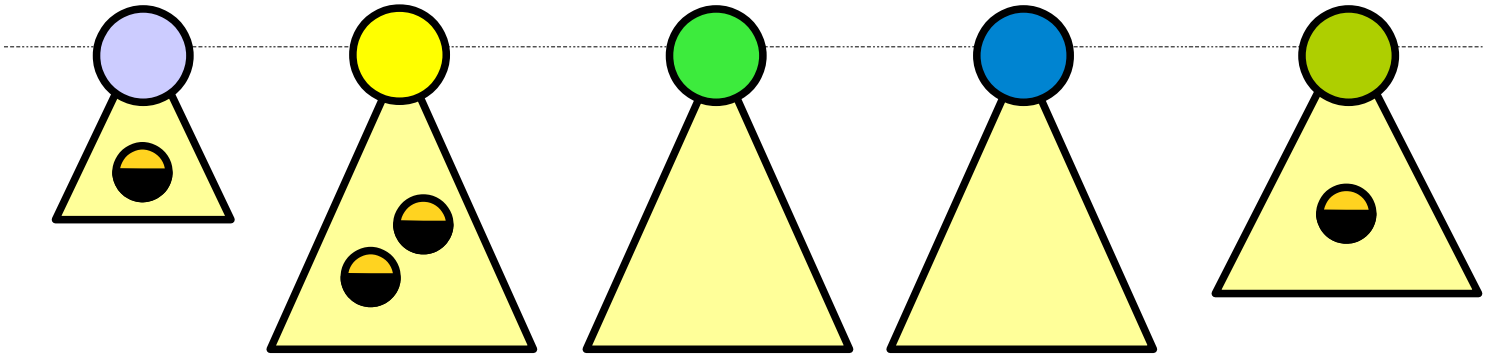
# Union



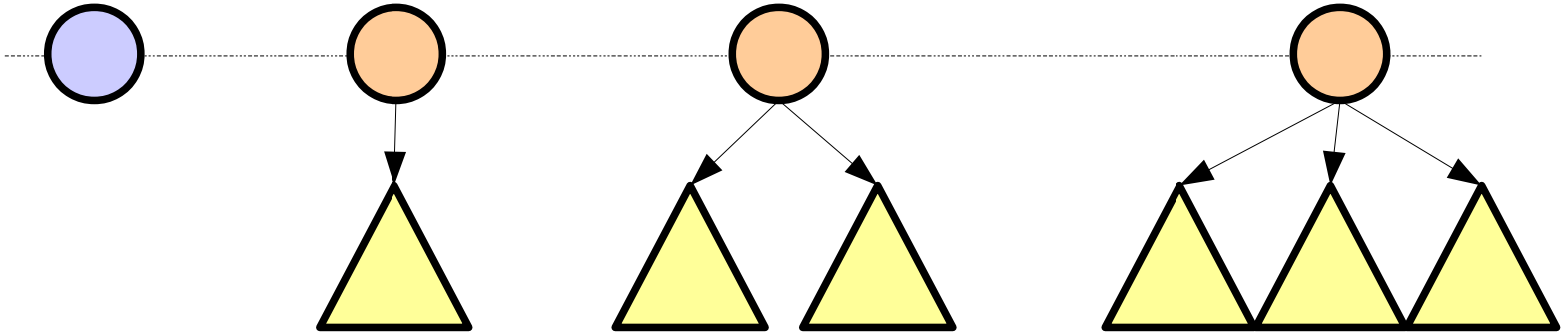
# Extraire le minimum



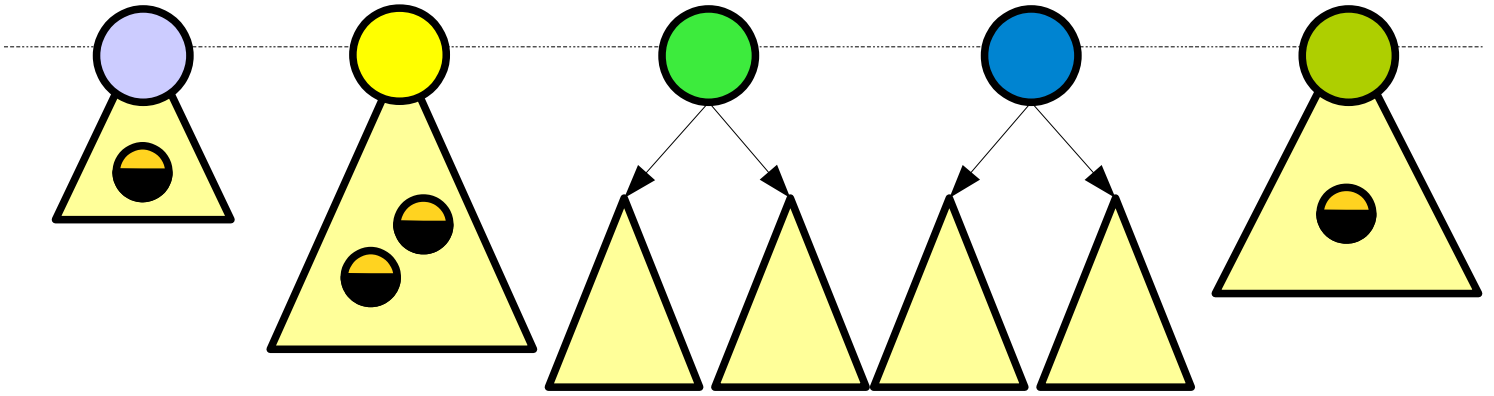
# Trouver le minimum tout en rangeant

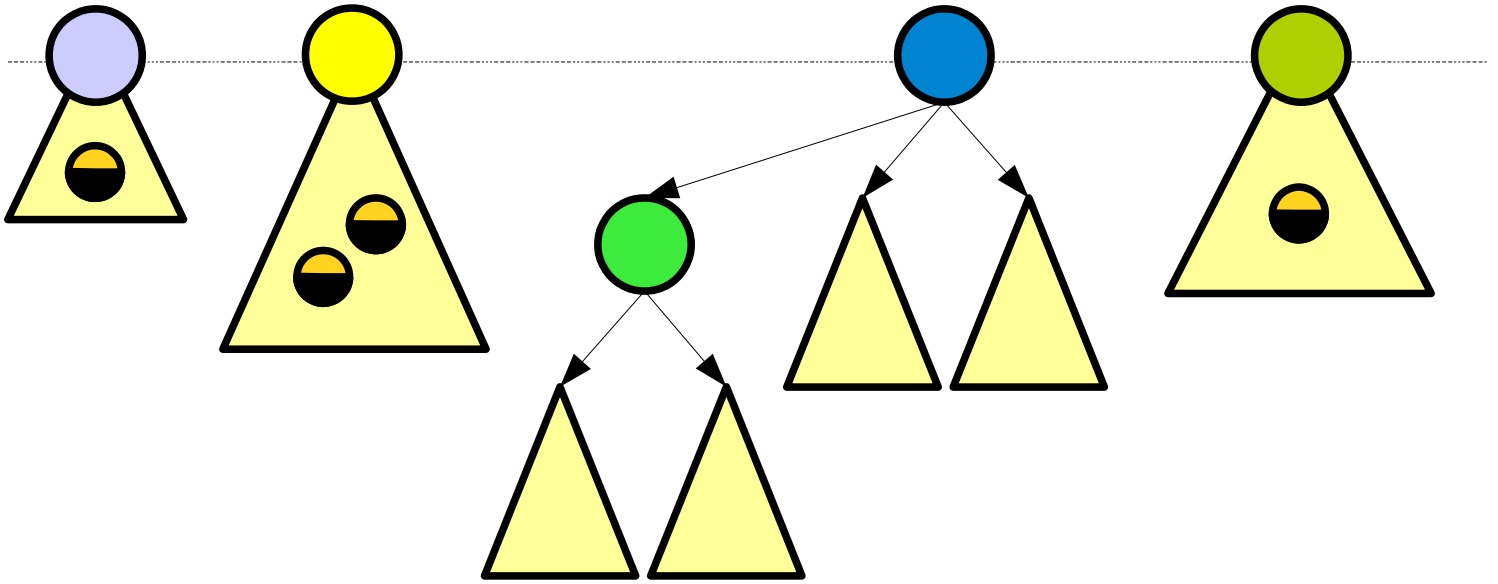


But : pas plus d'arbres que D(N) + 1 !

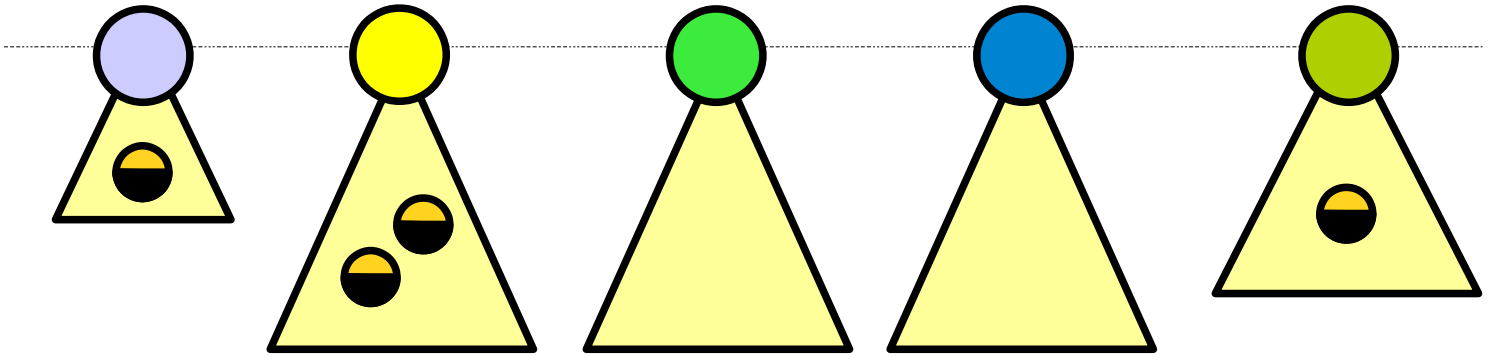


# Rangement

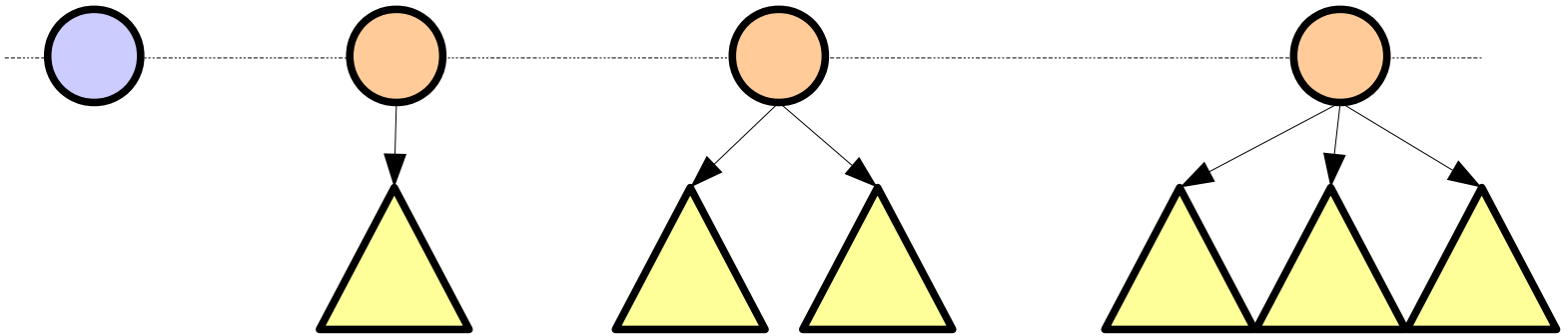




# Trouver le minimum tout en rangeant



But : pas plus d'arbres que D(N) + 1 !

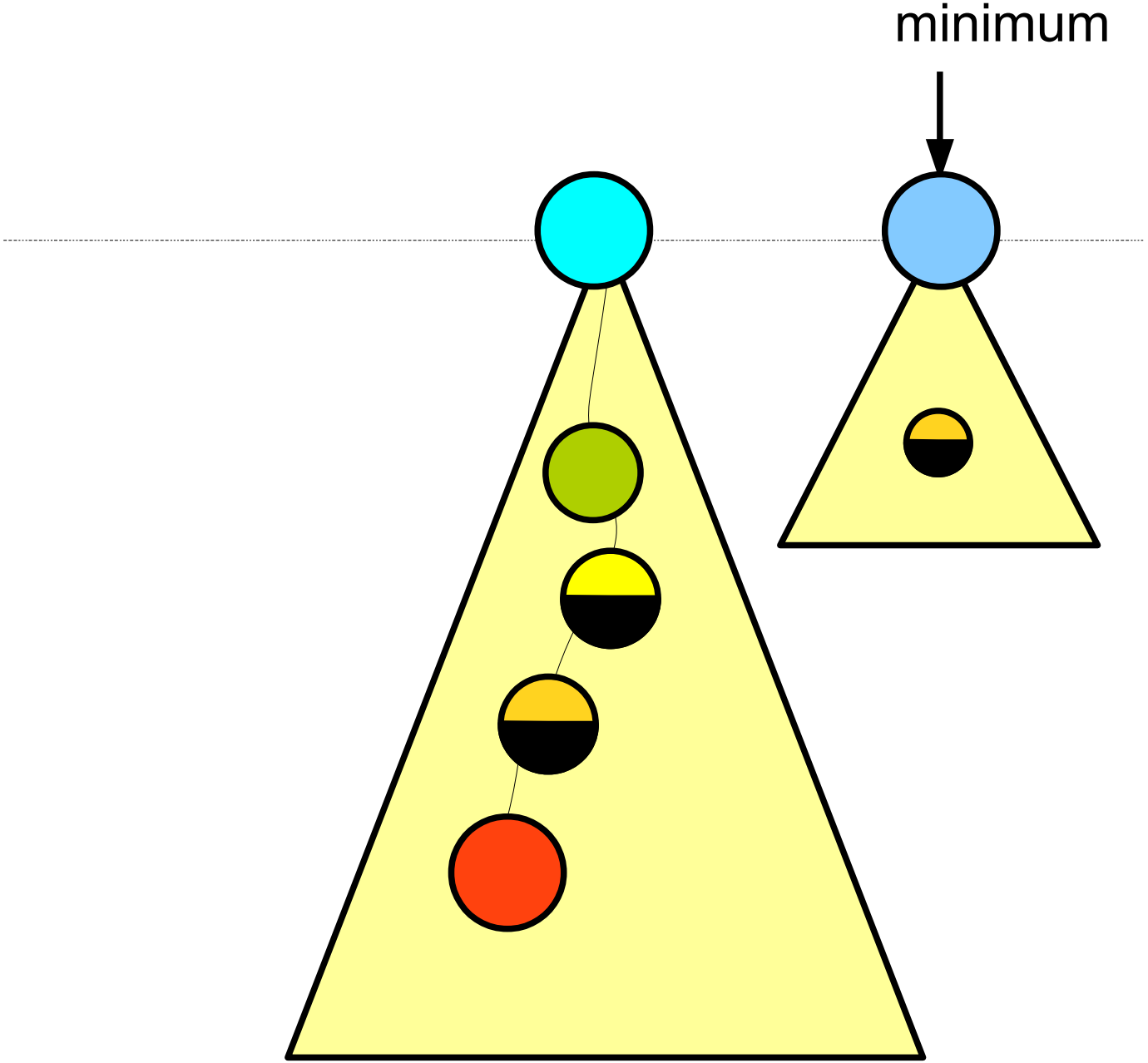


O(D(n) + a(T)) opérations

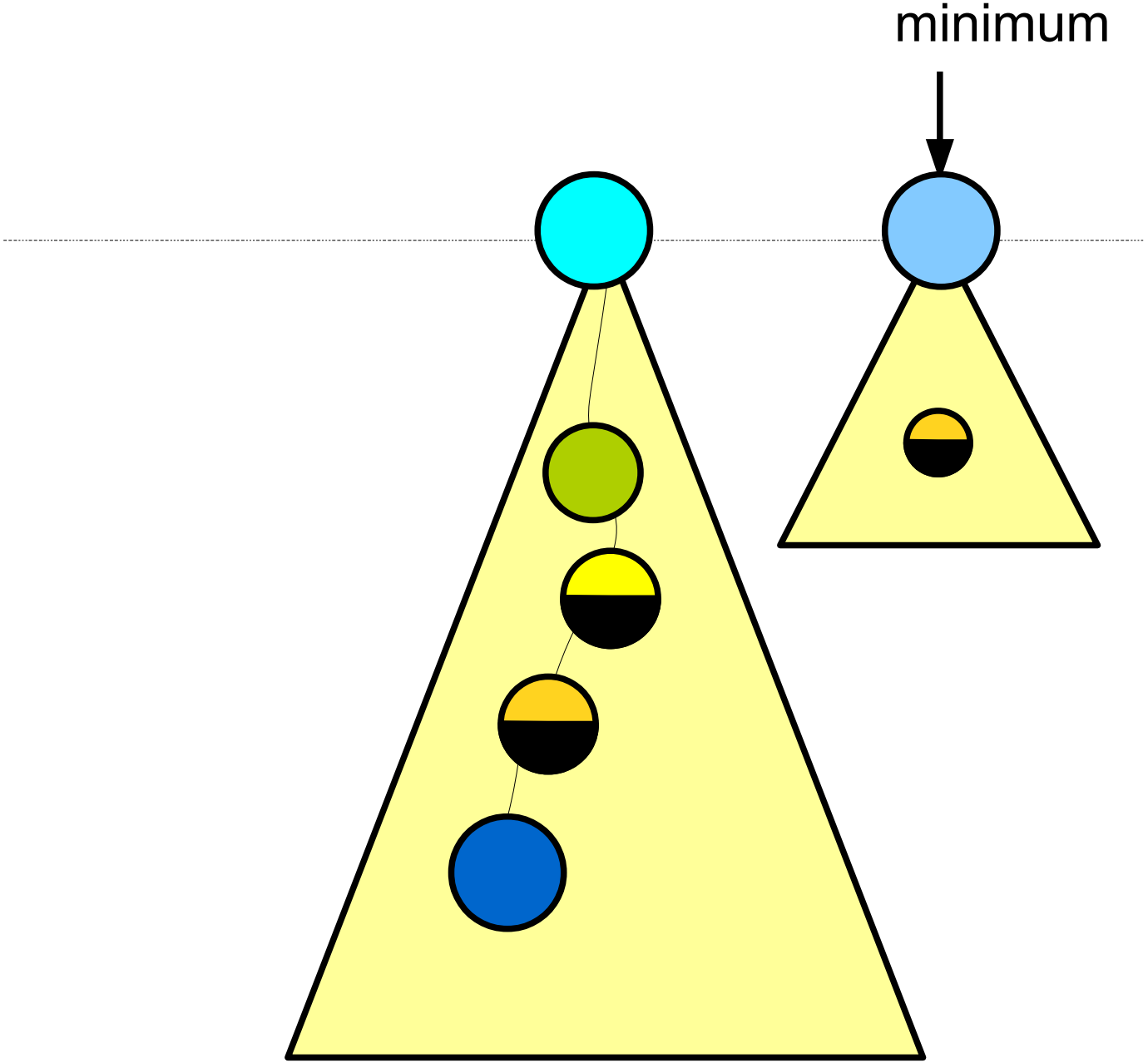
# Trouver le minimum tout en rangeant

- Coût réel : O(D(n) + a(T)) opérations
- Différence d'énergie : D(n) + 1 - a(T)
- Coût amorti : O(D(n)) = O(log n)

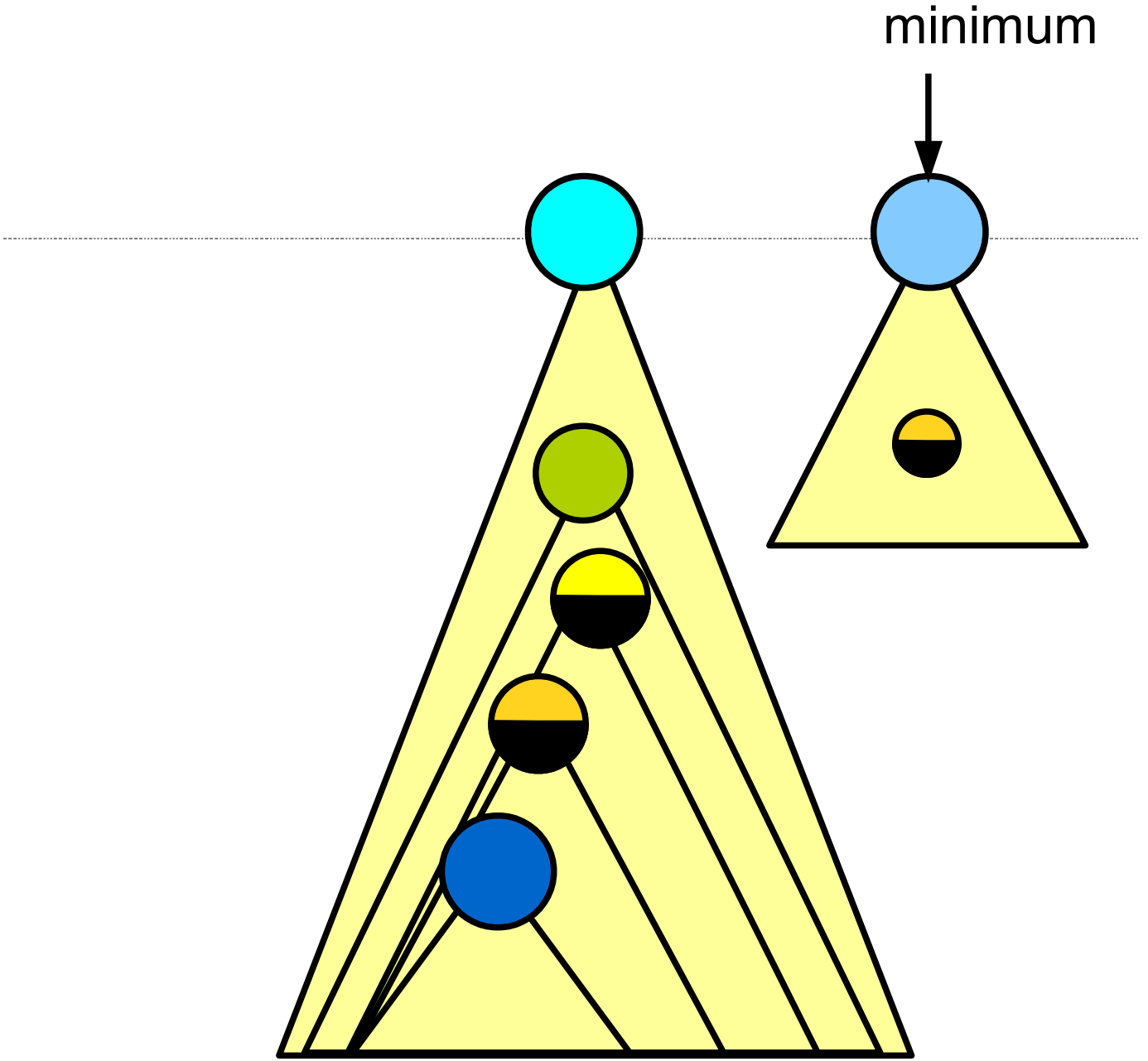
# Diminuer la clé d'un nœud



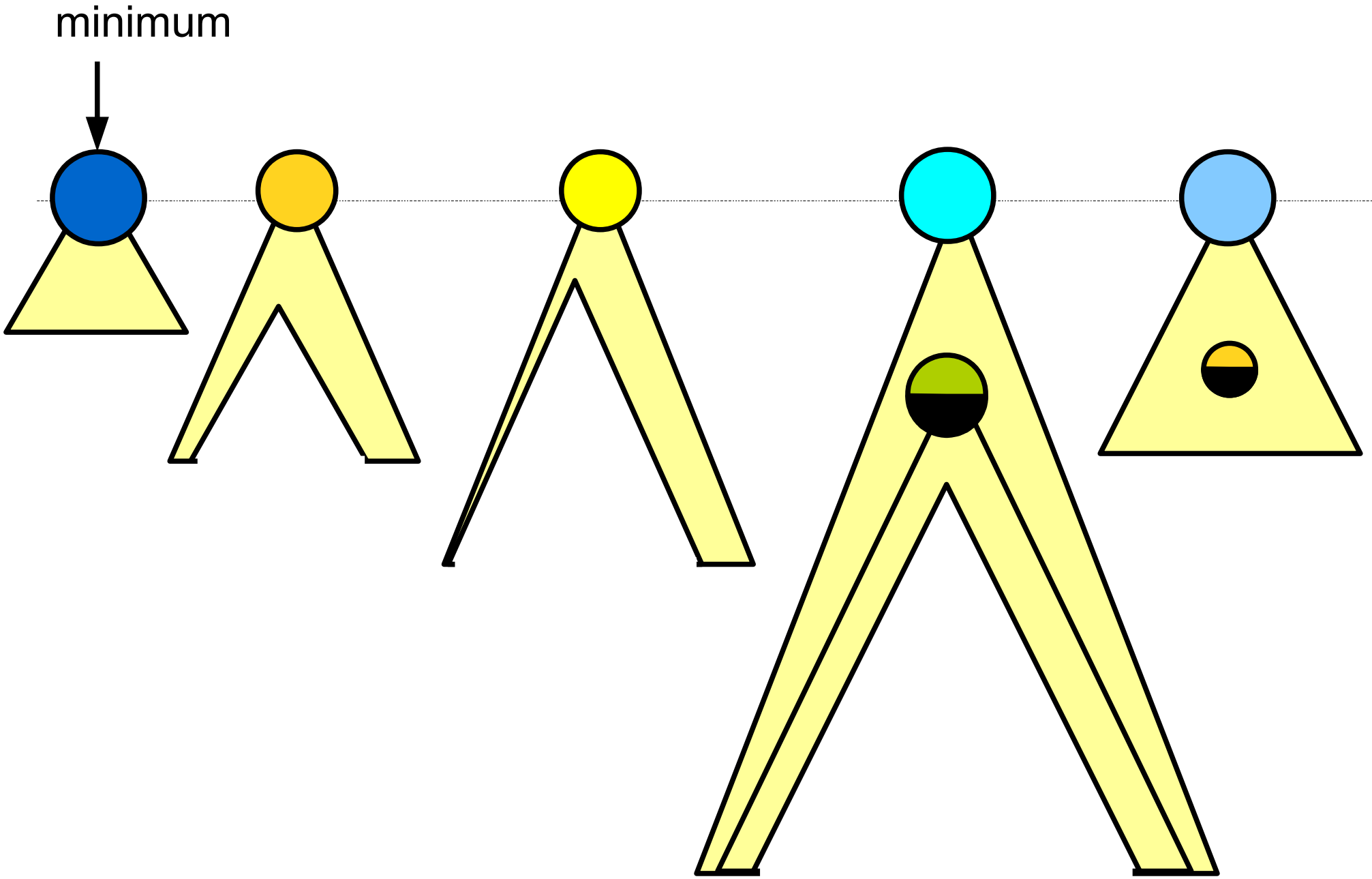
# Diminuer la clé d'un nœud

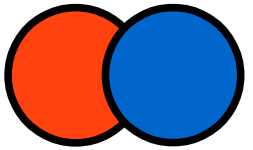


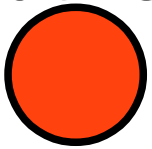
# Diminuer la clé d'un nœud



# Diminuer la clé d'un nœud



# Diminuer la clé d'un nœud 

- Coût réel : O(c) où c est le nombre d'ancêtres de  qui sont marqués
- Différence d'énergie : +c – 2(c – 1) = 4 – c
- Coût amorti : O(1)

# Idée de la preuve que $D(n) = O(\log n)$

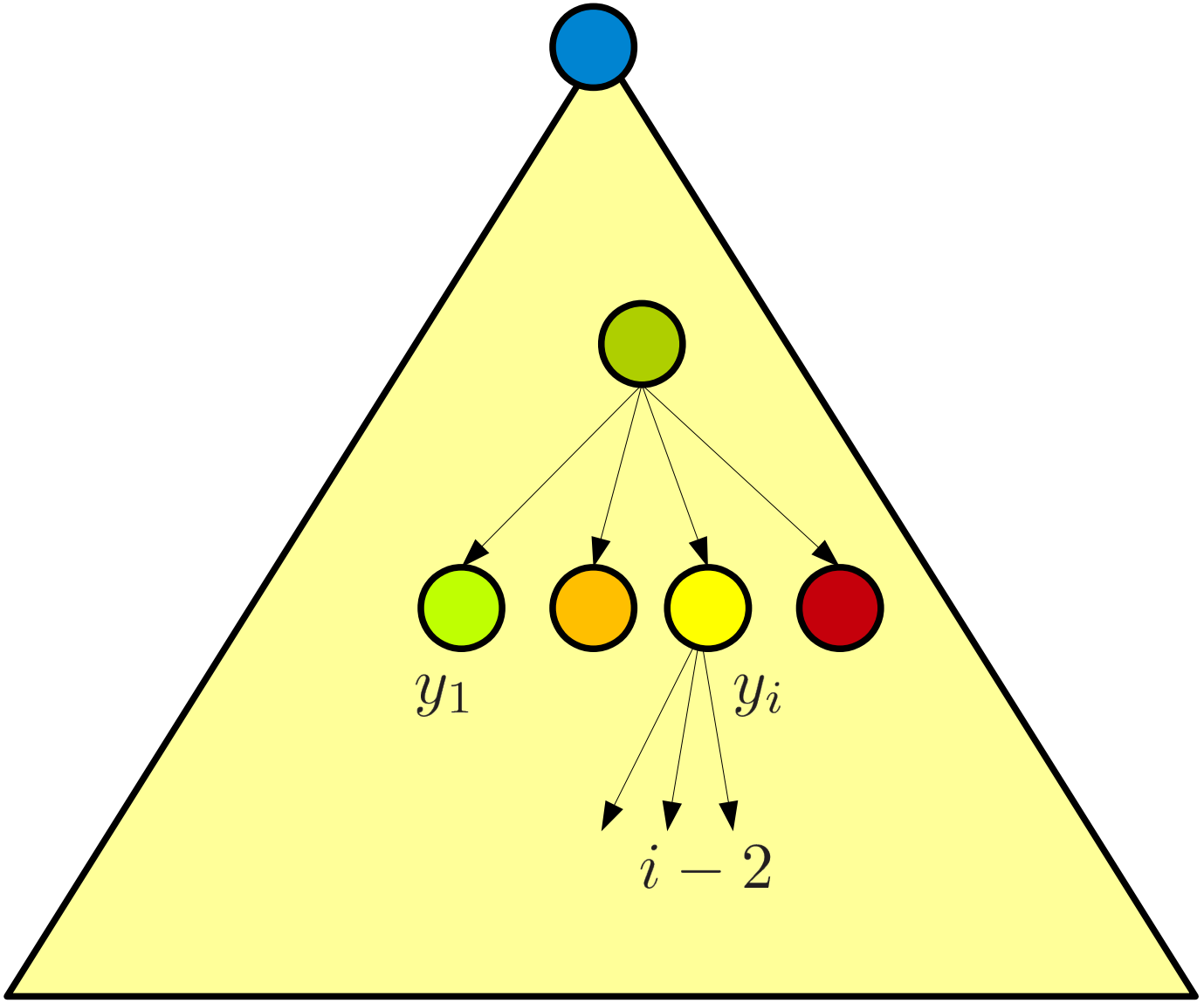
# Suite de Fibonacci

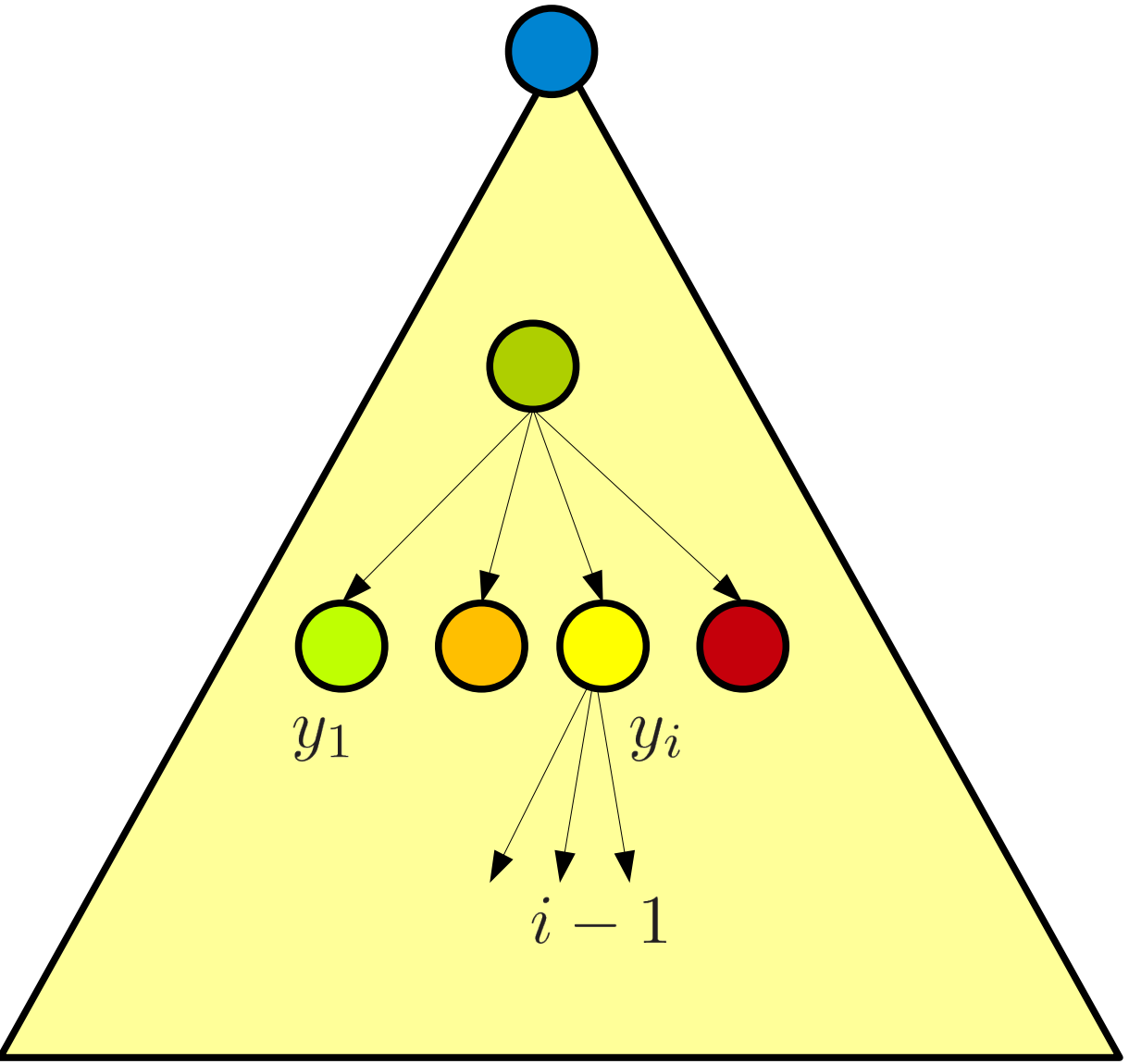
$$F_0 = 0$$
$$F_1 = 1$$
$$F_k = F_{k-1} + F_{k-2}$$

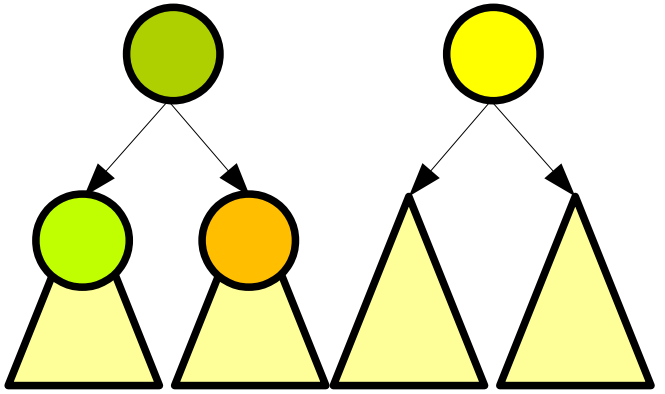
$$F_k = 1 + \sum_{i=0}^{k} F_i$$

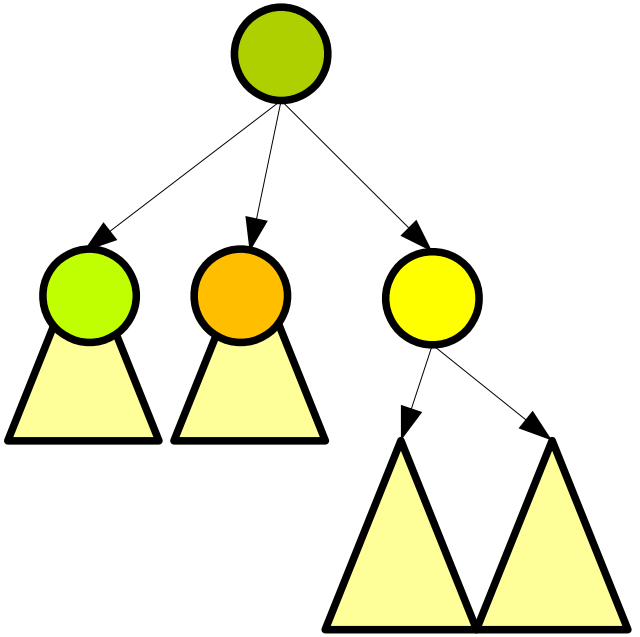
$$F_k \geq \phi^k$$

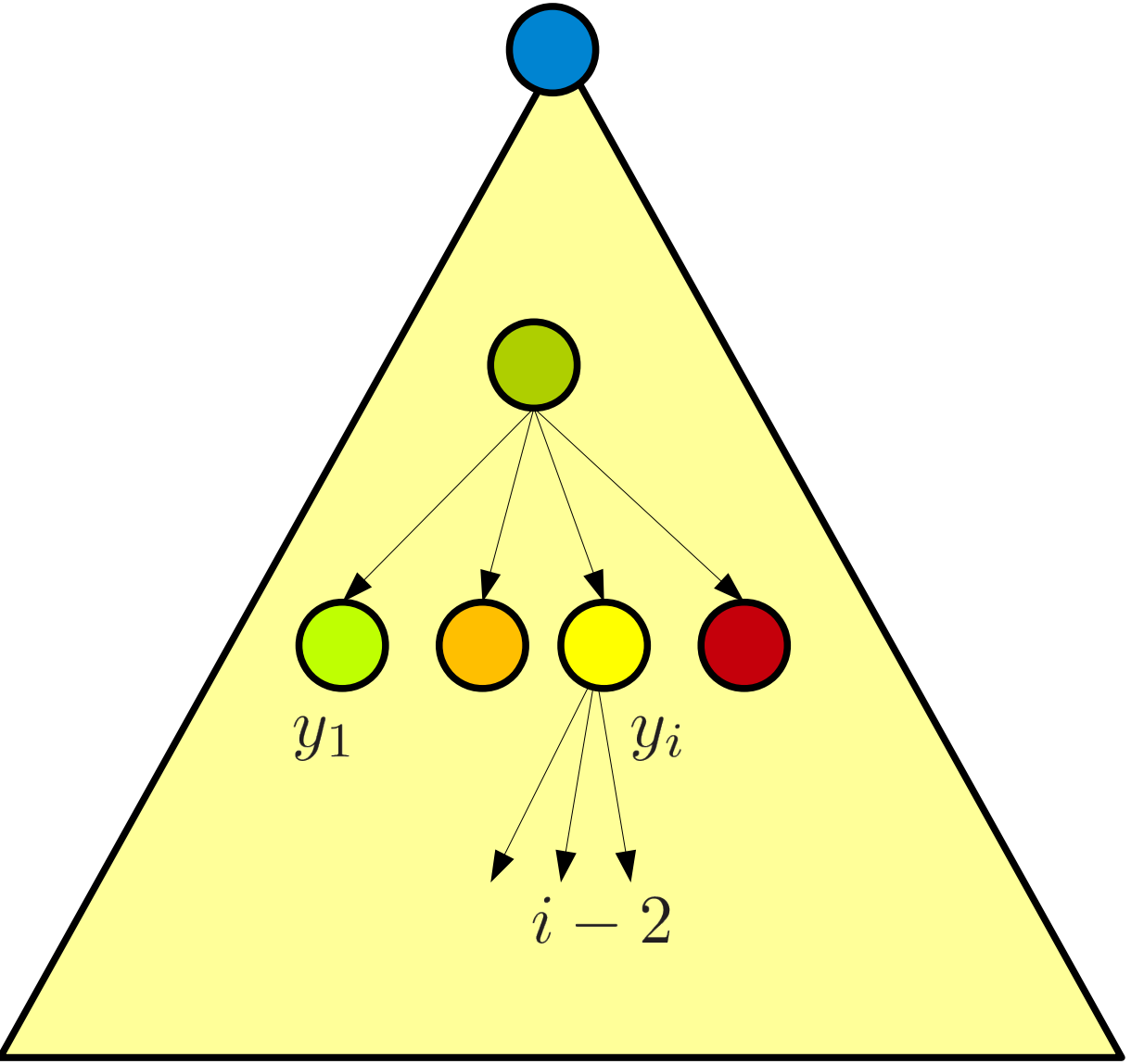
# Invariant sur les degrés

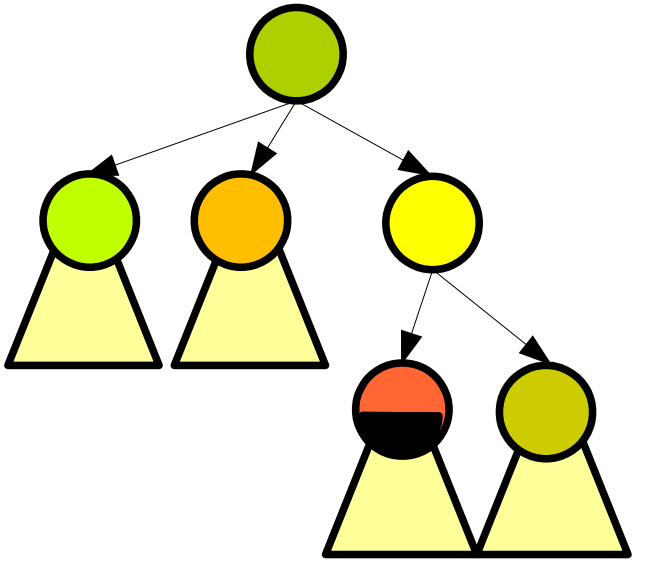


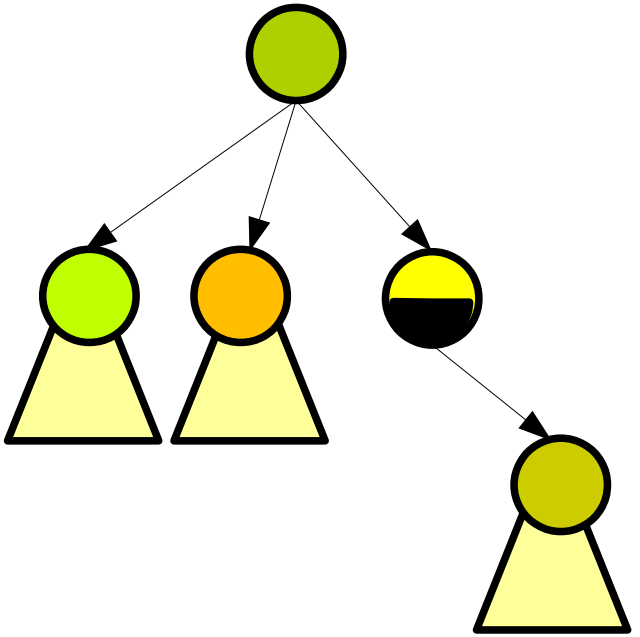

$y_1$
$y_i$
$i-1$









au plus une suppression
d'un fils de

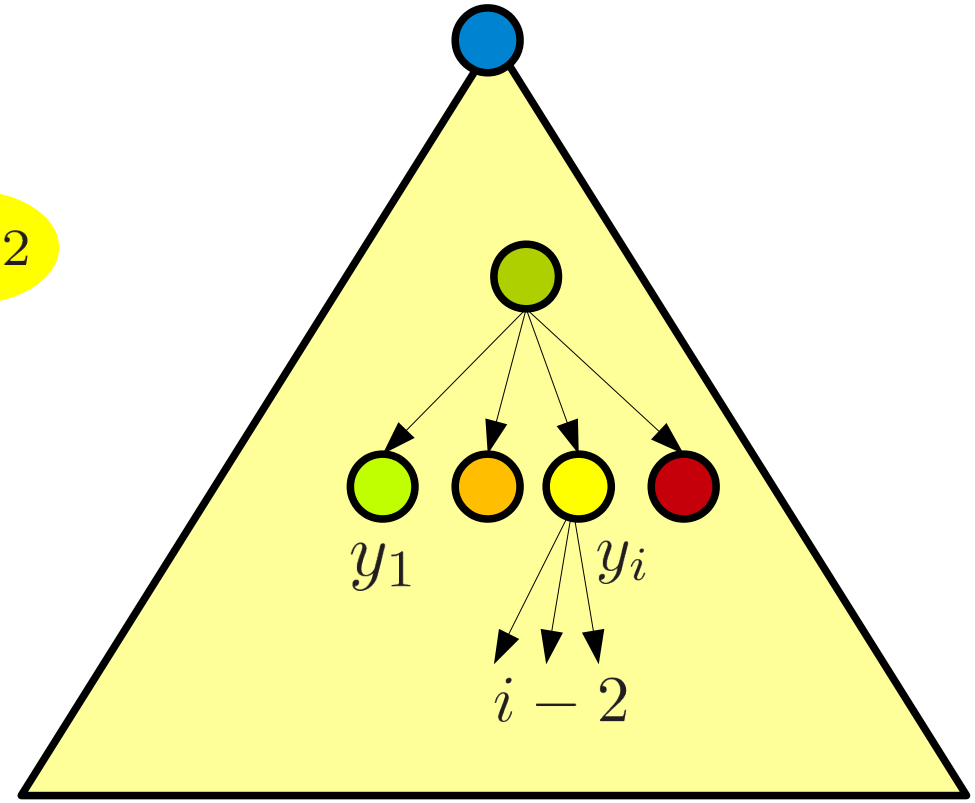
$s_d =$ taille minimale d'un nœud de degré d d'un tas de Fibonacci

$$s_d \geq F_{d+2}$$

$$s_d \geq 1 + 1 + \sum_{i=2}^{d} s_{i-2}$$

$$\geq 1 + 1 + \sum_{i=2}^{d} F_i$$

$$\geq F_{d+2}$$



$$taille(x) \geq s_d \geq F_{d+2} \geq \phi^k$$

$$log_\phi taille(x) \geq d$$