Ensemble $\mathcal{C}_n^k$ des parties à k éléments de l'ensemble $\mathcal{E}_n = \{1,\ 2 \ldots n\}$.

partie = sous-ensemble

$\mathcal{E}_4 = \{ \quad , \quad , \quad , \quad \}$

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

C_4^2

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\quad ,\quad ,\quad ,\quad ,\quad ,\quad \}$

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,\},\qquad\qquad ,\qquad\qquad ,\qquad\qquad ,\qquad\qquad ,\qquad\qquad \}$

C_4^2

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,1\},\qquad\qquad ,\qquad\qquad ,\qquad\qquad ,\qquad\qquad ,\qquad\qquad \}$

C_4^2

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,1\},\qquad ,\qquad ,\qquad ,\qquad ,\qquad \}$

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\qquad\qquad ,\qquad\qquad ,\qquad\qquad ,\qquad\qquad ,\qquad\qquad \}$

C_4^2

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\quad \{1,3\},\quad \{1,4\},\qquad ,\qquad ,\qquad \}$

C_4^2

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\quad \{1,\ 3\},\quad \{1,\ 4\},\quad \{2,\},\qquad\qquad ,\qquad\qquad \}$

C_4^2

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\quad \{1,\ 3\},\quad \{1,\ 4\},\quad \{2,\ 1\},\qquad\qquad ,\qquad\qquad \}$

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\quad \{1,\ 3\},\quad \{1,\ 4\},\quad \{2,\ 4\},\qquad ,\qquad \}$

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\quad \{1,\ 3\},\quad \{1,\ 4\},\quad \{2,\ 3\},\qquad\qquad ,\qquad\qquad \}$

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\quad \{1,\ 3\},\quad \{1,\ 4\},\quad \{2,\ 3\},\quad \{2,\ 4\},\quad \{3,\ 4\}\ \}$

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\quad \{1,\ 3\},\quad \{1,\ 4\},\quad \{2,\ 3\},\quad \{2,\ 4\},\quad \{3,\ 4\}\ \}$

La complexité de l'algorithme naïf qui construit toutes les parties à **deux** éléments d'un ensemble à n éléments est dans

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\quad \{1,\ 3\},\quad \{1,\ 4\},\quad \{2,\ 3\},\quad \{2,\ 4\},\quad \{3,\ 4\}\ \}$

La complexité de l'algorithme naïf qui construit toutes les parties à **deux** éléments d'un ensemble à n éléments est dans $\Theta(n^2)$

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\quad \{1,\ 3\},\quad \{1,\ 4\},\quad \{2,\ 3\},\quad \{2,\ 4\},\quad \{3,\ 4\}\ \}$

La complexité de l'algorithme naïf qui construit toutes les parties à **trois** éléments d'un ensemble à n éléments est dans $\Theta()$

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\quad \{1,\ 3\},\quad \{1,\ 4\},\quad \{2,\ 3\},\quad \{2,\ 4\},\quad \{3,\ 4\}\ \}$

La complexité de l'algorithme naïf qui construit toutes les parties à **trois** éléments d'un ensemble à n éléments est dans $\Theta(n^3)$

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\quad \{1,\ 3\},\quad \{1,\ 4\},\quad \{2,\ 3\},\quad \{2,\ 4\},\quad \{3,\ 4\}\ \}$

La complexité de l'algorithme naïf qui construit toutes les parties à **k** éléments d'un ensemble à n éléments est dans $\Theta()$

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\quad \{1,\ 3\},\quad \{1,\ 4\},\quad \{2,\ 3\},\quad \{2,\ 4\},\quad \{3,\ 4\}\ \}$

La complexité de l'algorithme naïf qui construit toutes les parties à **k** éléments d'un ensemble à n éléments est dans $\Theta(n^k)$

$\mathcal{E}_4 = \{\ 1,\quad 2,\quad 3,\quad 4\}$

$\mathcal{C}_4^2 = \{\ \{1,2\},\quad \{1,\ 3\},\quad \{1,\ 4\},\quad \{2,\ 3\},\quad \{2,\ 4\},\quad \{3,\ 4\}\ \}$

La complexité de l'algorithme naïf qui **imprime** toutes les parties à **k** éléments d'un ensemble à n éléments est dans $\Theta(n^k \times k)$

Peut-on faire mieux ?

Quel est le coût **minimum** de l'algorithme d'impression ?

Peut-on faire mieux ?

Quel est le coût **minimum** de l'algorithme d'impression ?

Coût d'impression d'un élément

$\times$

Nombre d'éléments

Peut-on faire mieux ?

Quel est le coût **minimum** de l'algorithme d'impression ?

Coût d'impression d'un élément $\Theta(k)$

$\times$

Nombre d'éléments

Peut-on faire mieux ?

Quel est le coût **minimum** de l'algorithme d'impression ?

Coût d'impression d'un élément $\Theta(k)$

$\times$

Nombre d'éléments C_n^k

Peut-on faire mieux ?

Quel est le coût **minimum** de l'algorithme d'impression ?

Coût d'impression d'un élément $\Theta(k)$

$\times$

Nombre d'éléments C_n^k

$\Theta(k.C_n^k)$

Peut-on faire mieux ?

Quel est le coût **minimum** de l'algorithme d'impression ?

Coût d'impression d'un élément $\Theta(k)$

$\times$

Nombre d'éléments C_n^k

$$\Theta(k.C_n^k)$$

L'algorithme naïf est en $\Theta(k \times n^k)$, le coût minimum est en $\frac{n!}{(k-1)!(n-k)!}$

$\frac{n!}{(n-k)!} =$

$$\frac{n!}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{(n-k)!} =$$

$$\frac{n!}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{1\times2\times\ldots\times(n-k)}$$

$$\frac{n!}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{1\times2\times\ldots\times(n-k)}$$

$$= \frac{\cancel{1\times2\times\ldots\times(n-k)}\times(n-k+1)\times\ldots\times n}{\cancel{1\times2\times\ldots\times(n-k)}} =$$

$$\frac{n!}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{1\times2\times\ldots\times(n-k)}$$

$$= \frac{\cancel{1\times2\times\ldots\times(n-k)}\times(n-k+1)\times\ldots\times n}{\cancel{1\times2\times\ldots\times(n-k)}} = \prod_{i=0}^{i=k-1}(n-i)$$

$$\frac{n!}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{1\times2\times\ldots\times(n-k)}$$

$$= \frac{\cancel{1\times2\times\ldots\times(n-k)}\times(n-k+1)\times\ldots\times n}{\cancel{1\times2\times\ldots\times(n-k)}} = \Pi_{i=0}^{i=k-1}(n-i)$$

n^k est beaucoup plus grand que $\Pi_{i=0}^{i=k-1}(n-i)$

$$\frac{n!}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{1\times2\times\ldots\times(n-k)}$$

$$= \frac{\cancel{1\times2\times\ldots\times(n-k)}\times(n-k+1)\times\ldots\times n}{\cancel{1\times2\times\ldots\times(n-k)}} = \Pi_{i=0}^{i=k-1}(n-i)$$

n^k est beaucoup plus grand que $\Pi_{i=0}^{i=k-1}(n-i)$

Donc le rapport du coût de l'algorithme naïf au coût minimal

$$\frac{n!}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{1\times2\times\ldots\times(n-k)}$$

$$= \frac{\cancel{1\times2\times\ldots\times(n-k)}\times(n-k+1)\times\ldots\times n}{\cancel{1\times2\times\ldots\times(n-k)}} = \Pi_{i=0}^{i=k-1}(n-i)$$

n^k est beaucoup plus grand que $\Pi_{i=0}^{i=k-1}(n-i)$

Donc le rapport du coût de l'algorithme naïf au coût minimal

$$\frac{k\times n^k}{\frac{\Pi_{i=0}^{i=k-1}(n-i)}{(k-1)!}} =$$

$$\frac{n!}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{1\times2\times\ldots\times(n-k)}$$

$$= \frac{\cancel{1\times2\times\ldots\times(n-k)}\times(n-k+1)\times\ldots\times n}{\cancel{1\times2\times\ldots\times(n-k)}} = \Pi_{i=0}^{i=k-1}(n-i)$$

n^k est beaucoup plus grand que $\Pi_{i=0}^{i=k-1}(n-i)$

Donc le rapport du coût de l'algorithme naïf au coût minimal $\frac{k\times n^k}{\frac{\Pi_{i=0}^{i=k-1}(n-i)}{(k-1)!}} = \frac{n^k}{\Pi_{i=0}^{i=k-1}(n-i)} \times k!$ est supèrieur à $k!$

$$\frac{n!}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{(n-k)!} = \frac{1\times2\times\ldots\times(n-k)\times(n-k+1)\times\ldots\times n}{1\times2\times\ldots\times(n-k)}$$

$$= \frac{\cancel{1\times2\times\ldots\times(n-k)}\times(n-k+1)\times\ldots\times n}{\cancel{1\times2\times\ldots\times(n-k)}} = \Pi_{i=0}^{i=k-1}(n-i)$$

n^k est beaucoup plus grand que $\Pi_{i=0}^{i=k-1}(n-i)$

Donc le rapport du coût de l'algorithme naïf au coût minimal $\frac{k\times n^k}{\frac{\Pi_{i=0}^{i=k-1}(n-i)}{(k-1)!}} = \frac{n^k}{\Pi_{i=0}^{i=k-1}(n-i)} \times k!$ est supèrieur à $k!$

Ça vaut le coup de tenter quelque chose par exemple quand $k \approx \frac{n}{2}$

$\mathcal{E}_5 = \{1,\ 2,\ 3,\ 4,\ 5\}$

$\mathcal{S}_3^5$

$\mathcal{E}_5 = \{1,\ 2,\ 3,\ 4,\ 5\}$

$\mathcal{C}_3^5 = \{$ $\}$

$\mathcal{E}_5 = \{1,\ 2,\ 3,\ 4,\ 5\}$

$\mathcal{C}_3^5 = \{(\ ,\ ,\ ),$ $\}$

$\mathcal{E}_5 = \{1,\ 2,\ 3,\ 4,\ 5\}$

$\mathcal{C}_3^5 = \{(1,2,3),$ $\}$

$$\mathcal{E}_5 = \{1,\ 2,\ 3,\ 4,\ 5\}$$

$$\mathcal{C}_3^5 = \{(1{,}2{,}3),(\ ,\ ,\ )\qquad\qquad\qquad\qquad\qquad\qquad\}$$

$$\mathcal{E}_5 = \{1,\ 2,\ 3,\ 4,\ 5\}$$

$$\mathcal{C}_3^5 = \{(1{,}2{,}3),(1{,}2,\ )\qquad\qquad\}$$

$\mathcal{E}_5 = \{1,\ 2,\ 3,\ 4,\ 5\}$

$\mathcal{C}_3^5 = \{(1,2,3),(1,2,4) \qquad \}$

$\mathcal{E}_5 = \{1,\ 2,\ 3,\ 4,\ 5\}$

$\mathcal{C}_3^5 = \{(1{,}2{,}3),(1{,}2{,}4)(1{,}2{,}5), \qquad \}$

$\mathcal{S}_3^5$

$\mathcal{E}_5 = \{1,\ 2,\ 3,\ 4,\ 5\}$

$\mathcal{C}_3^5 = \{(1,2,3),(1,2,4)(1,2,5),(1,\ ,\ )$ $\}$

$\mathcal{E}_5 = \{1,\ 2,\ 3,\ 4,\ 5\}$

$\mathcal{C}_3^5 = \{(1,2,3),(1,2,4)(1,2,5),(1,3,4)(1,3,5),(2,3,4)(2,3,5)(3,4,5)\}$

$\mathcal{E}_5 = \{1,\ 2,\ 3,\ 4,\ 5\}$

$\mathcal{C}_3^5 = \{(1,2,3),(1,2,4)(1,2,5),(1,3,4)(1,3,5),(2,3,4)(2,3,5)(3,4,5)\}$

On construit naturellement $\mathcal{S}_3^5$ quand on veut construire $\mathcal{C}_3^5$

Algorithme *Dernier*

Données:

- deux entiers $k \leq n$
- T un tableau de k éléments

Résultat: Un booléen qui indique si ce tableau représente la *dernière* (au sens de l'ordre lexicographique) des suites ordonnées de k éléments sur l'ensemble $\{1,\ 2, \ldots n\}$

Complexité :

Algorithme *Dernier*

Données:

- deux entiers $k \leq n$
- T un tableau de k éléments

Résultat: Un booléen qui indique si ce tableau représente la *dernière* (au sens de l'ordre lexicographique) des suites ordonnées de k éléments sur l'ensemble $\{1,\ 2, \ldots n\}$

Complexité : $\Theta(\ \ )$

Algorithme *Dernier*

Données:

- deux entiers $k \leq n$
- T un tableau de k éléments

Résultat: Un booléen qui indique si ce tableau représente la *dernière* (au sens de l'ordre lexicographique) des suites ordonnées de k éléments sur l'ensemble $\{1,\ 2, \ldots n\}$

Complexité : $\Theta(k)$

Quelques exemples de *Suivant*

Quel est (avec $n = 8$) la suite suivant la suite

$$[1\ 2\ 3\ 4\ 5\ ]$$

Quelques exemples de *Suivant*

Quel est (avec $n = 8$) la suite suivant la suite

$[1\ 2\ 3\ 4\ 5\ ]$ $[1\ 2\ 3\ 4\ 6\ ]$

Quelques exemples de *Suivant*

Quel est (avec $n = 8$) la suite suivant la suite

[1 2 3 4 5] [1 2 3 4 6]

[1 5 6 7 8]

Quelques exemples de *Suivant*

Quel est (avec $n = 8$) la suite suivant la suite

[1 2 3 4 5]	[1 2 3 4 6]
[1 5 6 7 8]	[2 3 4 5 6]

Quelques exemples de *Suivant*

Quel est (avec $n = 8$) la suite suivant la suite

[1 2 3 4 5]
[1 5 6 7 8]
[1 2 3 4 8]

[1 2 3 4 6]
[2 3 4 5 6]

Quelques exemples de *Suivant*

Quel est (avec $n = 8$) la suite suivant la suite

[1 2 3 4 5]	[1 2 3 4 6]
[1 5 6 7 8]	[2 3 4 5 6]
[1 2 3 4 8]	[1 2 3 5 6]

Quelques exemples de *Suivant*

Quel est (avec $n = 8$) la suite suivant la suite

[1 2 3 4 5]	[1 2 3 4 6]
[1 5 6 7 8]	[2 3 4 5 6]
[1 2 3 4 8]	[1 2 3 5 6]
[1 2 3 7 8]	

Quelques exemples de *Suivant*

Quel est (avec $n = 8$) la suite suivant la suite

[1 2 3 4 5]	[1 2 3 4 6]
[1 5 6 7 8]	[2 3 4 5 6]
[1 2 3 4 8]	[1 2 3 5 6]
[1 2 3 7 8]	[1 2 4 5 6]

Suivant

Données: Un tableau de taille k

Résultat: Un tableau de taille k

Complexité :

Suivant

Données: Un tableau de taille k représentant une suite ordonnée de k entiers de $\{1 \ldots n\}$

Résultat: Un tableau de taille k

Complexité :

Suivant

Données: Un tableau de taille k représentant une suite ordonnée de k entiers de $\{1 \dots n\}$

Résultat: Un tableau de taille k représentant la suite ordonnée suivante

Complexité :

Suivant

Données: Un tableau de taille k représentant une suite ordonnée de k entiers de $\{1 \dots n\}$

Résultat: Un tableau de taille k représentant la suite ordonnée suivante

Complexité : $\Theta(\ )$

Suivant

Données: Un tableau de taille k représentant une suite ordonnée de k entiers de $\{1 \dots n\}$

Résultat: Un tableau de taille k représentant la suite ordonnée suivante

Complexité : $\Theta(k)$

Suivant

Données: Un tableau de taille k représentant une suite ordonnée de k entiers de $\{1 \dots n\}$

Résultat: Un tableau de taille k représentant la suite ordonnée suivante

Complexité : $\Theta(k)$

Reste à l'écrire

Suivant

Données: Un tableau de taille k représentant une suite ordonnée de k entiers de $\{1 \ldots n\}$

Résultat: Un tableau de taille k représentant la suite ordonnée suivante

Complexité : $\Theta(k)$

Reste à l'écrire ce qui est laissé en exercice.

Suivant

Données: Un tableau de taille k représentant une suite ordonnée de k entiers de $\{1 \ldots n\}$

Résultat: Un tableau de taille k représentant la suite ordonnée suivante

Complexité : $\Theta(k)$

Reste à l'écrire ce qui est laissé en exercice.

Premier :

Suivant

Données: Un tableau de taille k représentant une suite ordonnée de k entiers de $\{1 \dots n\}$

Résultat: Un tableau de taille k représentant la suite ordonnée suivante

Complexité : $\Theta(k)$

Reste à l'écrire ce qui est laissé en exercice.

Premier :

Résultat: Un tableau de taille k représentant la première suite ordonnée

Complexité :

Algorithmes *Suivant* et *Premier*

Suivant

Données: Un tableau de taille k représentant une suite ordonnée de k entiers de $\{1 \dots n\}$

Résultat: Un tableau de taille k représentant la suite ordonnée suivante

Complexité : $\Theta(k)$

Reste à l'écrire ce qui est laissé en exercice.

Premier :

Résultat: Un tableau de taille k représentant la première suite ordonnée

Complexité : $\Theta(\ )$

Suivant

Données: Un tableau de taille k représentant une suite ordonnée de k entiers de $\{1 \dots n\}$

Résultat: Un tableau de taille k représentant la suite ordonnée suivante

Complexité : $\Theta(k)$

Reste à l'écrire ce qui est laissé en exercice.

Premier :

Résultat: Un tableau de taille k représentant la première suite ordonnée

Complexité : $\Theta(k)$

algorithme final

Données: $k \leq n$ deux entiers
Résultat: $\mathcal{C}_n^k$ a été imprimé

$T \leftarrow$ Premier();
Imprimer T;

tant que *non Dernier(T)* **faire**
  $T \leftarrow$ Suivant(T);
  Imprimer(T);
fin

algorithme final

Données: $k \leq n$ deux entiers
Résultat: $\mathcal{C}_n^k$ a été imprimé

$\Theta(k)$ $T \leftarrow$ Premier();
$\Theta(k)$ Imprimer T;

tant que *non Dernier(T)* **faire**
  $T \leftarrow$ Suivant(T);
  Imprimer(T);
fin

algorithme final

Données: $k \leq n$ deux entiers
Résultat: $\mathcal{C}_n^k$ a été imprimé

$\Theta(k)$ $T \leftarrow$ Premier();
$\Theta(k)$ Imprimer T;

tant que *non Dernier(T)* **faire**
  $\Theta(k)$ $T \leftarrow$ Suivant(T);
  $\Theta(k)$ Imprimer(T);
fin

algorithme final

Données: $k \leq n$ deux entiers
Résultat: $\mathcal{C}_n^k$ a été imprimé

$\Theta(k)$ $T \leftarrow$ Premier();
$\Theta(k)$ Imprimer T;

tant que *non Dernier(T)* **faire**
  $\Theta(k)$ $T \leftarrow$ Suivant(T);
  $\Theta(k)$ Imprimer(T);
fin

On passe dans la répétitive fois

algorithme final

Données: $k \leq n$ deux entiers
Résultat: $\mathcal{C}_n^k$ a été imprimé

$\Theta(k)$ $T \leftarrow$ Premier();
$\Theta(k)$ Imprimer T;

tant que *non Dernier(T)* **faire**
  $\Theta(k)$ $T \leftarrow$ Suivant(T);
  $\Theta(k)$ Imprimer(T);
fin

On passe dans la répétitive C_n^k fois

algorithme final

Données: $k \leq n$ deux entiers
Résultat: $\mathcal{C}_n^k$ a été imprimé

$\Theta(k)$ $T \leftarrow$ Premier();
$\Theta(k)$ Imprimer T;

tant que *non Dernier(T)* **faire**
  $\Theta(k)$ $T \leftarrow$ Suivant(T);
  $\Theta(k)$ Imprimer(T);
fin

On passe dans la répétitive C_n^k fois

La complexité $\Theta(k \times C_n^k)$ est optimale !