Méthodes de preuve de NP-complétude

- **transformations identité**

  (*restrictions*, *cas particuliers*)

  CLIQUE, SSP, PM, ACDB, …
- **transformations locales**

  3-SAT, Partition, PPET, …
- **transformations globales**

  SAT, VC, CIRCUITHAM, …

Une dernière remarque concernant la NP-complétude

Problèmes de reconnaissance vs optimisation

Si on sait reconnaître en temps polynomial, alors en un nombre O(log *Rés*) de reconnaissances on peut trouver l'optimum.

Si on sait trouver l'optimum alors on sait répondre au problème de reconnaissance.

Remarque : pour les problèmes d'optimisation on parle de NP-difficulté seulement.

Conclusion

Que peut-on faire quand-même ?

- heuristiques
- algorithmes d'approximation

Algorithmes d'approximation

L'idée

Ayant un problème NP-complete, que fait-on ?

ABANDONNER ?

- **Si possible – algo exponentiel**
- **Sinon, algorithme polynomial d'approximation.**
- **S'il existe, algorithme probabiliste**

Approximation ?

Qu'est-ce ?

Soit Π un pb de minimisation (optimisation)

- **C^* - la valeur d'une solution optimale**
- **C - la valeur d'une solution approchée**

$$C > C^*$$

Ratio bound $\max\{C/C^*, C^*/C\} \leq \rho(n)$

Ratio bound

$$\max\{C/C^*, C^*/C\} \leq \rho(n)$$

Erreur relative

$$|C-C^*|/C^* \leq \varepsilon(n)$$

Relation

$$\varepsilon(n) \leq \rho(n) - 1$$

Le problème Bin Packing

NOM : BIN PACKING

DONNEES : ensemble fini d'éléments U (taille $\in \mathbb{N}$) et une taille B $\in \mathbb{N}$.

QUESTION : Quel est le plus petit k tel qu'on peut partitionner U en $U_1, U_2, \ldots, U_k$ sans que la somme des tailles des éléments des U_i dépasse B ?

Le problème de décision Bin Packing

NOM : BIN PACKING

DONNEES : ensemble fini d'éléments U (taille $\in$N), une taille B $\in$N et un nombre k $\in$N.

QUESTION : Peut-on partitionner U en $U_1, U_2, \ldots, U_k$ sans que la somme des tailles des éléments des U_i dépasse B ?

La NP-complétude

Théorème :

BIN PACKING est NP-complet.

Preuve :

i) **BIN PACKING $\in$ NP**

ii) **BIN PACKING est NP-difficile**

nous le montrons par

PARTITION $\propto$ BIN PACKING

La transformation

BIN PACKING est une généralisation de **PARTITION** !

En effet, si B est la moitié de la somme des tailles des éléments et k=2, alors un **BIN PACKING** existe si et seulement si un **PARTITION** existe.

(ou … **PARTITION** est un cas particulier de **BIN PACKING** avec deux bins de taille la moitié de la somme des éléments)

Algorithme pseudo-polynomial

Pour un cas spécial :

- ***n* objets (mais seulement *k* tailles différentes)**
- **bins de taille *B***

Ainsi les données deviennent : $I = (i_1, i_2, \ldots, i_k)$**,**
avec i_j **le nombre d'objets de taille** j**.**

On utilise la programmation dynamique.

suite

- BINS($i_1, i_2, \ldots, i_k$) est le nombre minimum de bins nécessaires pour $I = (i_1, i_2, \ldots, i_k)$.
- Soit ($n_1, n_2, \ldots, n_k$) une donnée $\Sigma_i n_i = n$.
- On calcule Q, l'ensemble de tous les k-uplets tels que BINS($q_1, q_2, \ldots, q_k$) = 1.

  Au plus O(n^k), donc se calcule en temps O(n^k).

suite

On rempli la table BINS(i_1, i_2, …, i_k), avec $i_j \leq n_j$.

1. $\forall q \in Q$ BINS(q_1, q_2, …, q_k) = 1.
2. Si $\exists$ j, t.q. $i_j < 0$, alors BINS(i_1, i_2, …, i_k) = ∞.
3. Pour tout autre q, utiliser la récurrence

$$\text{BINS}(i_1, i_2, \ldots, i_k) = 1 + \min_{q \in Q} \text{BINS}(i_1 - q_1, i_2 - q_2, \ldots, i_k - q_k).$$

suite

Question : **dans quel ordre faut-il calculer les valeurs de BINS ?**

Complexité :

- **chaque entrée du tableau : O(n^k)**
- **le tableau est de taille : O(n^k)**
- **temps total : O(n^{2k})**

Un heuristique : FF (first fit)

- Pour chaque élément, on examine les bins, et on le place dans le premier où il y a de la place
- Complexité : $\Theta(nk)$ où k est le résultat.

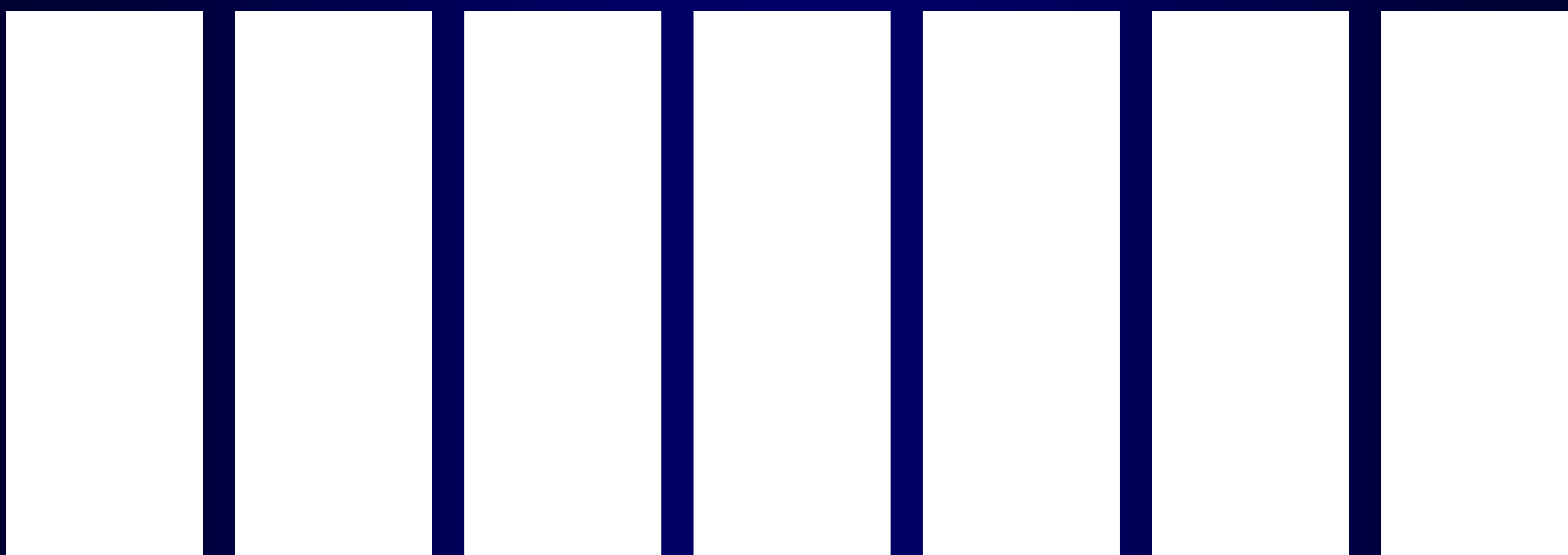
- Est-ce un "bon" heuristique ? Pour une donnée D, soit FF(D) le résultat obtenu par FF et OPT(D) la solution optimale.

FF (suite)

Pour se faire une idée de la "qualité" de l'heuristique, nous proposons deux exemples

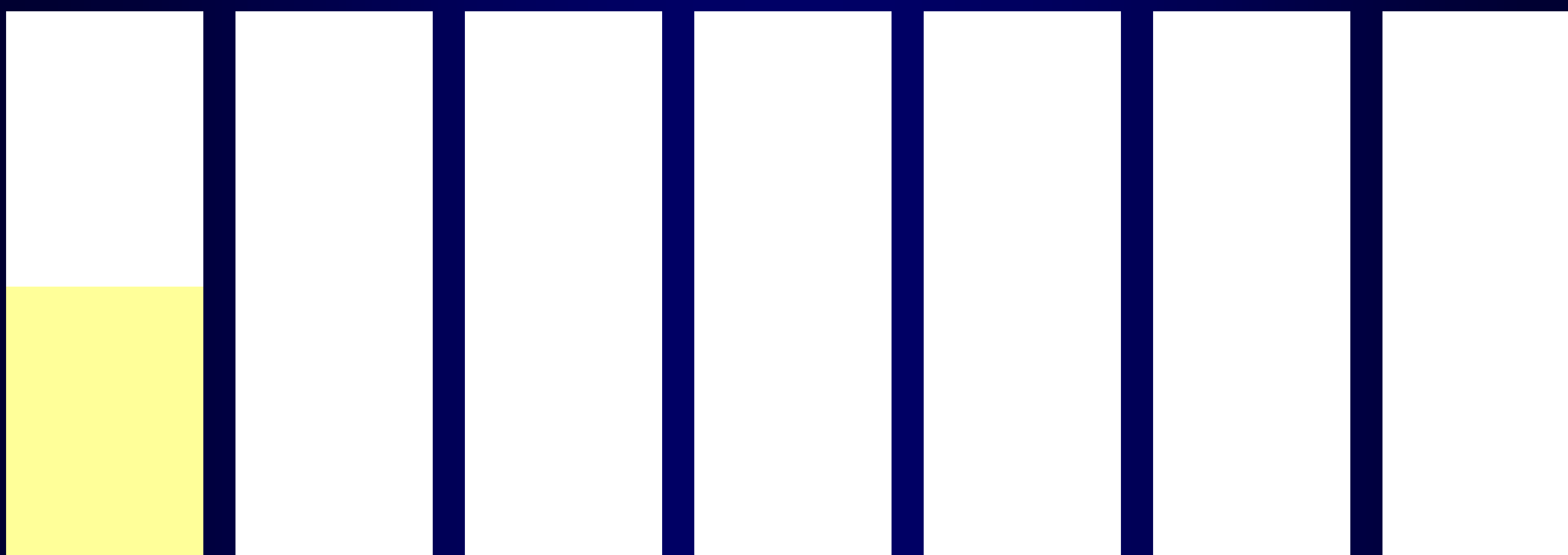
FF (exemple)

B = 420



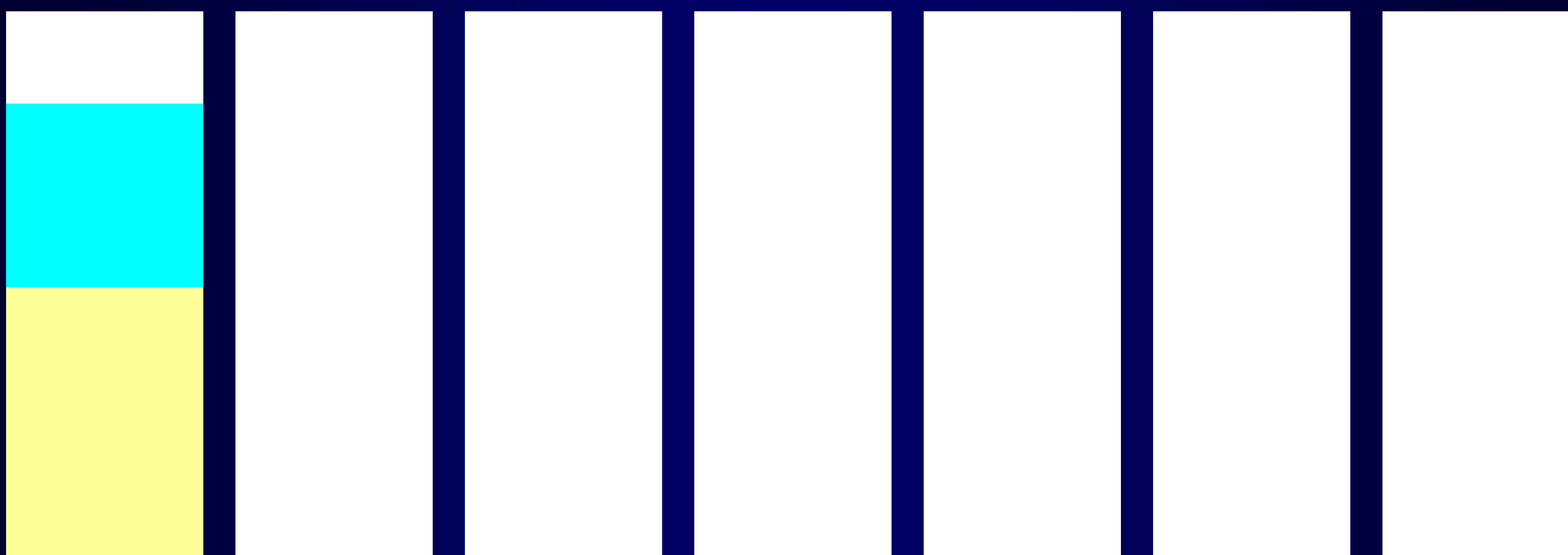
FF (exemple)

B = 420 et 211



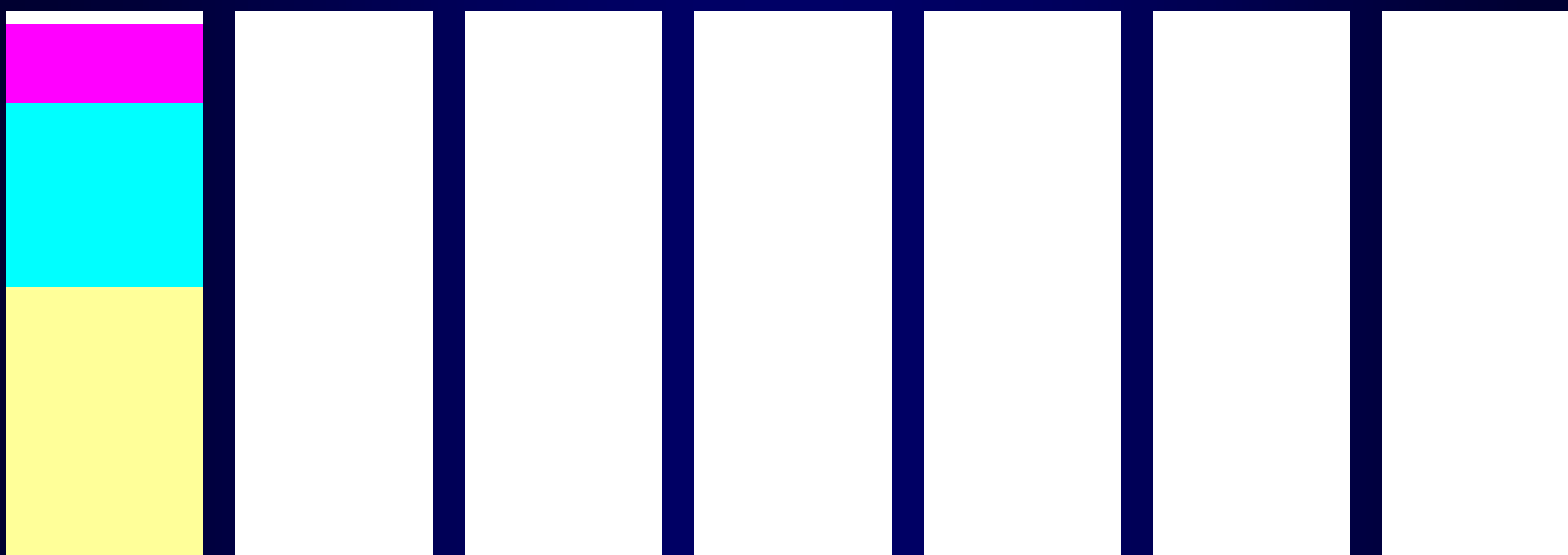
FF (exemple)

B = 420 et 211, 141



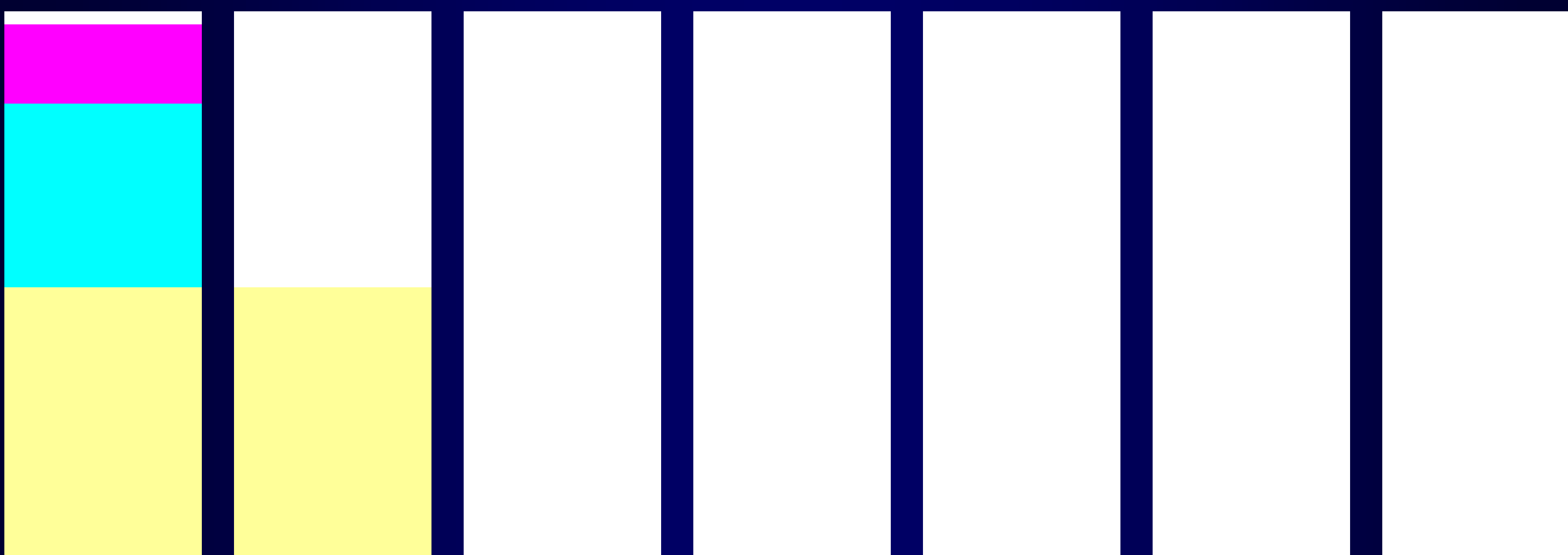
FF (exemple)

B = 420 et 211, 141, 61



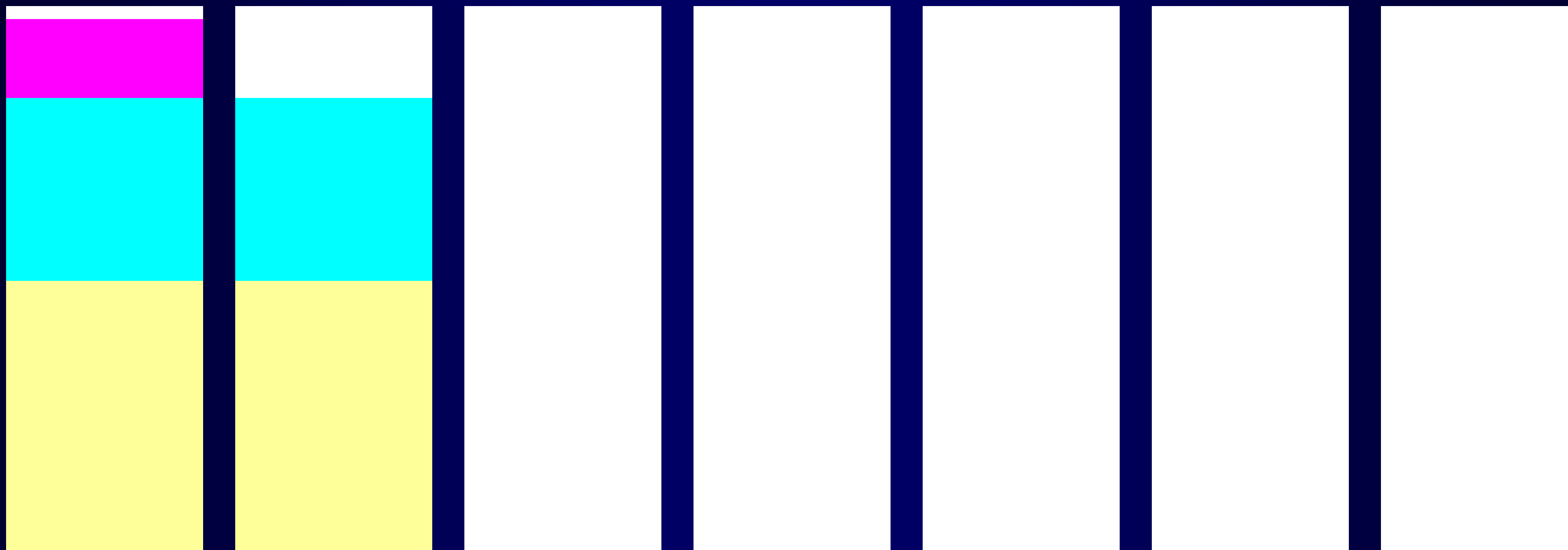
FF (exemple)

B = 420 et 211, 141, 61, 211



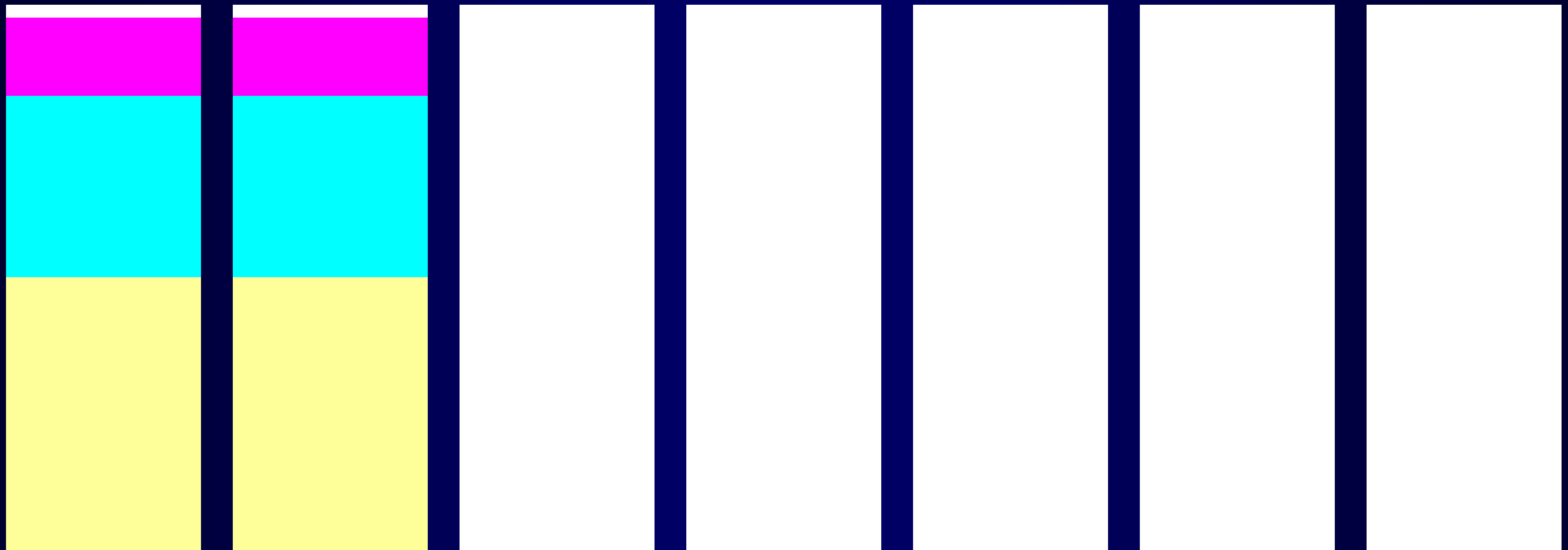
FF (exemple)

B = 420 et 211, 141, 61, 211, 141,



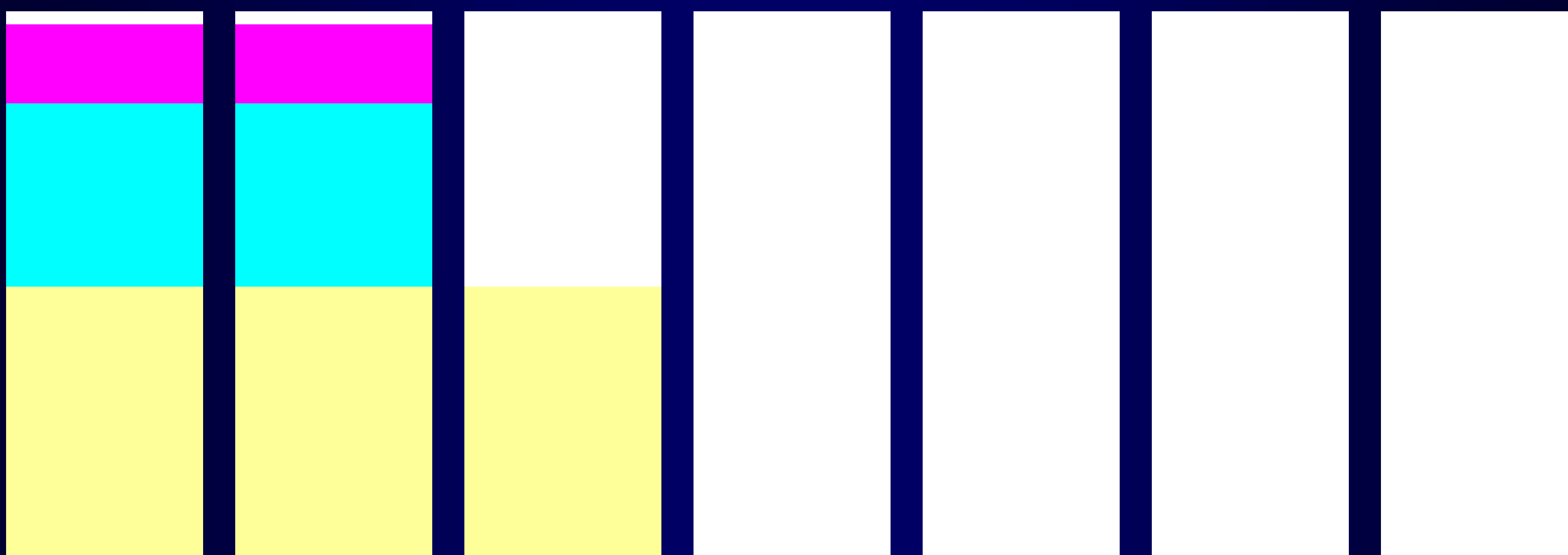
FF (exemple)

B = 420 et 211, 141, 61, 211, 141, 61



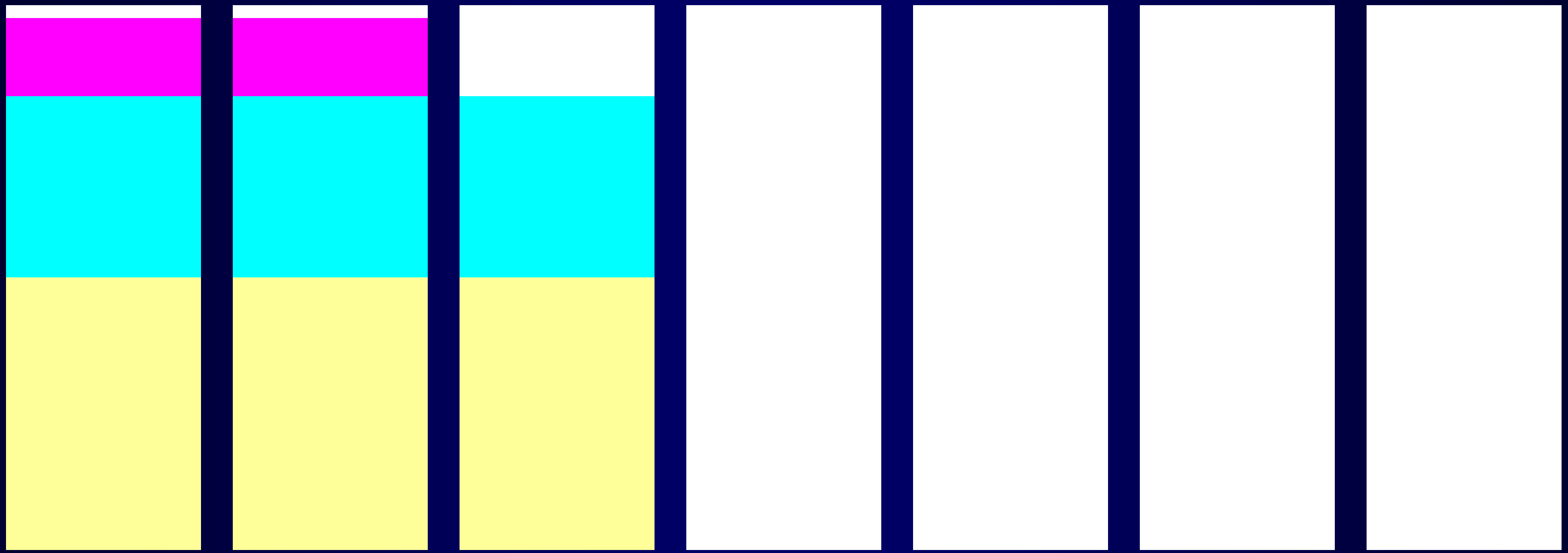
FF (exemple)

B = 420 et 211, 141, 61, 211, 141, 61, 211



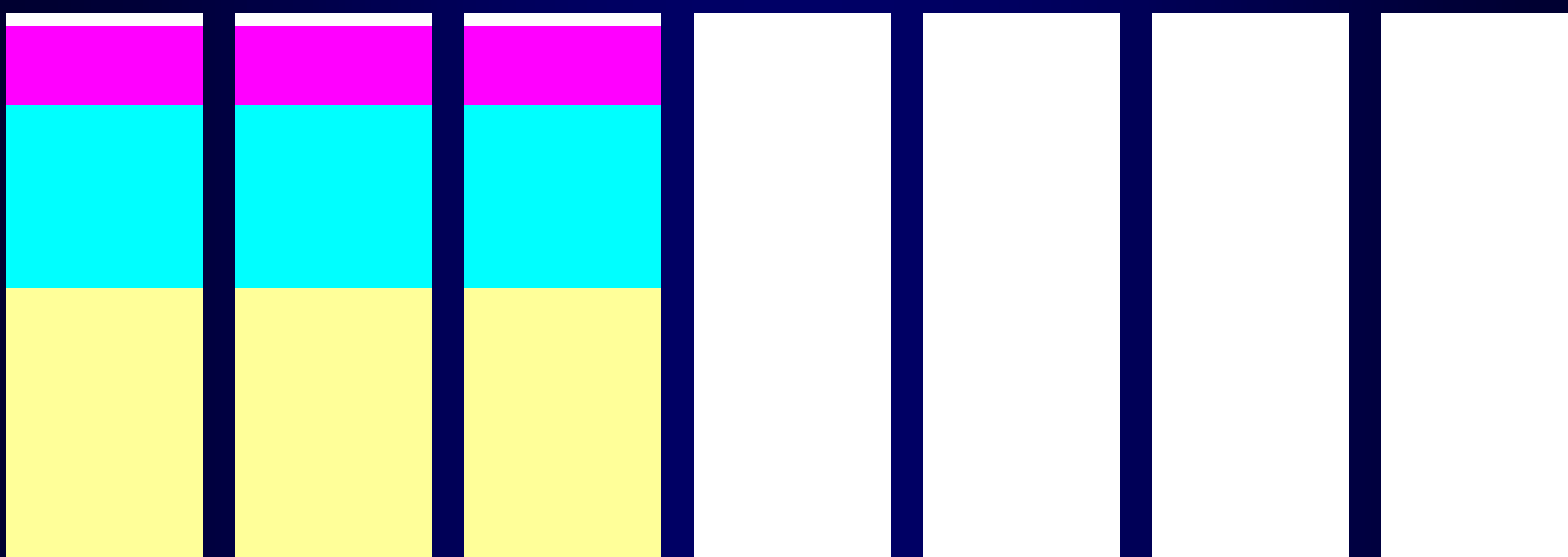
FF (exemple)

B = 420 et 211, 141, 61, 211, 141, 61, 211, 141



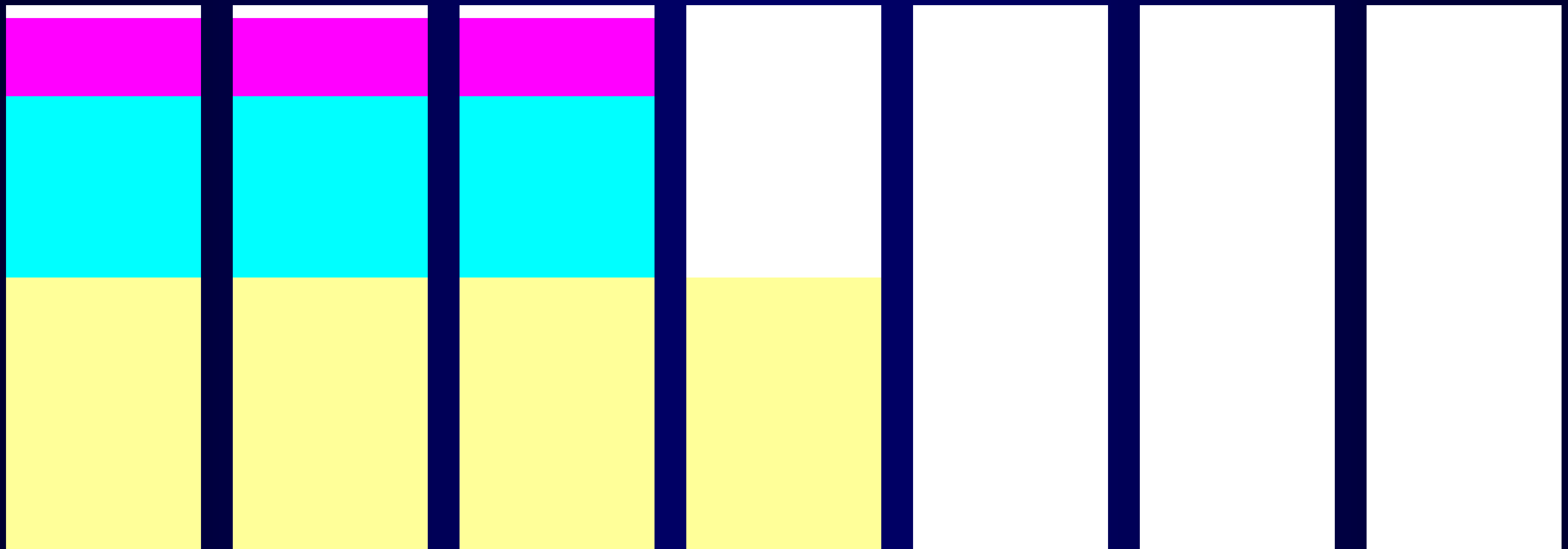
FF (exemple)

B = 420 et 211, 141, 61, 211, 141, 61, 211, 141, 61



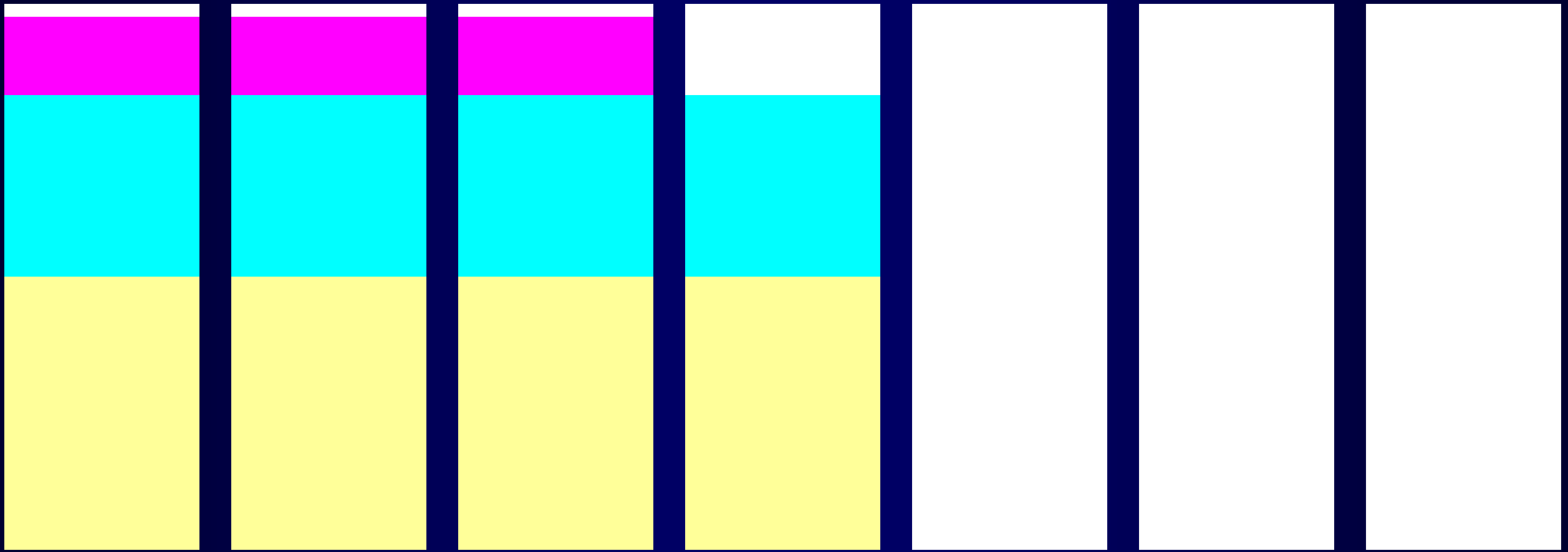
FF (exemple)

B = 420 et 211, 141, 61, 211, 141, 61, 211, 141, 61, 211



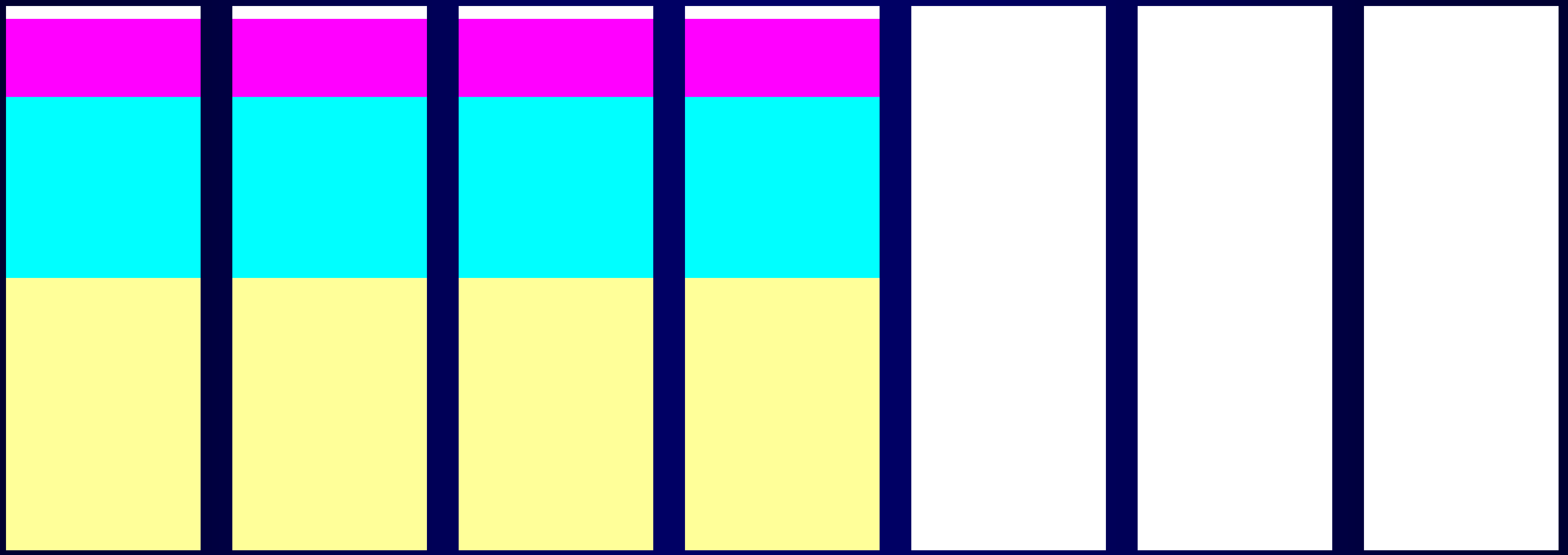
FF (exemple)

B = 420 et 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141



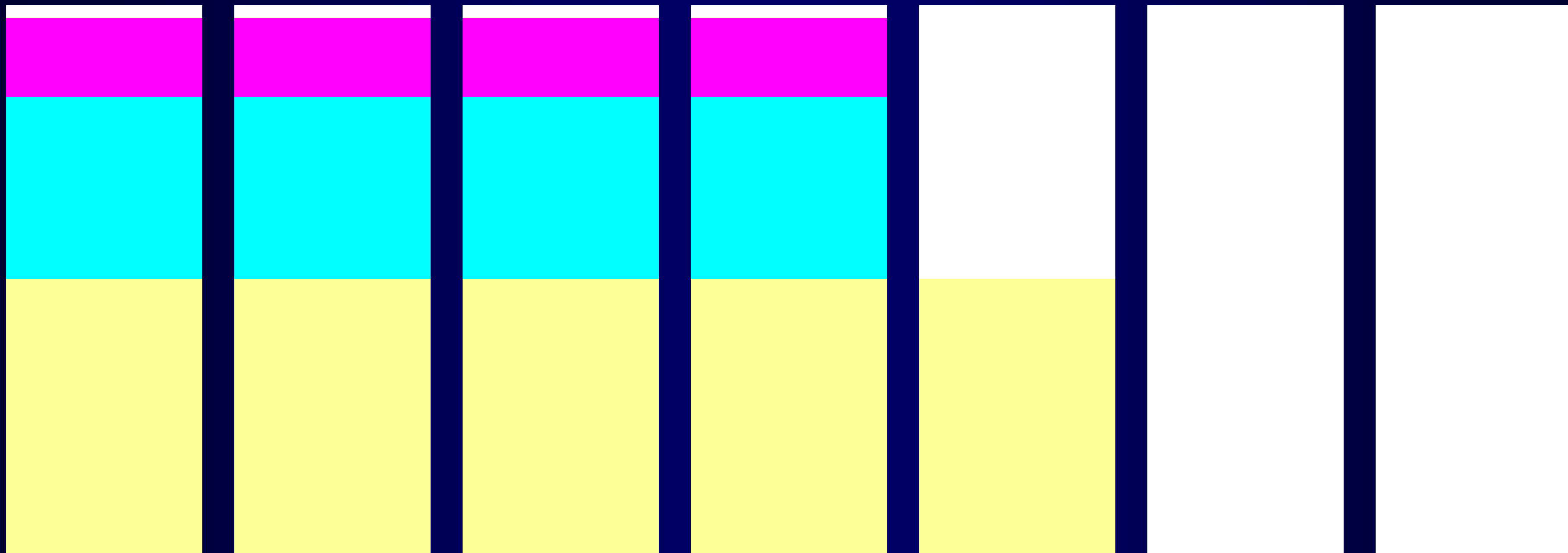
FF (exemple)

B = 420 et 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141, 61



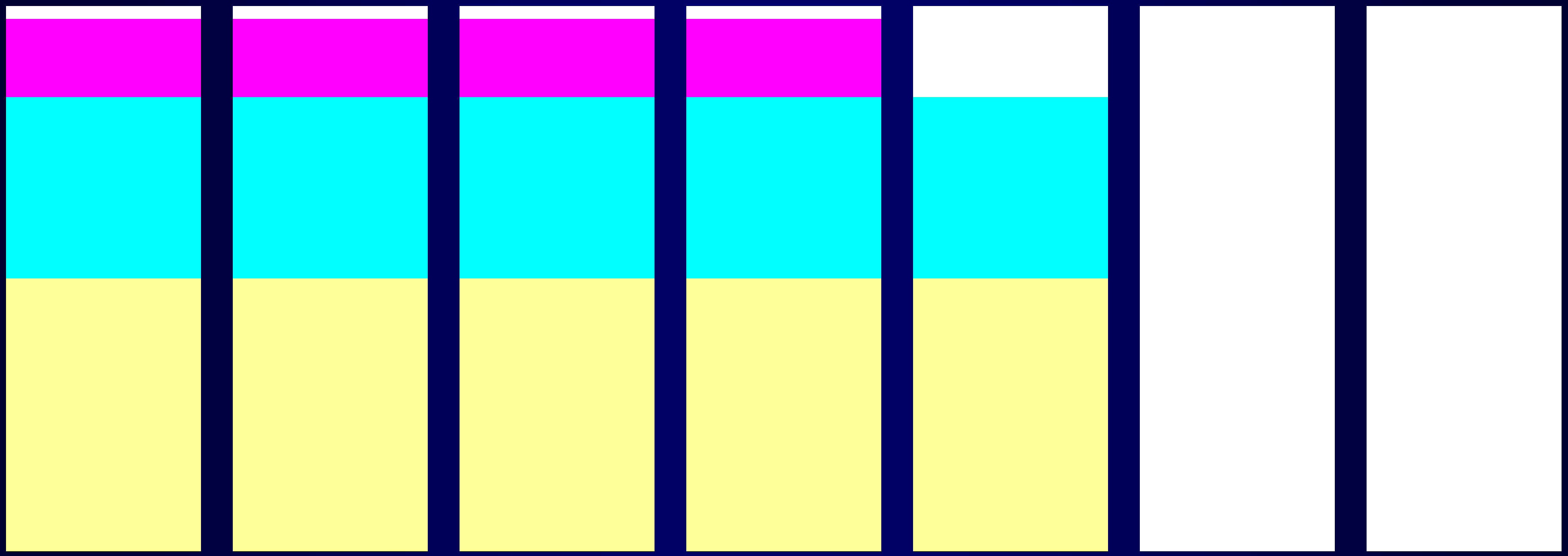
FF (exemple)

B = 420 et 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141, 61, 211



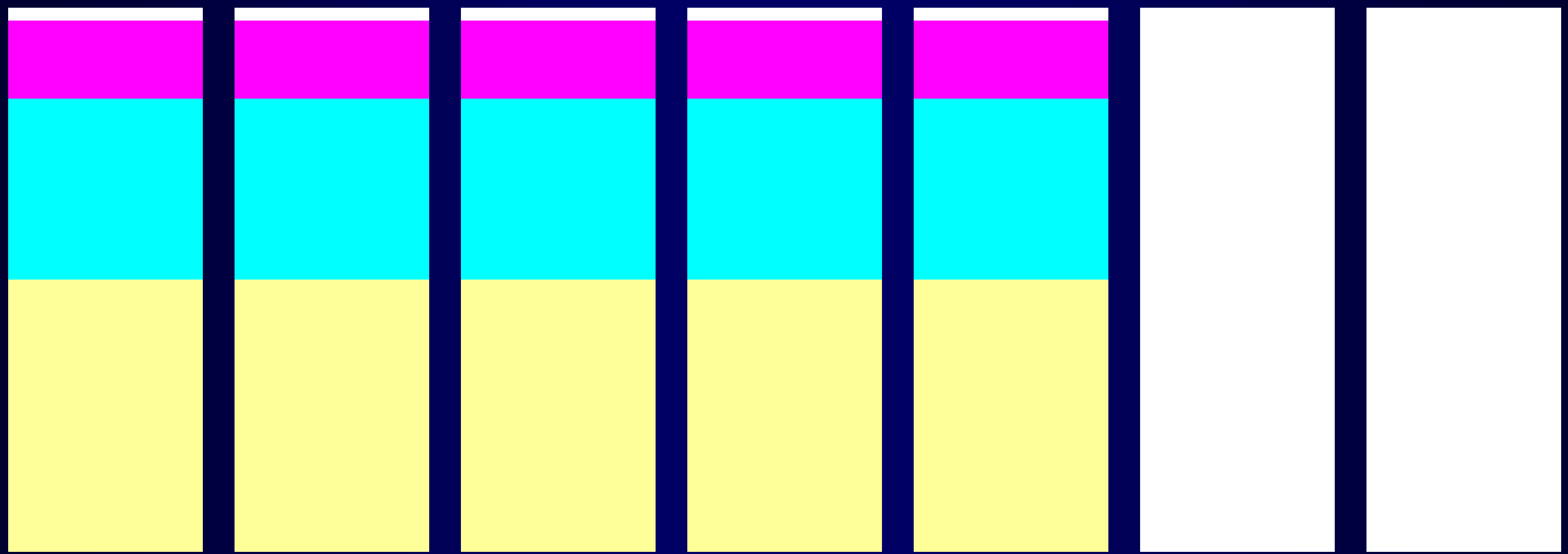
FF (exemple)

B = 420 et 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141



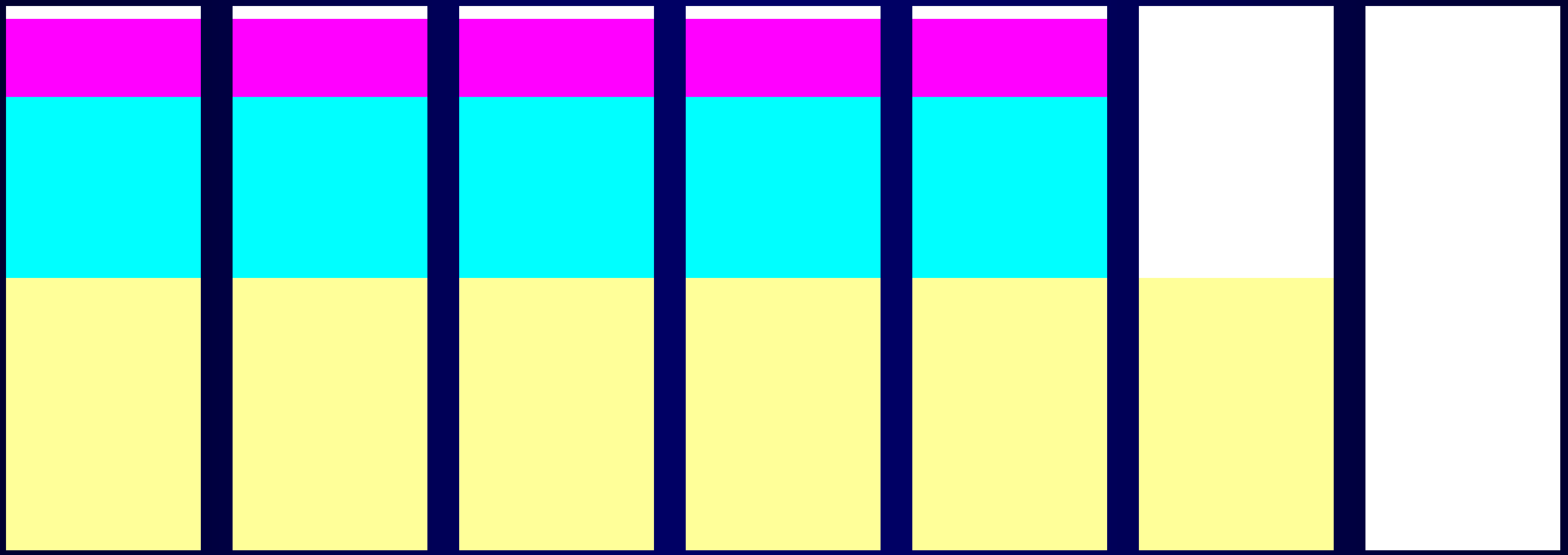
FF (exemple)

B = 420 et 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141, 61



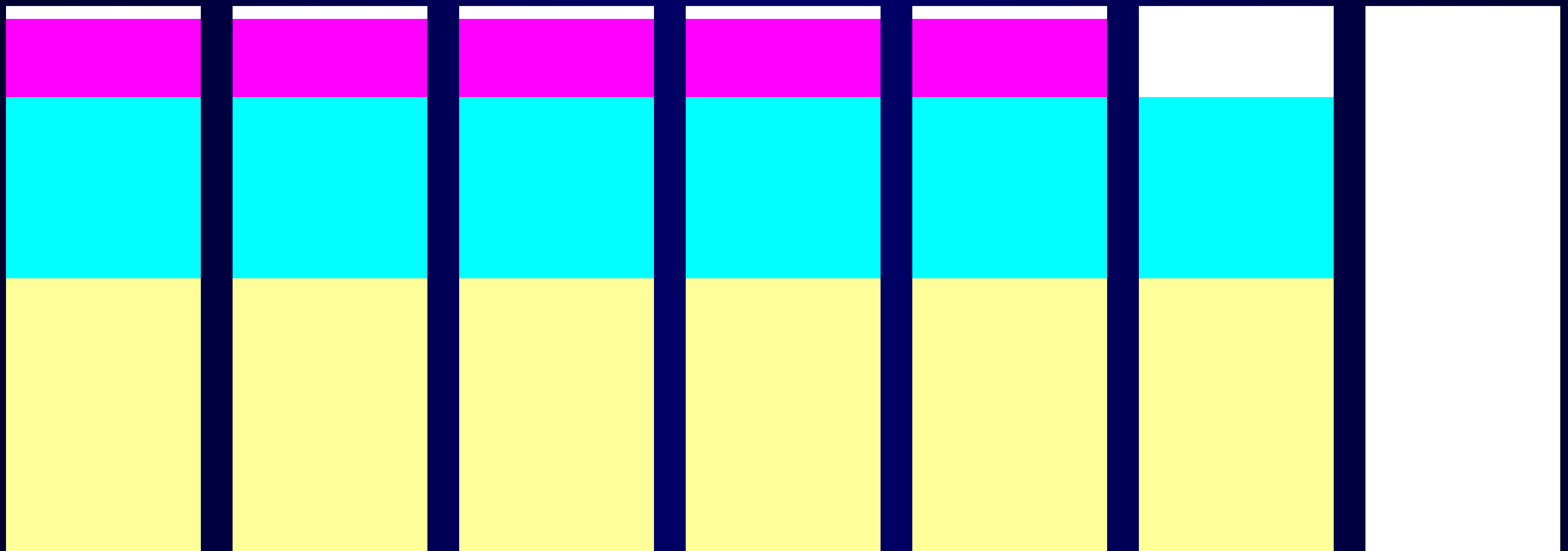
FF (exemple)

B = 420 et 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141, 61, 211



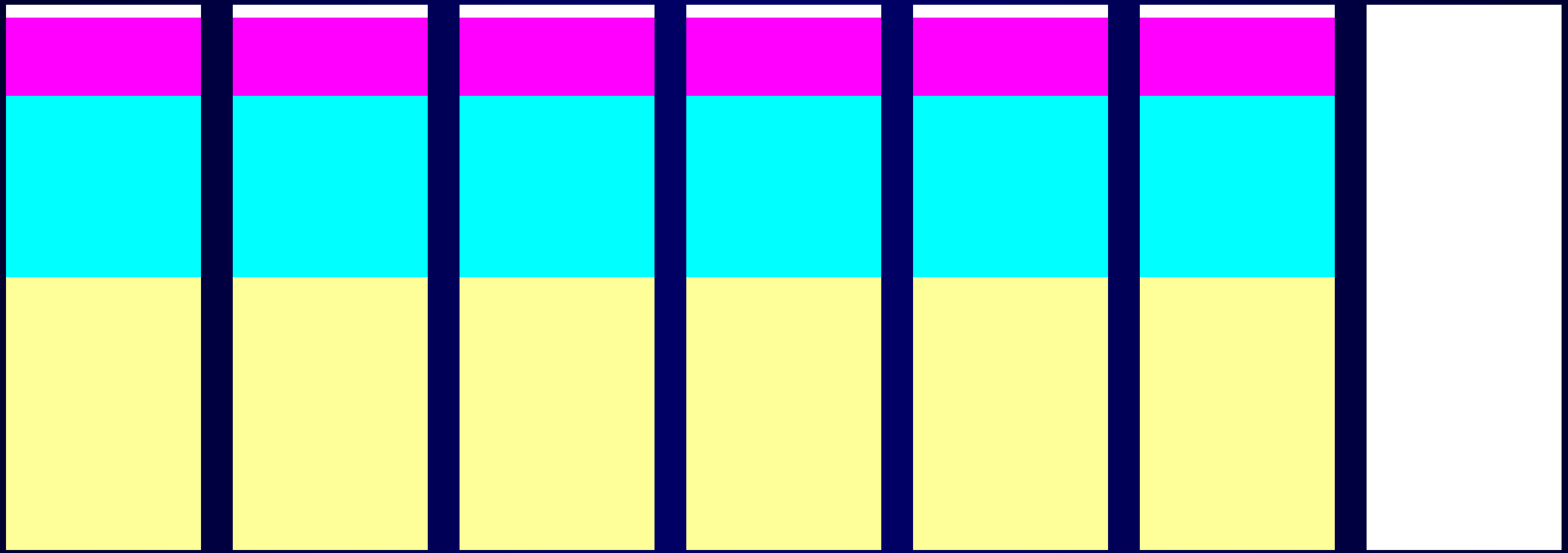
FF (exemple)

B = 420 et 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141



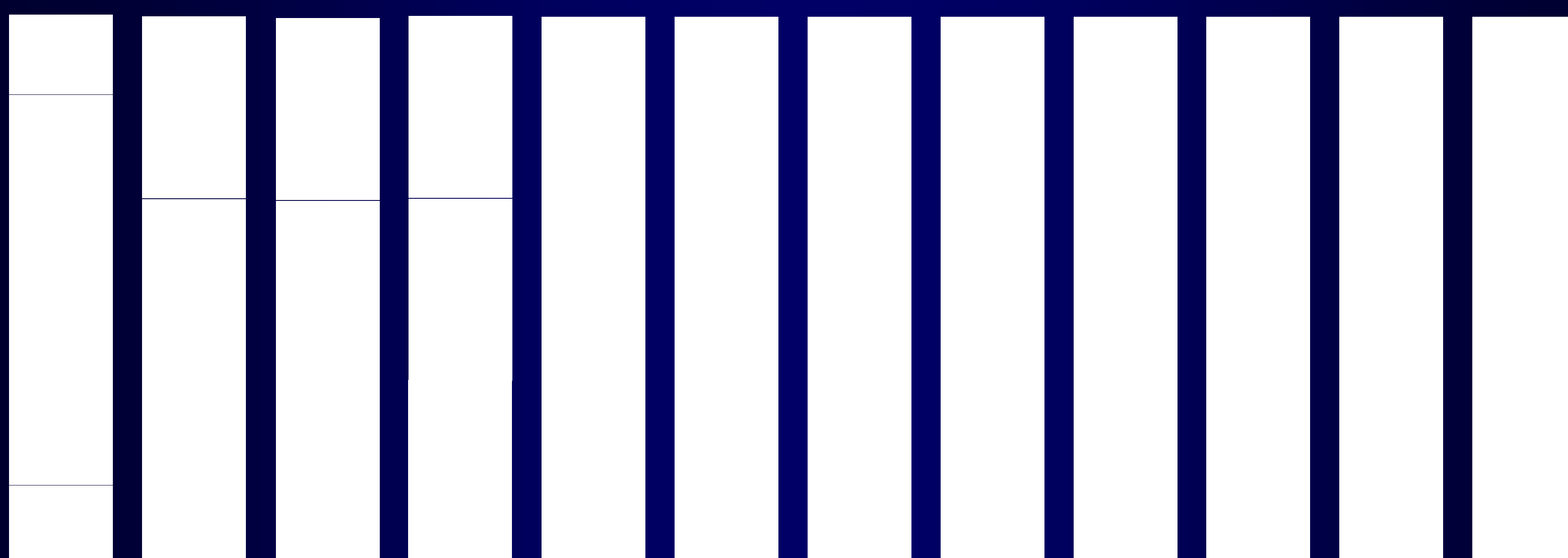
FF (exemple)

B = 420 et 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141, 61, 211, 141, 61.



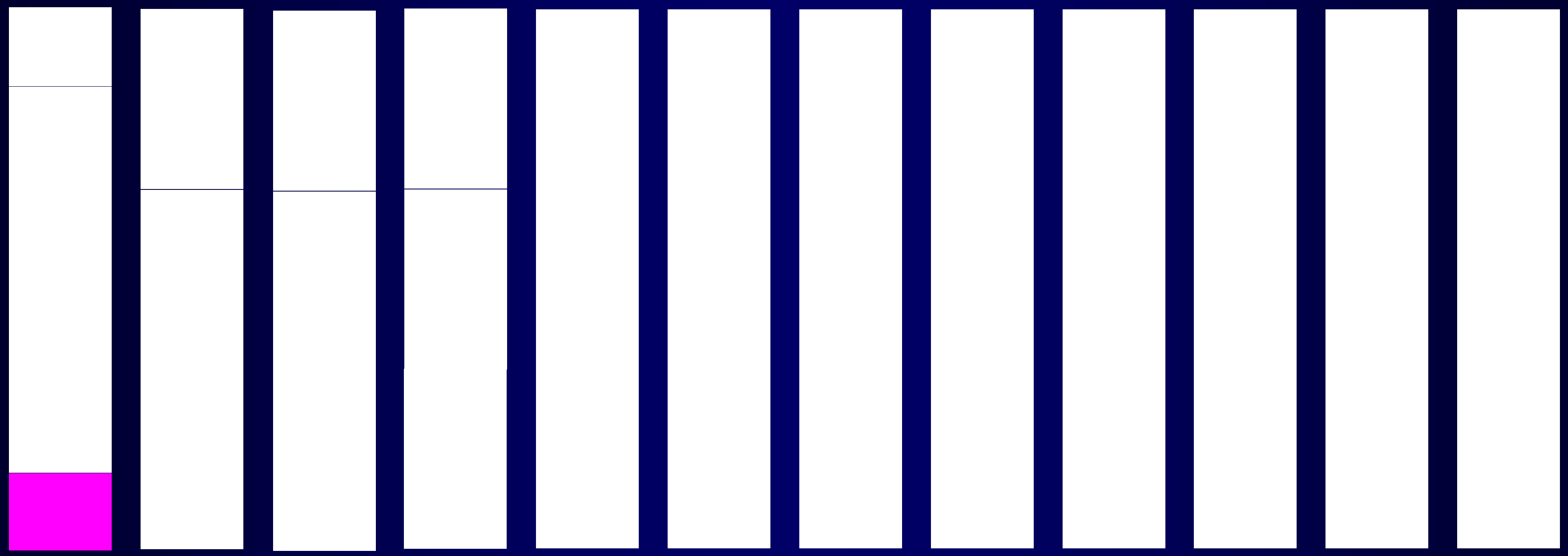
FF (exemple 2)

B = 420 et



FF (exemple 2)

B = 420 et 61

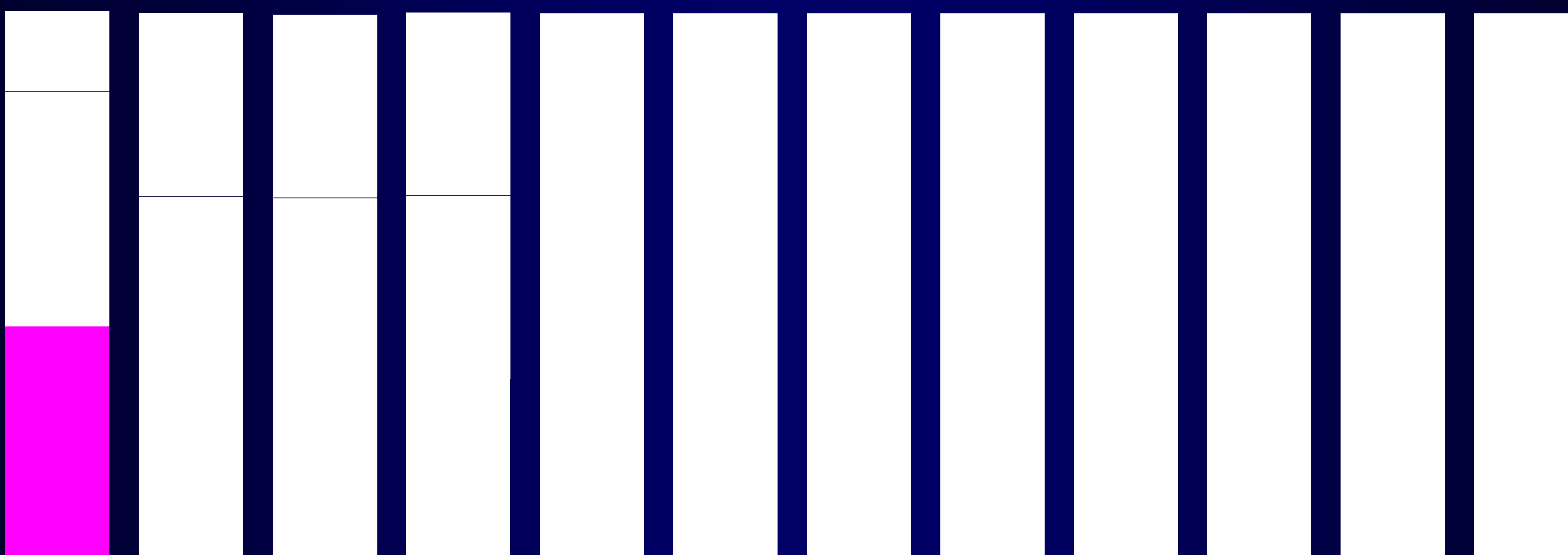


FF (exemple 2)

B = 420 et 61, 61

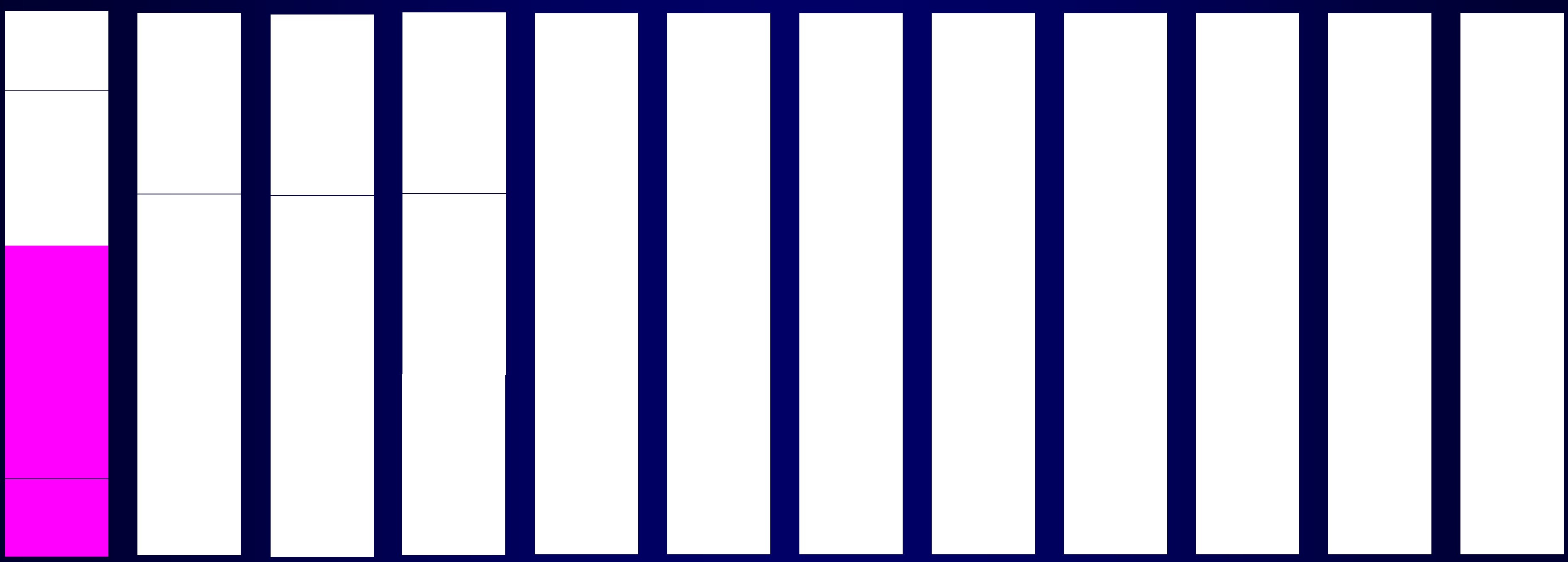
FF (exemple 2)

B = 420 et 61, 61, 61



FF (exemple 2)

B = 420 et 61, 61, 61, 61

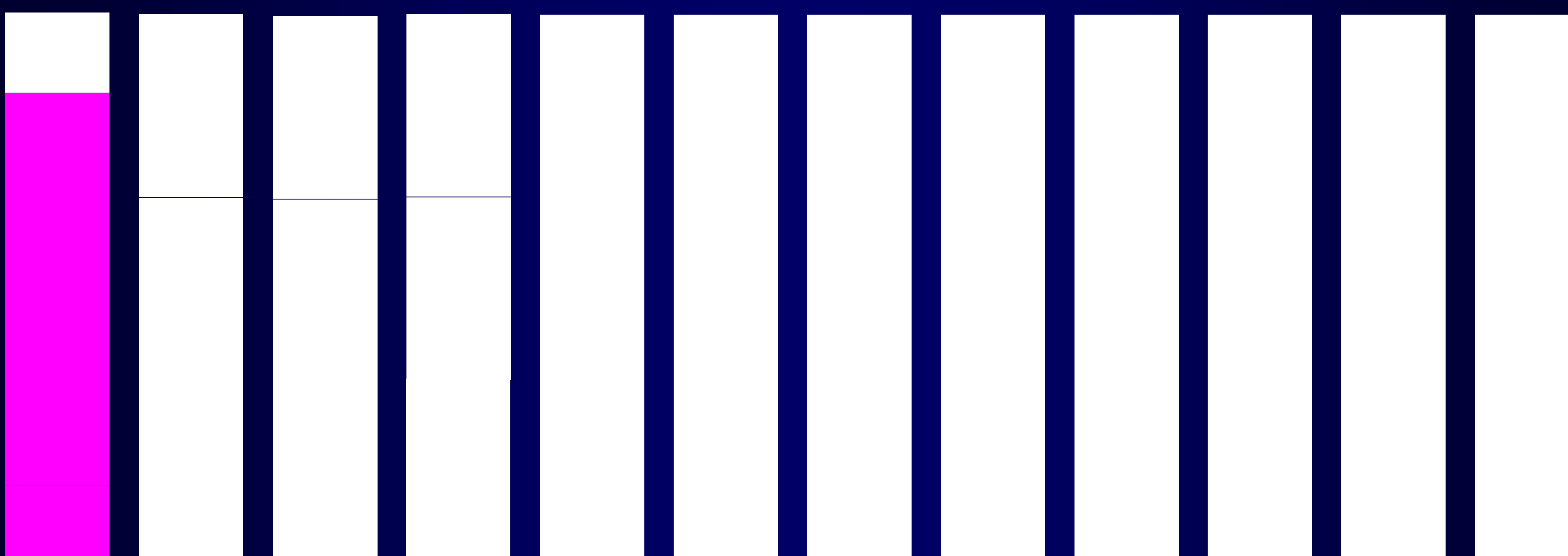


FF (exemple 2)

B = 420 et 61, 61, 61, 61, 61

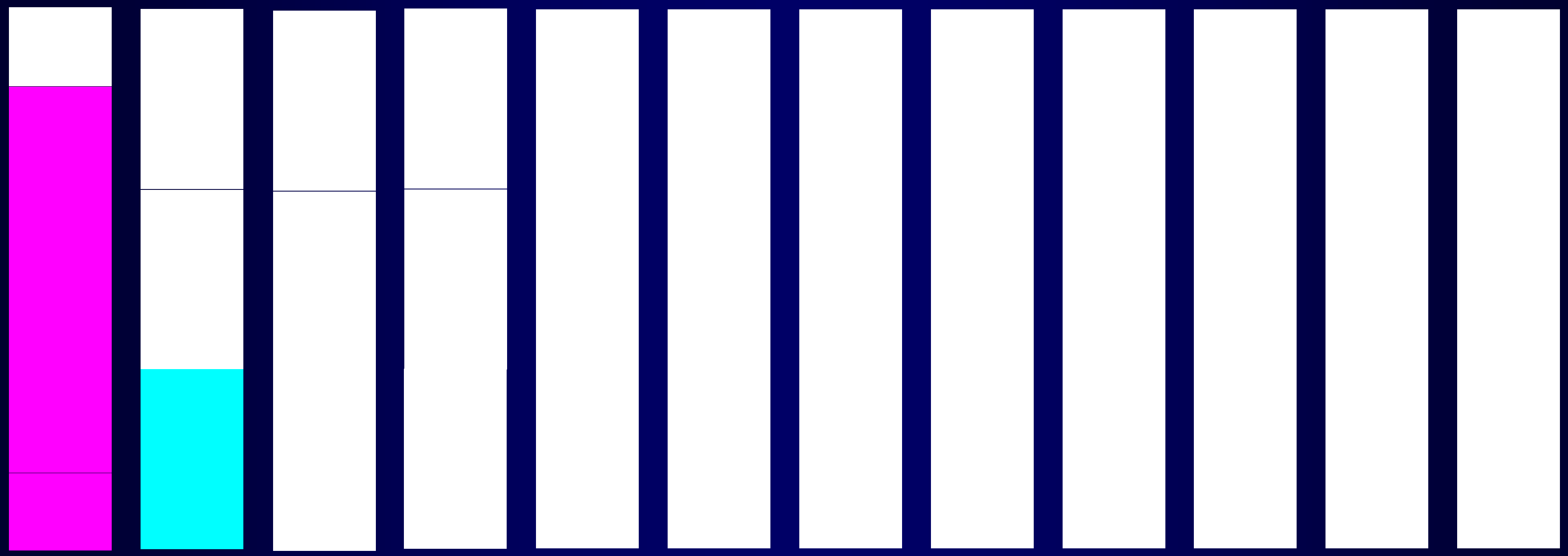
FF (exemple 2)

B = 420 et 61, 61, 61, 61, 61, 61



FF (exemple 2)

B = 420 et 61, 61, 61, 61, 61, 61, 141



FF (exemple 2)

B = 420 et 61, 61, 61, 61, 61, 61, 141, 141

FF (exemple 2)

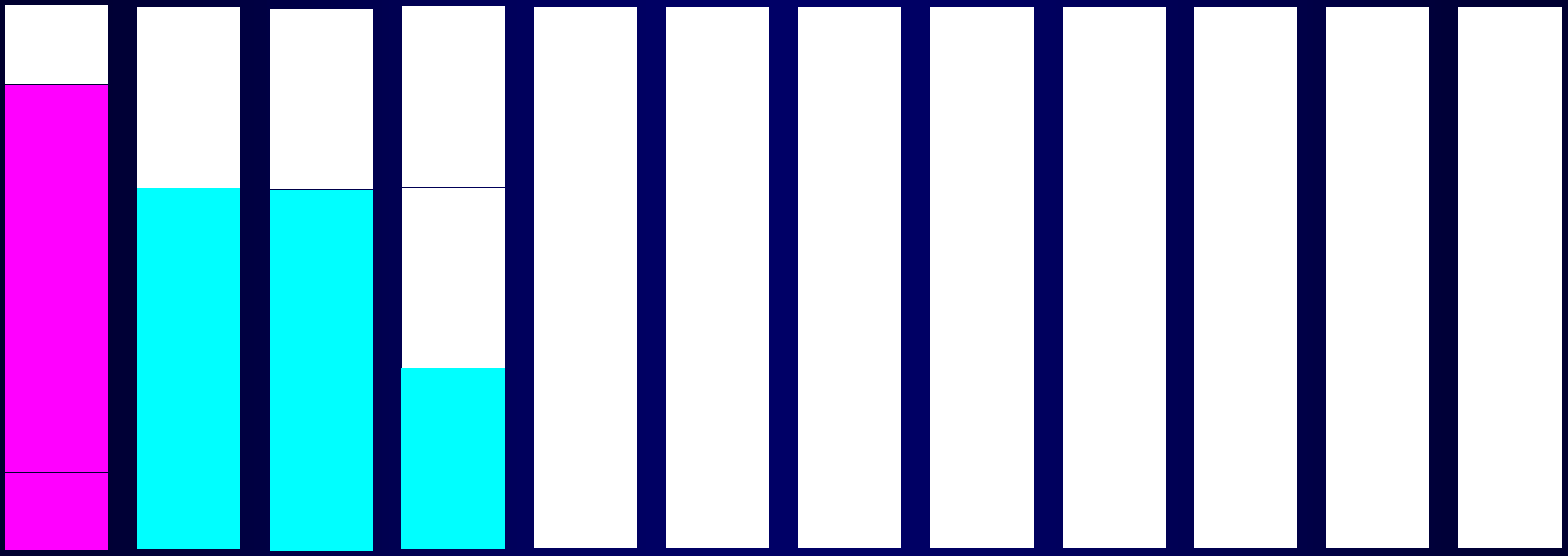
B = 420 et 61, 61, 61, 61, 61, 61, 141, 141, 141

FF (exemple 2)

B = 420 et 61, 61, 61, 61, 61, 61, 141, 141, 141, 141

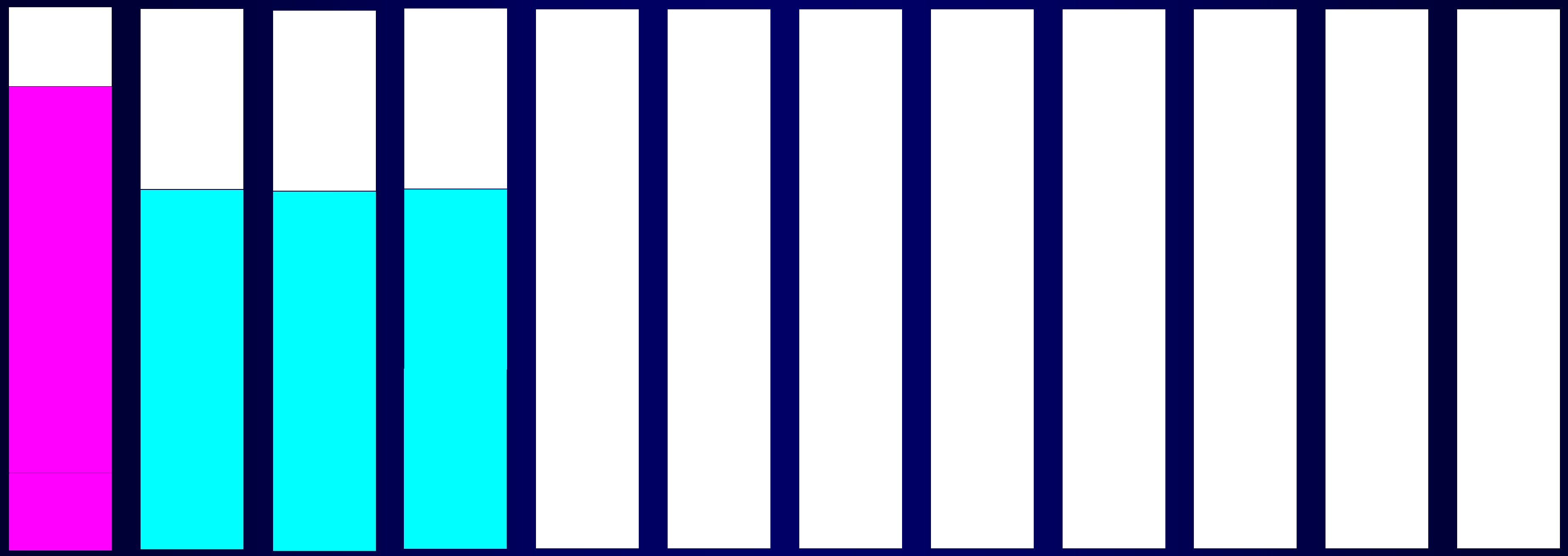
FF (exemple 2)

B = 420 et 61, 61, 61, 61, 61, 61, 141, 141, 141, 141, 141



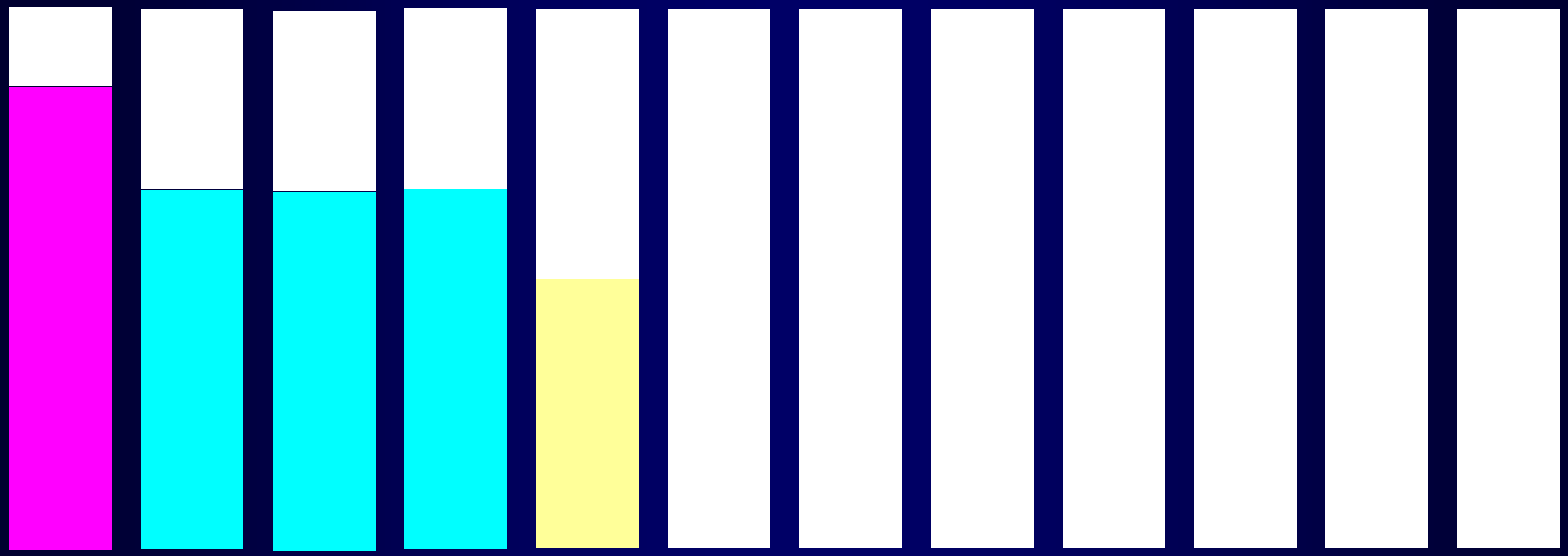
FF (exemple 2)

B = 420 et 61, 61, 61, 61, 61, 61, 141, 141, 141, 141, 141, 141



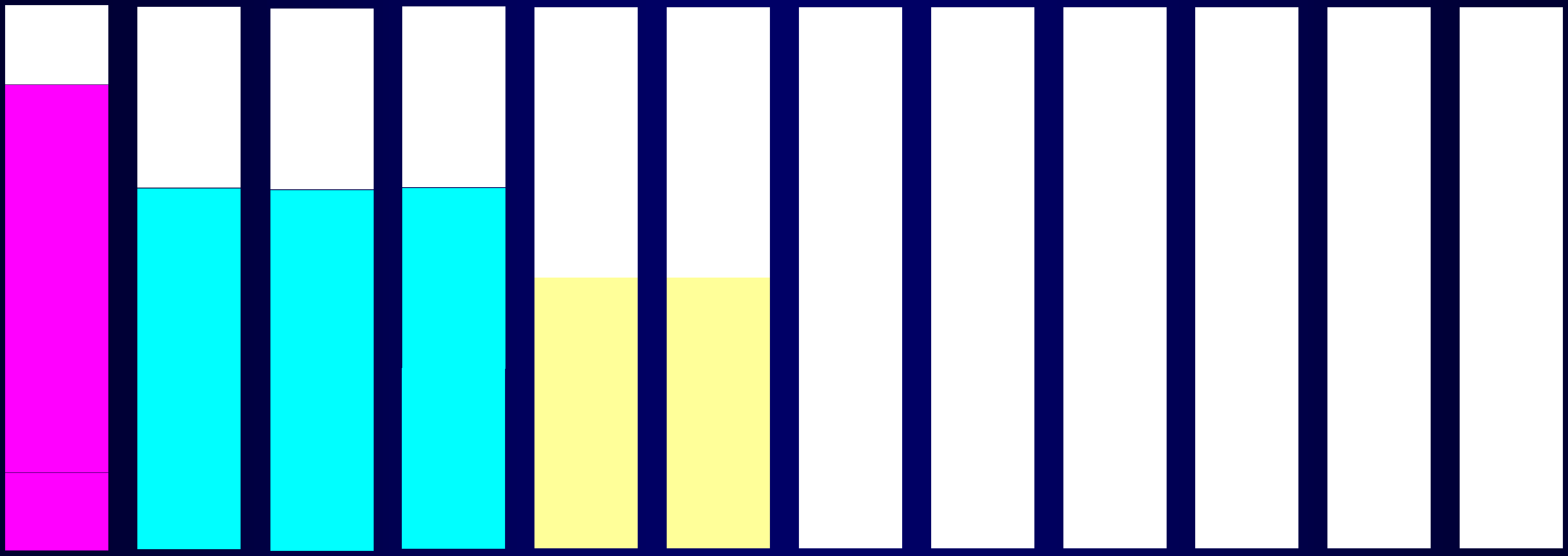
FF (exemple 2)

B = 420 et 61, 61, 61, 61, 61, 61, 141, 141, 141, 141, 141, 141, 211



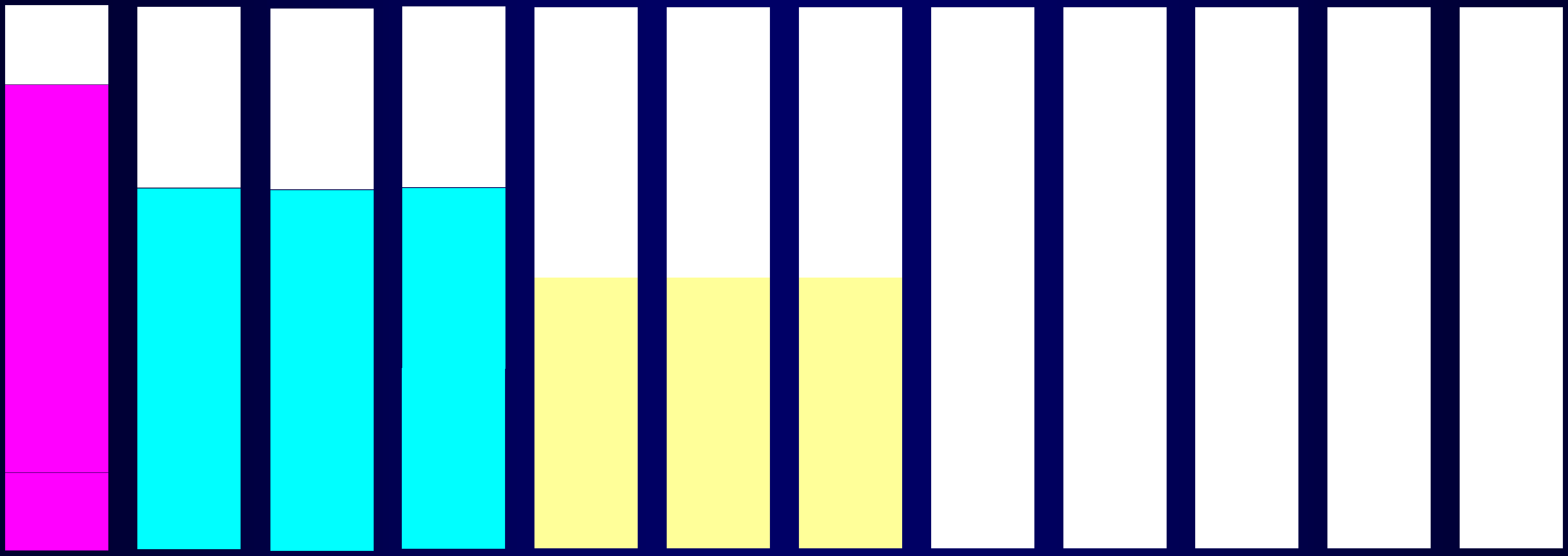
FF (exemple 2)

B = 420 et 61, 61, 61, 61, 61, 61, 141, 141, 141, 141, 141, 141, 211, 211,



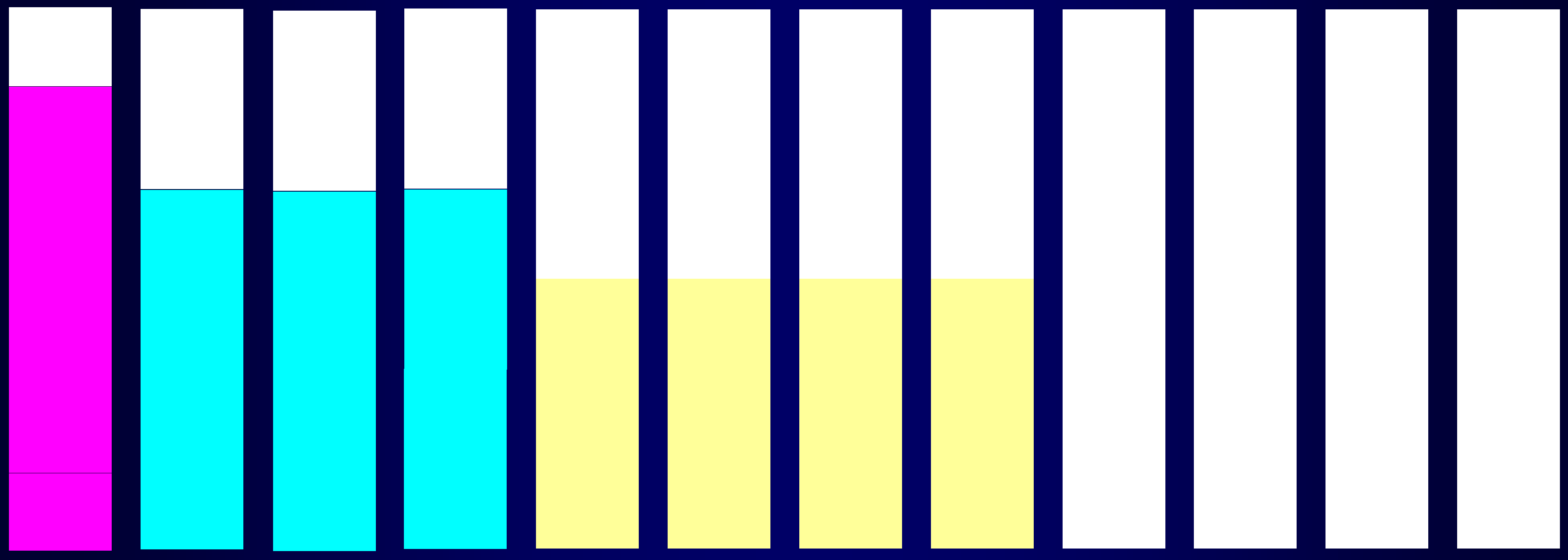
FF (exemple 2)

B = 420 et 61, 61, 61, 61, 61, 61, 141, 141, 141, 141, 141, 141, 211, 211, 211



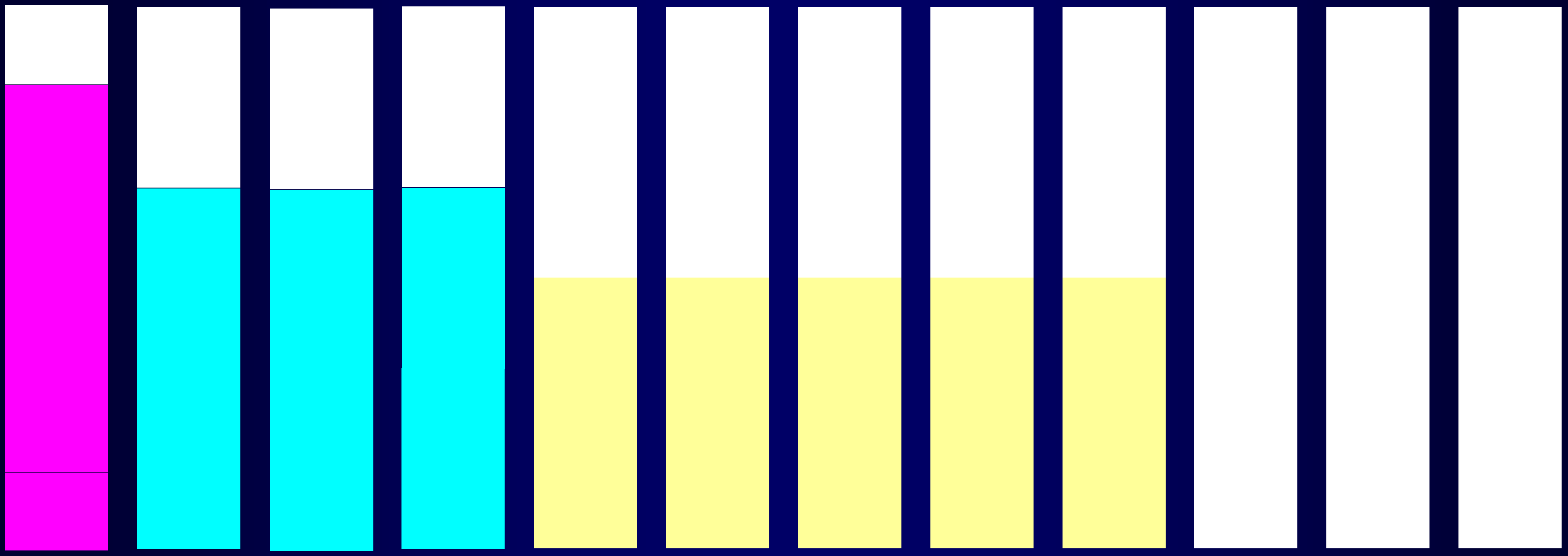
FF (exemple 2)

B = 420 et 61, 61, 61, 61, 61, 61, 141, 141, 141, 141, 141, 141, 211, 211, 211, 211,



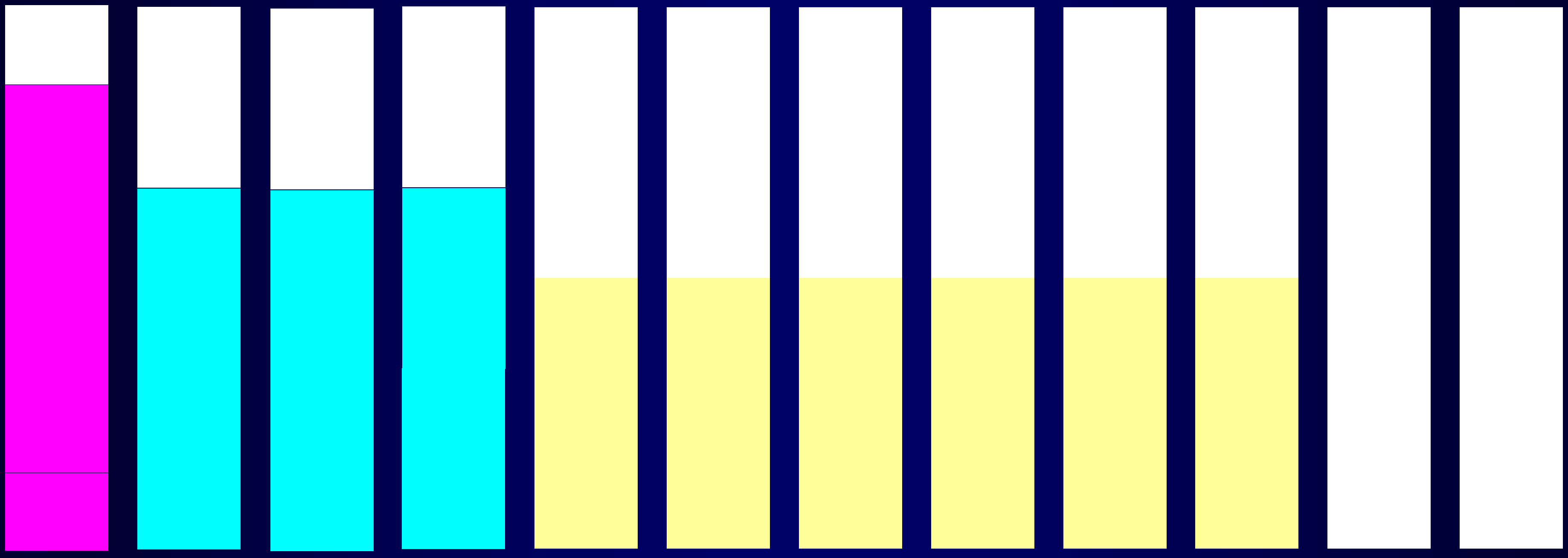
FF (exemple 2)

B = 420 et 61, 61, 61, 61, 61, 61, 141, 141, 141, 141, 141, 141, 211, 211, 211, 211, 211



FF (exemple 2)

B = 420 et 61, 61, 61, 61, 61, 61, 141, 141, 141, 141, 141, 141, 211, 211, 211, 211, 211, 211.



FF (suite)

On peut remarquer que pour les mêmes données, on a utilisé dans le premier exemple 6 bins et 10 dans le deuxième.

Le problème Bin Packing (2)

NOM : BIN PACKING

DONNEES : ensemble fini d'éléments U (taille $\in (0,1]$).

QUESTION : Quel est le plus petit k tel qu'on peut partitionner U en $U_1, U_2, \ldots, U_k$ sans que la somme des tailles des éléments des U_i dépasse 1 ?

Généralisation des exemples

$n = 18m$ et les valeurs sont :

$$u_i = \begin{cases} 1/7 + \varepsilon & 1 \leq i \leq 6m \\ 1/3 + \varepsilon & 6m < i \leq 12m \\ 1/2 + \varepsilon & 12m < i \leq 18m \end{cases}$$

Solution obtenue par FF :

- m bins contenant 6 objets de taille $1/7 + \varepsilon$
- $3m$ bins contenant 2 objets de taille $1/3 + \varepsilon$
- $6m$ bins contenant 1 objet de taille $1/2 + \varepsilon$

Solution obtenue par FF :

- **m bins contenant 6 objets de taille $1/7 + \varepsilon$**
- **$3m$ bins contenant 2 objets de taille $1/3 + \varepsilon$**
- **$6m$ bins contenant 1 objet de taille $1/2 + \varepsilon$**

Donc FF(D) = 10m

Solution optimale :

- **$6m$ bins contenant 3 objets (un de taille $1/7 + \varepsilon$, un de taille $1/3 + \varepsilon$ et un de taille $1/2 + \varepsilon$)**

Donc OPT(D) = 6m

FF (suite)

Propriété : Pour tout D

$$FF(D) < 2OPT(D)$$

Preuve :

On peut remarquer que la somme du contenu du bin 1 et celui du bin 2 est > 1

Et cela est vrai pour bin *i* et bin *i*+1 !

suite

Ainsi nous avons

$b_1 + b_2 > 1$

$b_2 + b_3 > 1$

…

$b_{k\text{-}1} + b_k > 1$

$b_1 + b_k > 1$

donc

$\mathrm{FF(D)} = \mathrm{k} < 2\Sigma_{i=1..n}\, u_i$

Par ailleurs,

$\Sigma_{i=1..n}\, u_i \leq \mathrm{OPT(D)}$

d'où

$\mathrm{FF(D)} < 2\mathrm{OPT(D)}$

CQFD

meilleures bornes ?

Théorème :

Pour toute donnée D

$$FF(D) < 17OPT(D)/10$$

Il existe des données arbitrairement grandes D, telles que

$$FF(D) > 17\big(OPT(D)-1\big)/10$$

Peut-on faire beaucoup mieux ?

NON

Un résultat négatif

Théorème :

Pour tout ε>0, il n'existe pas d'approximation de ratio 3/2-ε pour le problème de BIN PACKING, sauf si P=NP.

On prouve :

Pour tout ε>0, le problème de l'approximation de ratio 3/2-ε pour le problème de BIN PACKING, est NP-difficile.

la preuve

Preuve :

On réduit PARTITION

Pour ce faire on prend comme taille des bin, 1/2S

Ainsi OPT(D) sera 2, et tout approximation de ratio 3/2-ε comprendra <u>moins</u> de 3 bins (donc 2)

ce qui revient à 2 bins.

CQFD

L'heuristique FFD

FFD = First-Fit-Decreasing

- on trie les objets en ordre décroissant
- on applique FF

Complexité : O(n log n + nk)

Bornes pour FFD

Théorème :

Pour toute donnée D

$$FFD(D) < 11OPT(D)/9 + 4$$

Il existe des données arbitrairement grandes D, telles que

$$FF(D) > 11OPT(D)/9$$

Schéma d'approximation

Pour toute valeur $\varepsilon > 0$ elle propose une approximation avec erreur rélative ε.

PTAS (Polynomial time approximation scheme)

Schéma polynomial d'approximation : pour tout $\varepsilon > 0$ le schéma est polynomiale en n (taille du problème).

Schéma d'approximation

FPTAS

(Fully polynomial time approximation scheme)

Schéma polynomial d'approximation : si le schéma est polynomiale en n (taille du problème) et $1/\varepsilon$.

APTAS

Asymptotic polynomial-time approximation scheme

- ***k* donné. Pour tout $\varepsilon > 0$ il existe un algorithme A_ε tel que**

$$A_\varepsilon(D) \leq (1+\varepsilon)\ OPT(D) + k.$$

L'algorithme A_ε est polynomial en $|D|$.

pour le cas d'un pb. de maximisation :

$$A_\varepsilon(D) \geq (1-\varepsilon)\ OPT(D) - k$$

Un résultat pour le BIN PACKING

Théorème :

Pour tout ε, $0 \leq \varepsilon \leq ½$ il existe un algorithme polynomiale A_ε qui trouve une solution utilisant au plus $(1+\varepsilon)$ OPT(D) + 1 bins.

La hiérarchie des classes de complexité