Un dernier cours

P. B.

5 mai 2008

Survol du cours

- Quelques concepts
- Algos par classes de complexité

One more thing

Une petite tranche d'histoire

Survol du cours

Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :

Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).

Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).
  - Pile d'appel.

Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).
  - Pile d'appel.
- Pour étudier les algorithmes :

Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).
  - Pile d'appel.
- Pour étudier les algorithmes :
  - Notation asymptotique.

Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).
  - Pile d'appel.
- Pour étudier les algorithmes :
  - Notation asymptotique.
  - Optimalité.

Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).
  - Pile d'appel.
- Pour étudier les algorithmes :
  - Notation asymptotique.
  - Optimalité.
  - Invariant de boucle.

Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).
  - Pile d'appel.
- Pour étudier les algorithmes :
  - Notation asymptotique.
  - Optimalité.
  - Invariant de boucle.
  - Arbre de décision. Arbre d'appel.

Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).
  - Pile d'appel.
- Pour étudier les algorithmes :
  - Notation asymptotique.
  - Optimalité.
  - Invariant de boucle.
  - Arbre de décision. Arbre d'appel.
  - Séries (arithmétique, géométrique, formelle).

Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).
  - Pile d'appel.
- Pour étudier les algorithmes :
  - Notation asymptotique.
  - Optimalité.
  - Invariant de boucle.
  - Arbre de décision. Arbre d'appel.
  - Séries (arithmétique, géométrique, formelle).
- Pour écrire des algorithmes :

Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).
  - Pile d'appel.
- Pour étudier les algorithmes :
  - Notation asymptotique.
  - Optimalité.
  - Invariant de boucle.
  - Arbre de décision. Arbre d'appel.
  - Séries (arithmétique, géométrique, formelle).
- Pour écrire des algorithmes :
  - Programmation récursive.

Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).
  - Pile d'appel.
- Pour étudier les algorithmes :
  - Notation asymptotique.
  - Optimalité.
  - Invariant de boucle.
  - Arbre de décision. Arbre d'appel.
  - Séries (arithmétique, géométrique, formelle).
- Pour écrire des algorithmes :
  - Programmation récursive.
  - Construction à partir d'un invariant.

Concepts, outils, méthodes

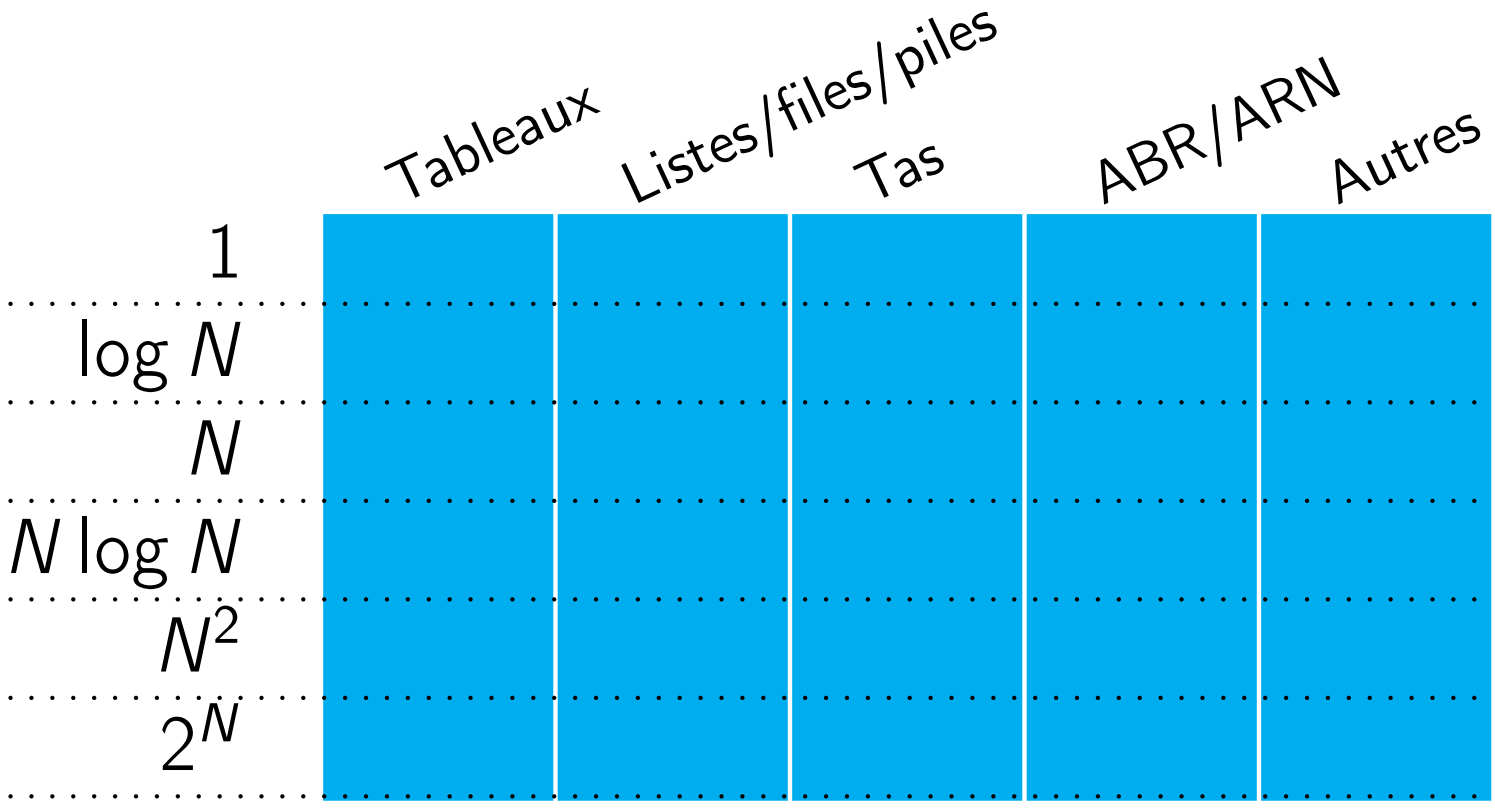
- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).
  - Pile d'appel.
- Pour étudier les algorithmes :
  - Notation asymptotique.
  - Optimalité.
  - Invariant de boucle.
  - Arbre de décision. Arbre d'appel.
  - Séries (arithmétique, géométrique, formelle).
- Pour écrire des algorithmes :
  - Programmation récursive.
  - Construction à partir d'un invariant.
  - Diviser pour régner.

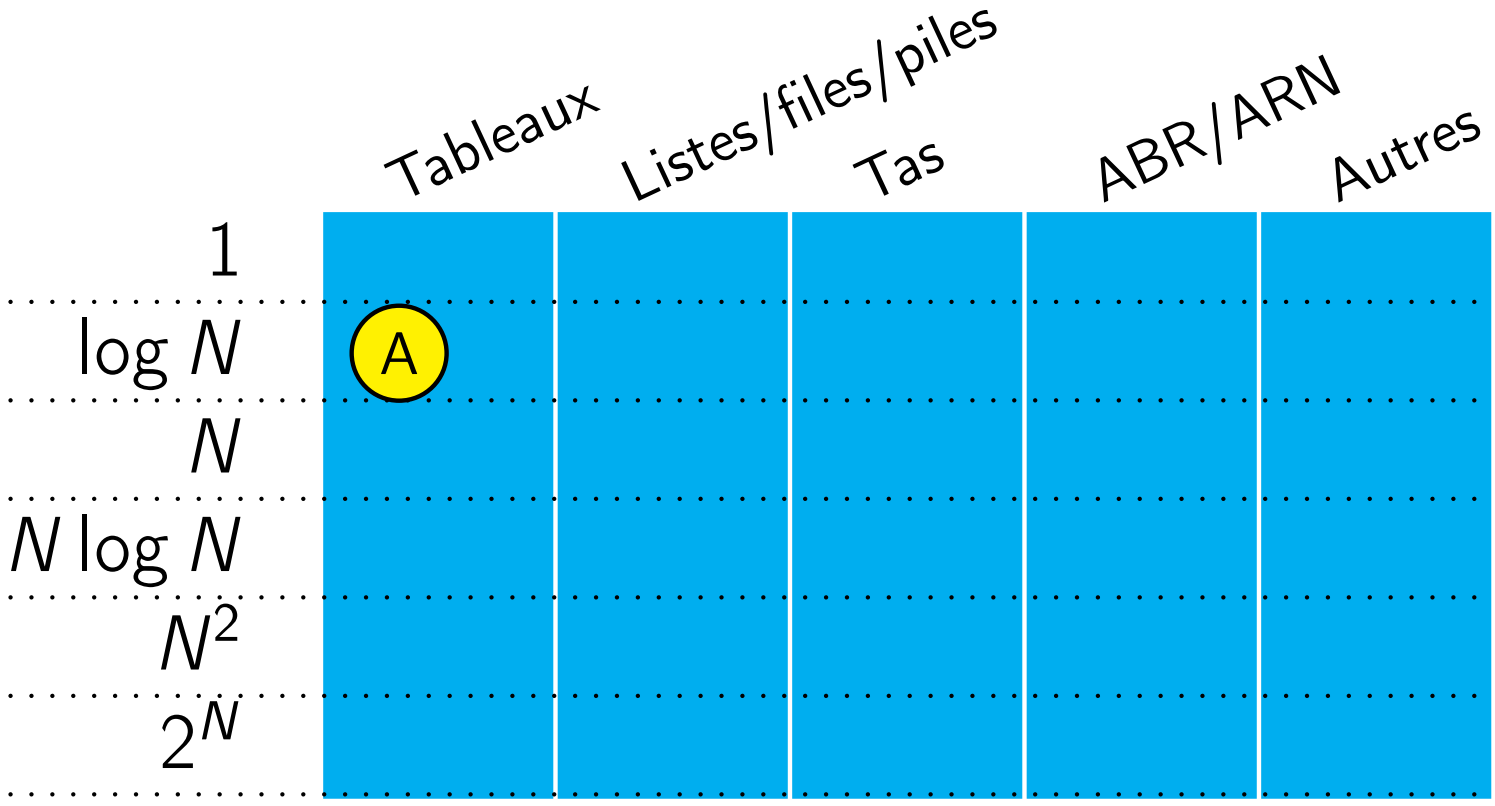
Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).
  - Pile d'appel.
- Pour étudier les algorithmes :
  - Notation asymptotique.
  - Optimalité.
  - Invariant de boucle.
  - Arbre de décision. Arbre d'appel.
  - Séries (arithmétique, géométrique, formelle).
- Pour écrire des algorithmes :
  - Programmation récursive.
  - Construction à partir d'un invariant.
  - Diviser pour régner.
  - Structures de données (piles, listes chaînées, arbres).

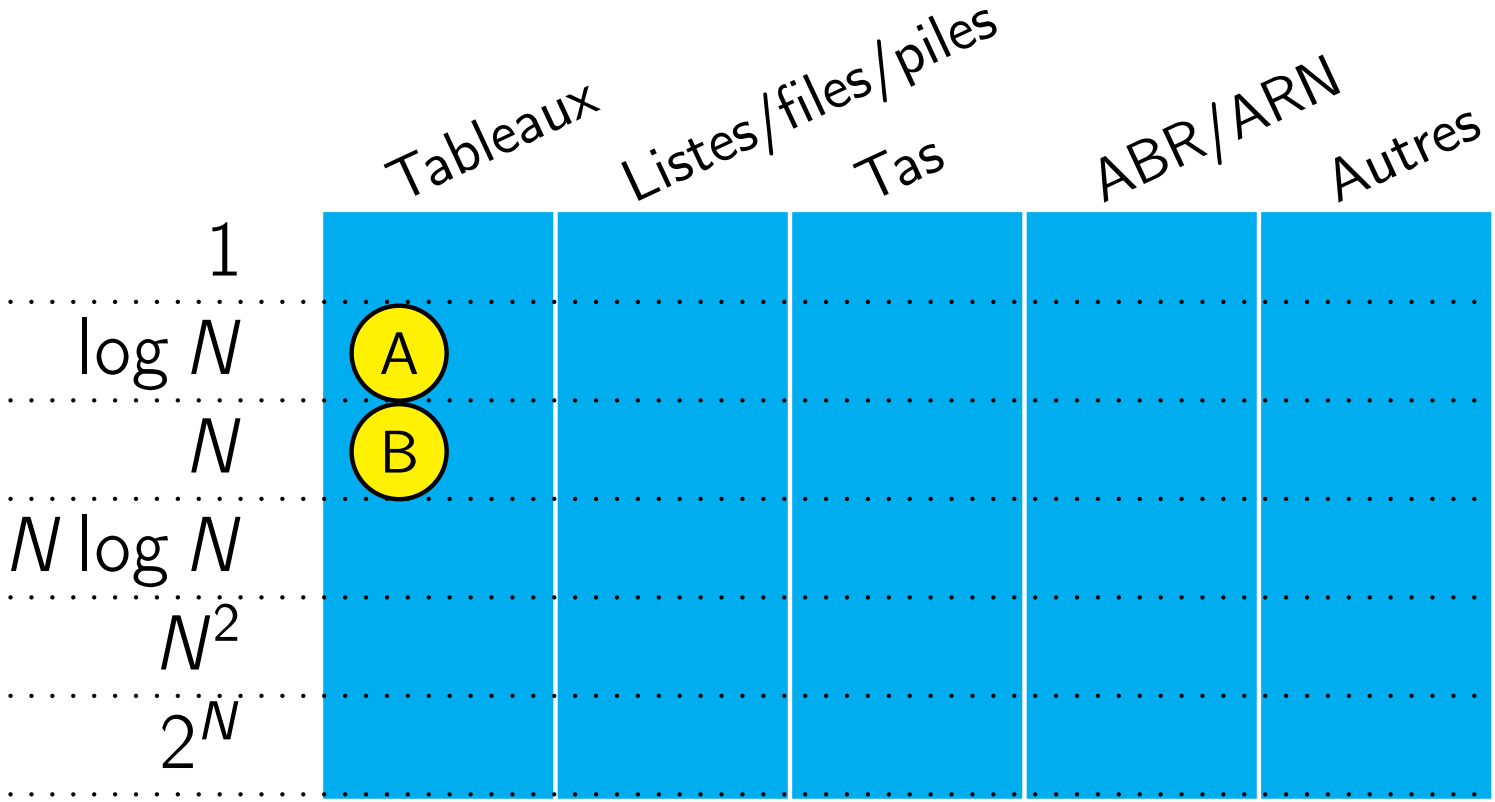
Concepts, outils, méthodes

- Fonctionnement des programmes informatiques :
  - Nombres binaires, log (base 2).
  - Pile d'appel.
- Pour étudier les algorithmes :
  - Notation asymptotique.
  - Optimalité.
  - Invariant de boucle.
  - Arbre de décision. Arbre d'appel.
  - Séries (arithmétique, géométrique, formelle).
- Pour écrire des algorithmes :
  - Programmation récursive.
  - Construction à partir d'un invariant.
  - Diviser pour régner.
  - Structures de données (piles, listes chaînées, arbres).
  - Programmation dynamique (le robot cupide).


Tableaux
Listes/files/piles
Tas
ABR/ARN
Autres
1
log N
N
N log N
N²
2^N

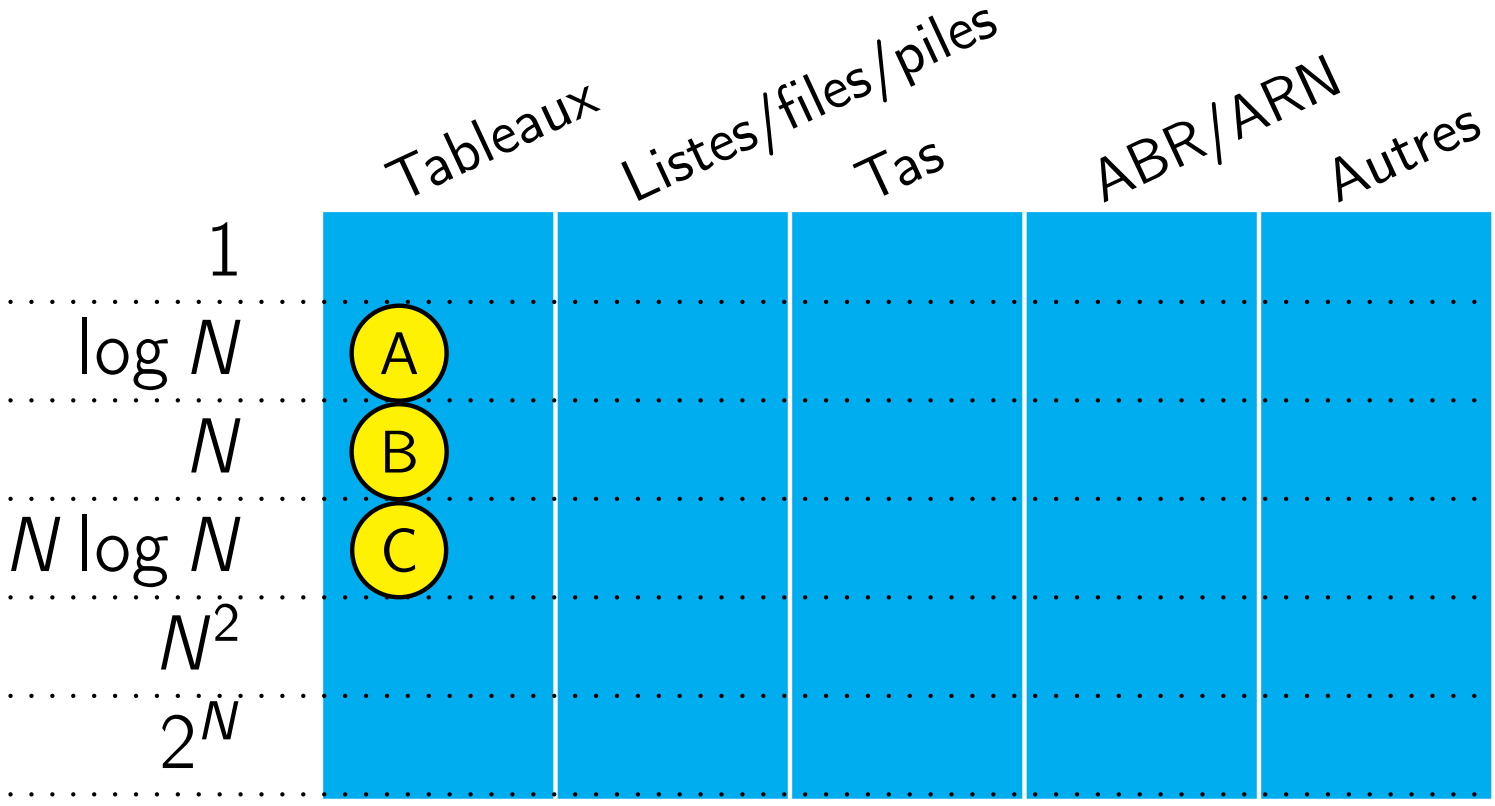


A La recherche dichotomique (tableau trié). Division par 2.

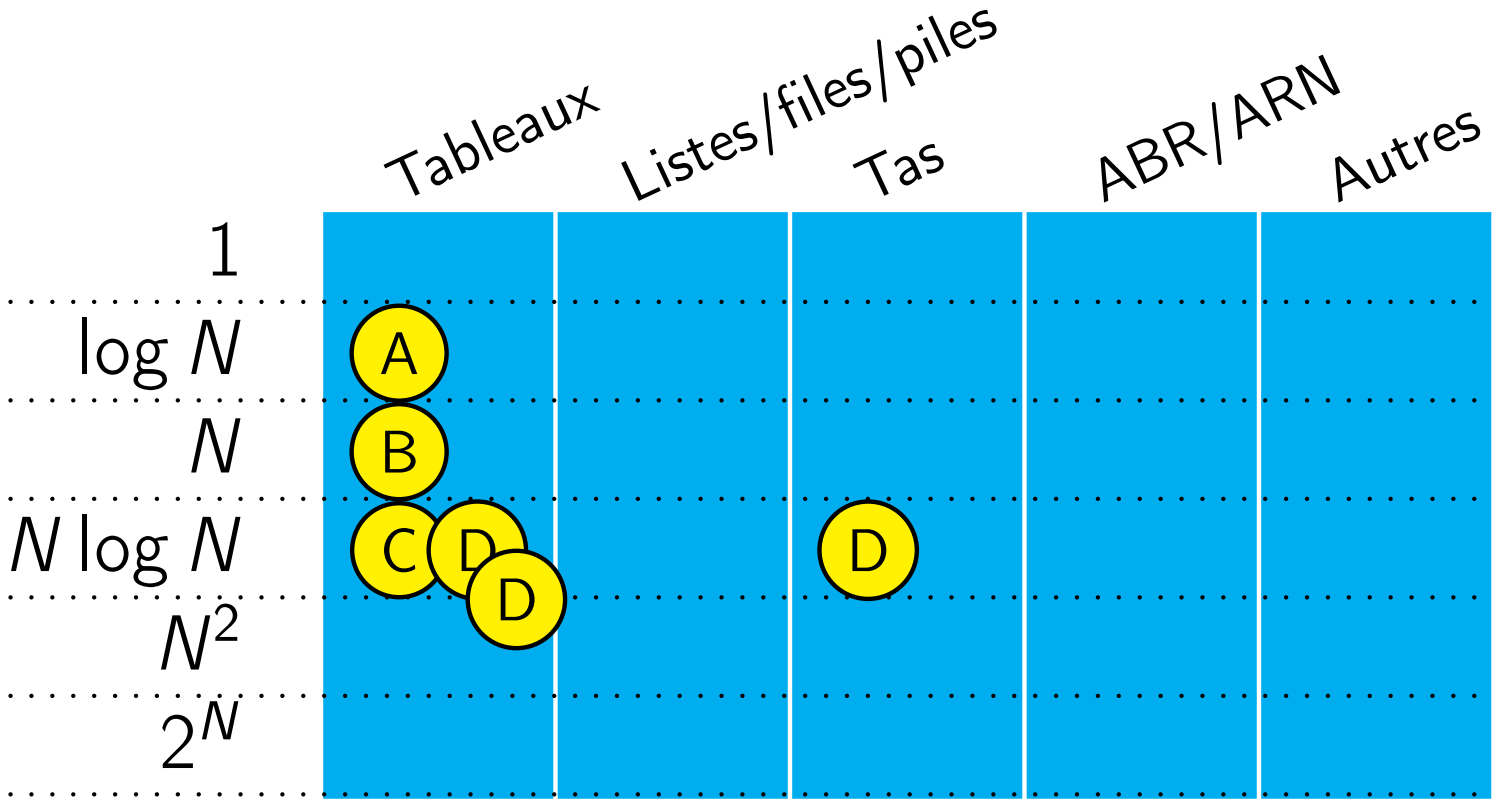


A La recherche dichotomique (tableau trié). Division par 2.

B Par parcours. Recherche du maximum, ou d'un élément dans un tableau non trié (1 parcours). Ajout dans un tableau trié ($\frac{1}{2}$ p.). Certains tris sur des ensembles restreints de clés : tri par dénombrement, tri du postier, tri par base (k p.). Interclassement (1 p.). Partition (1 p.).

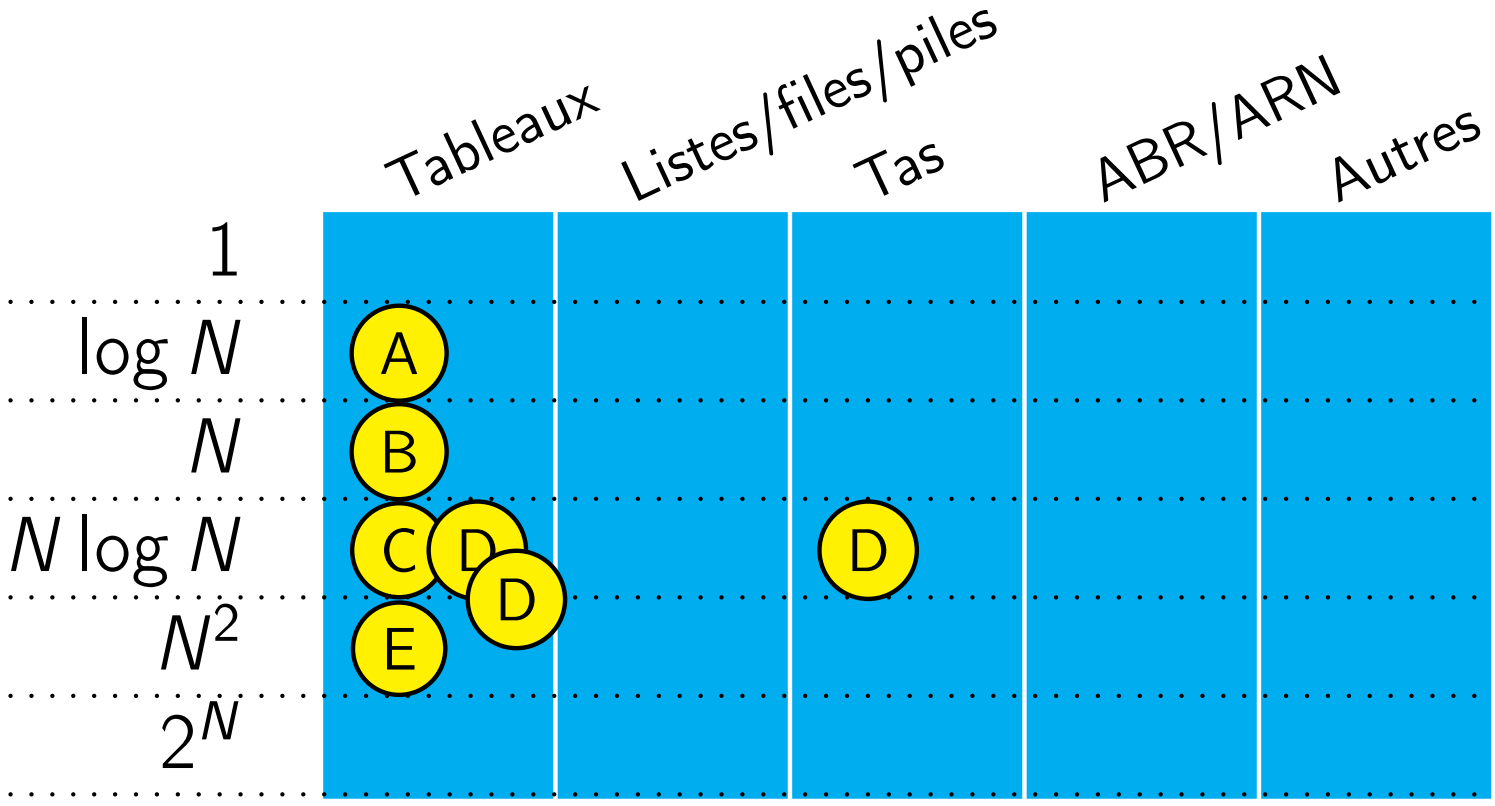


C Nombre minimum de comparaisons en moyenne des tris par comparaison. ($\log(N!)$).



C Nombre minimum de comparaisons en moyenne des tris par comparaison. ($\log(N!)$).

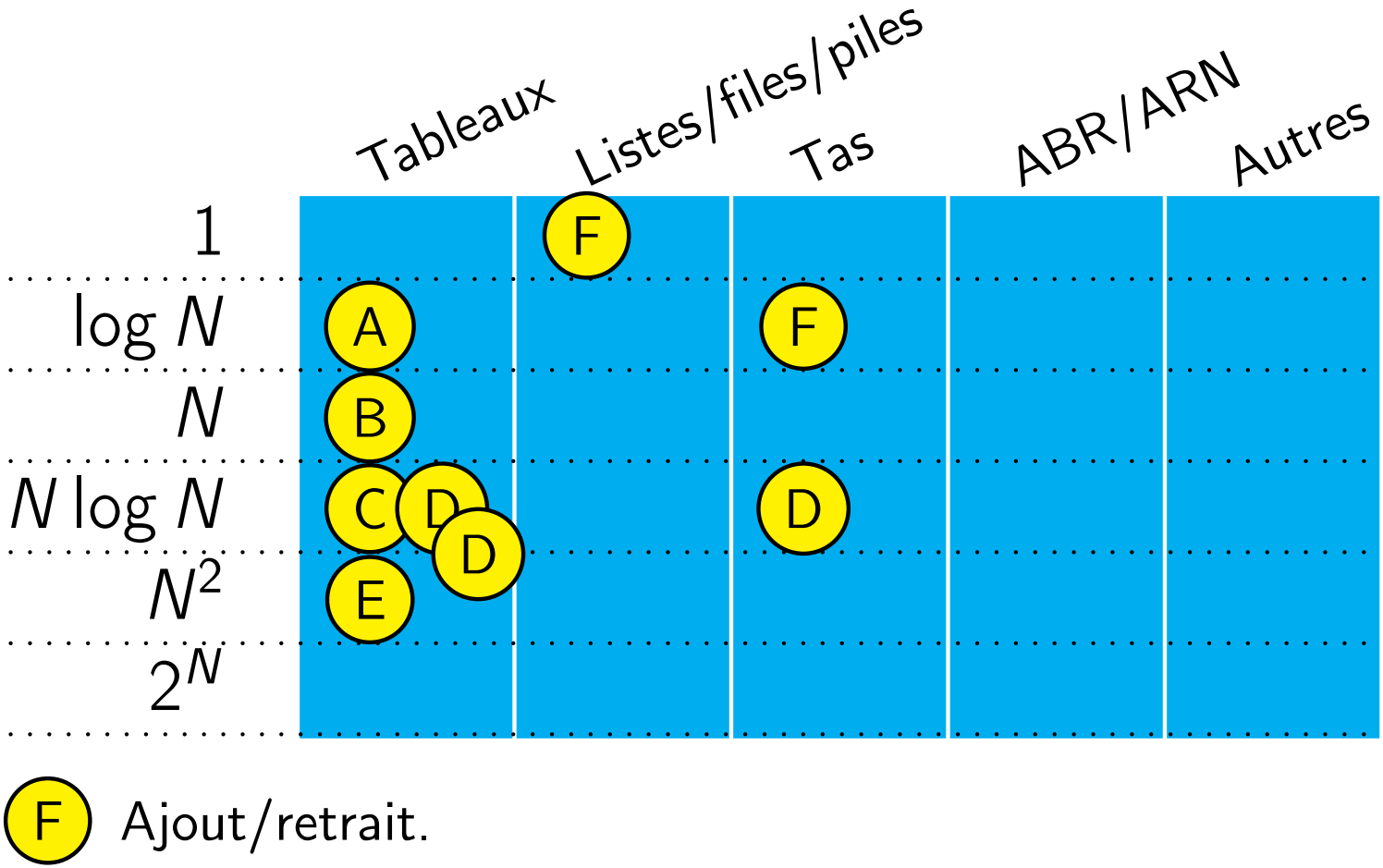
D Tris *diviser pour régner* : fusion et quicksort en moyenne (div. par 2 et parcours). Tri par tas.



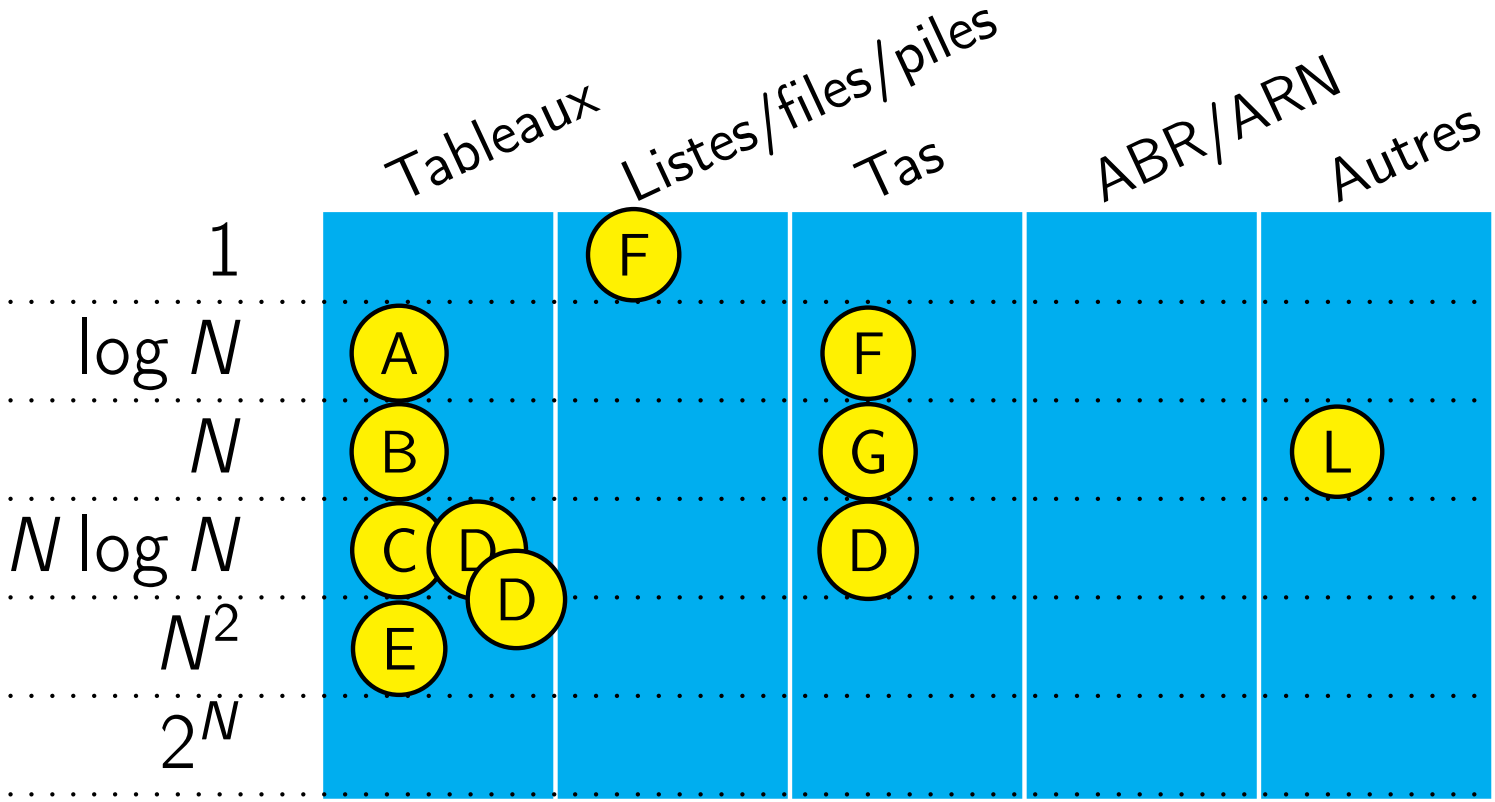
C Nombre minimum de comparaisons en moyenne des tris par comparaison. ($\log(N!)$).

D Tris *diviser pour régner* : fusion et quicksort en moyenne (div. par 2 et parcours). Tri par tas.

E Tris plus naïfs (bulle, sélection, insertion). Deux parcours imbriqués (p. $\times$ p.).

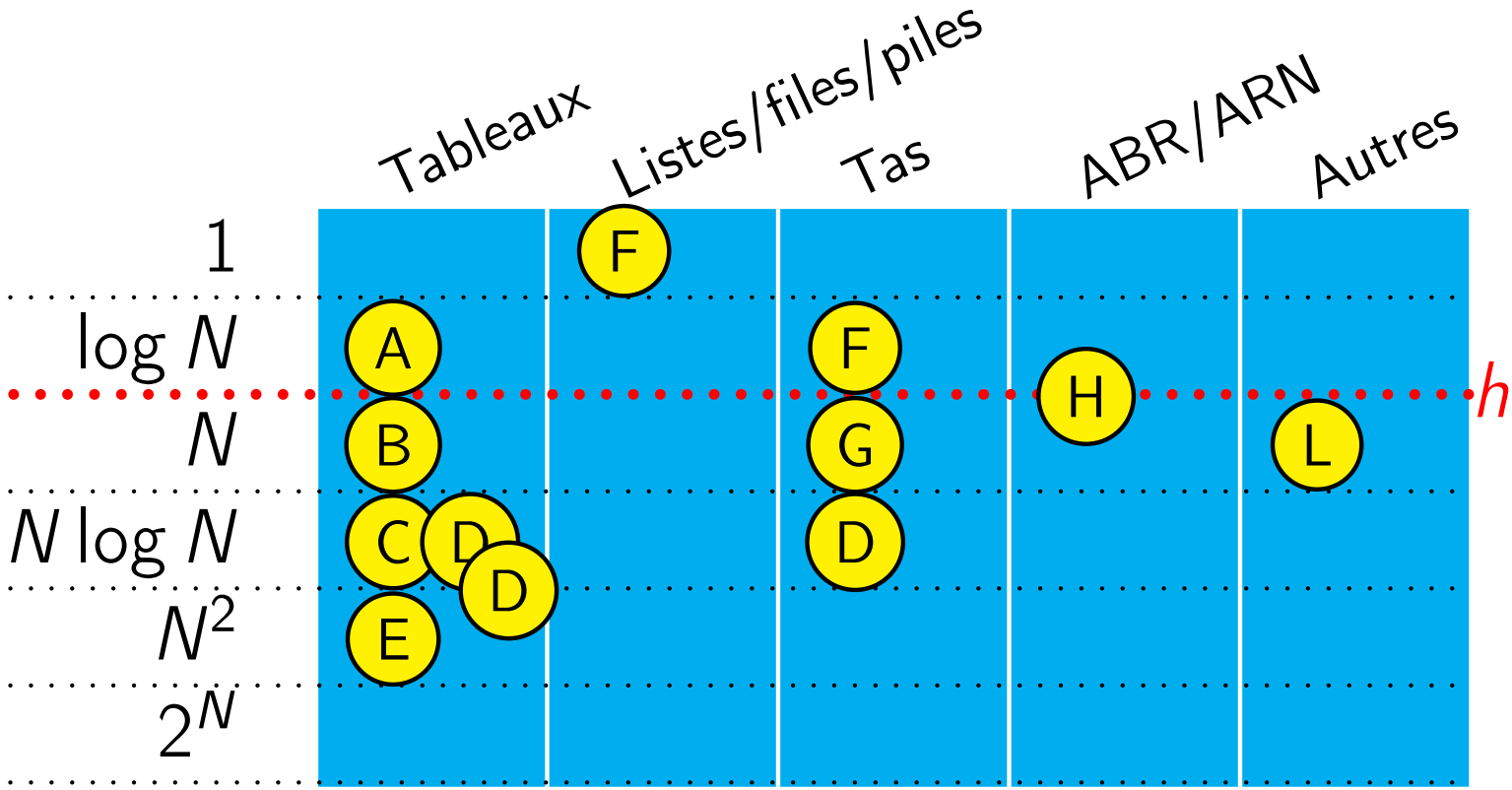


F Ajout/retrait.

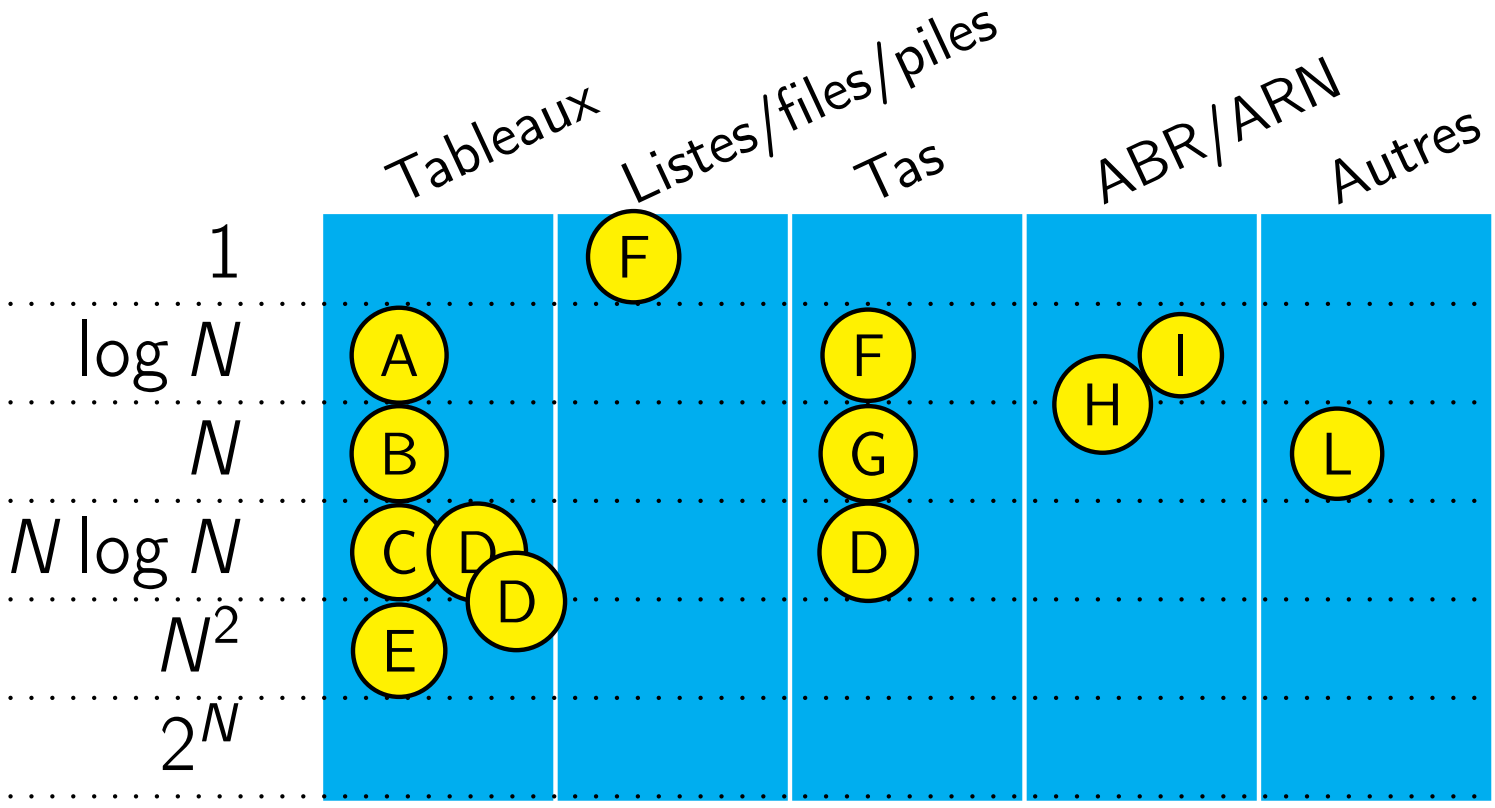


F Ajout/retrait.

G Former un tas.

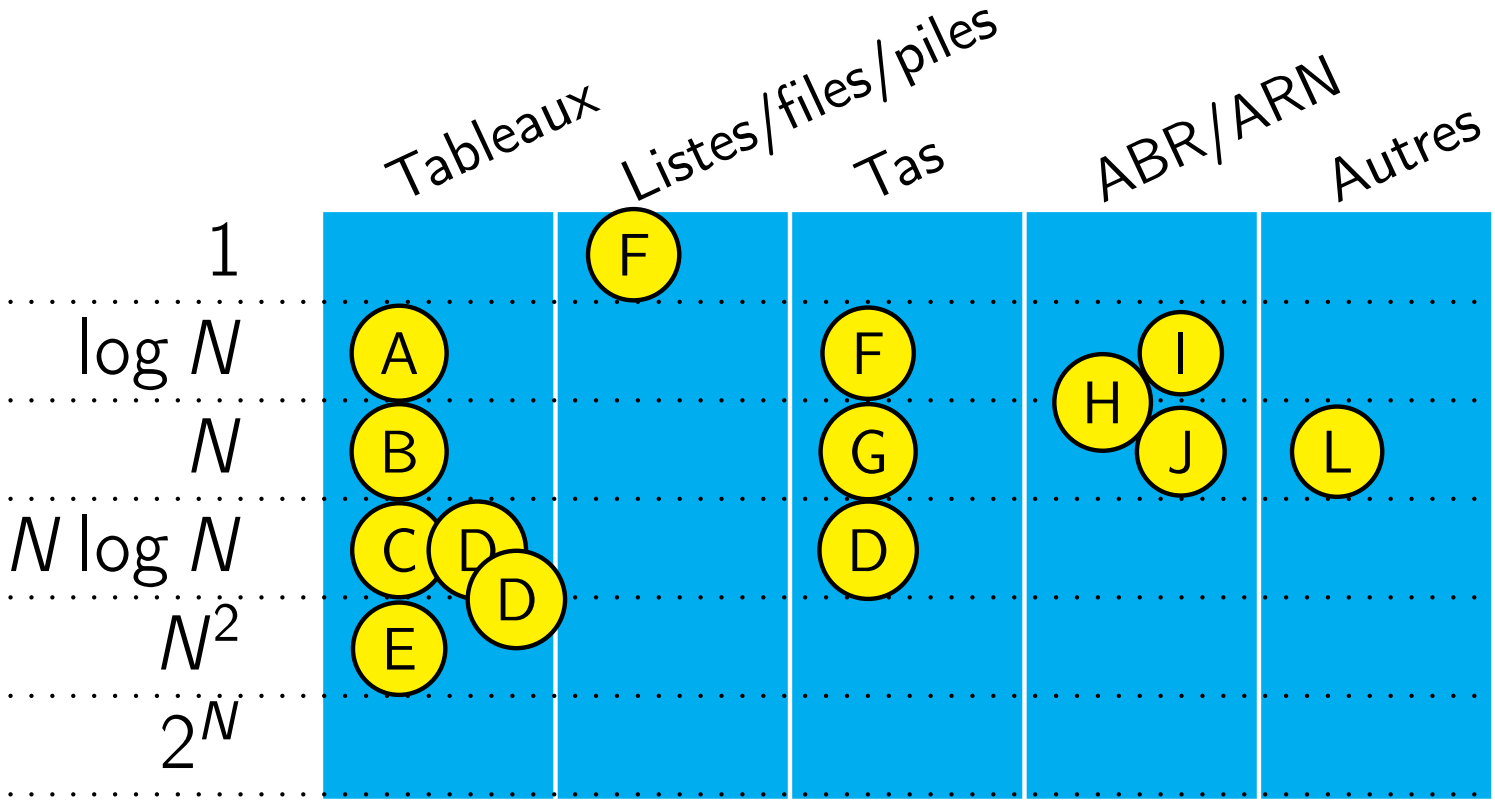


H Ajout/retrait/recherche dans les ABR.



H Ajout/retrait/recherche dans les ABR.

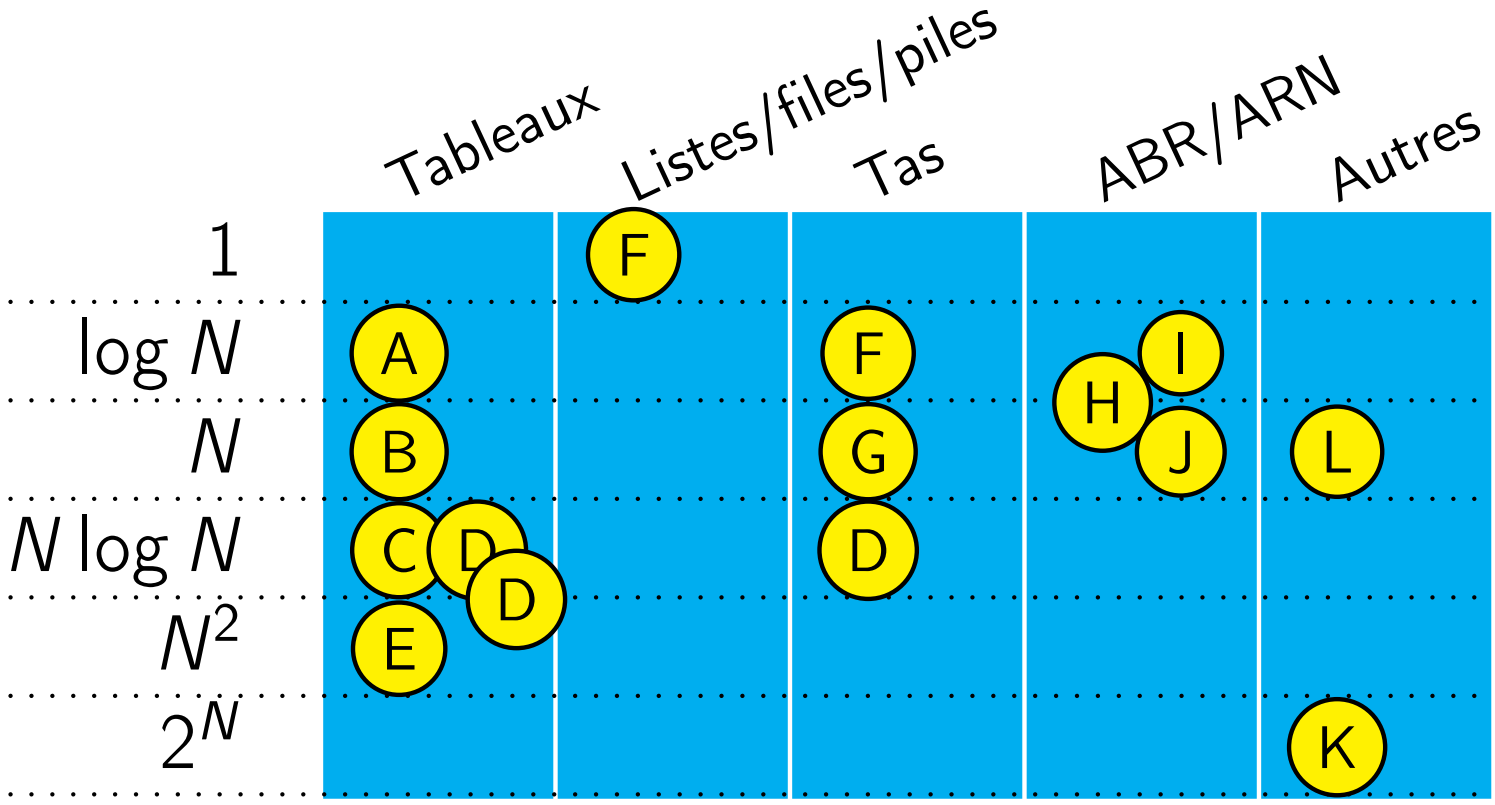
I Ajout/retrait/recherche dans les ABR rouge noir.



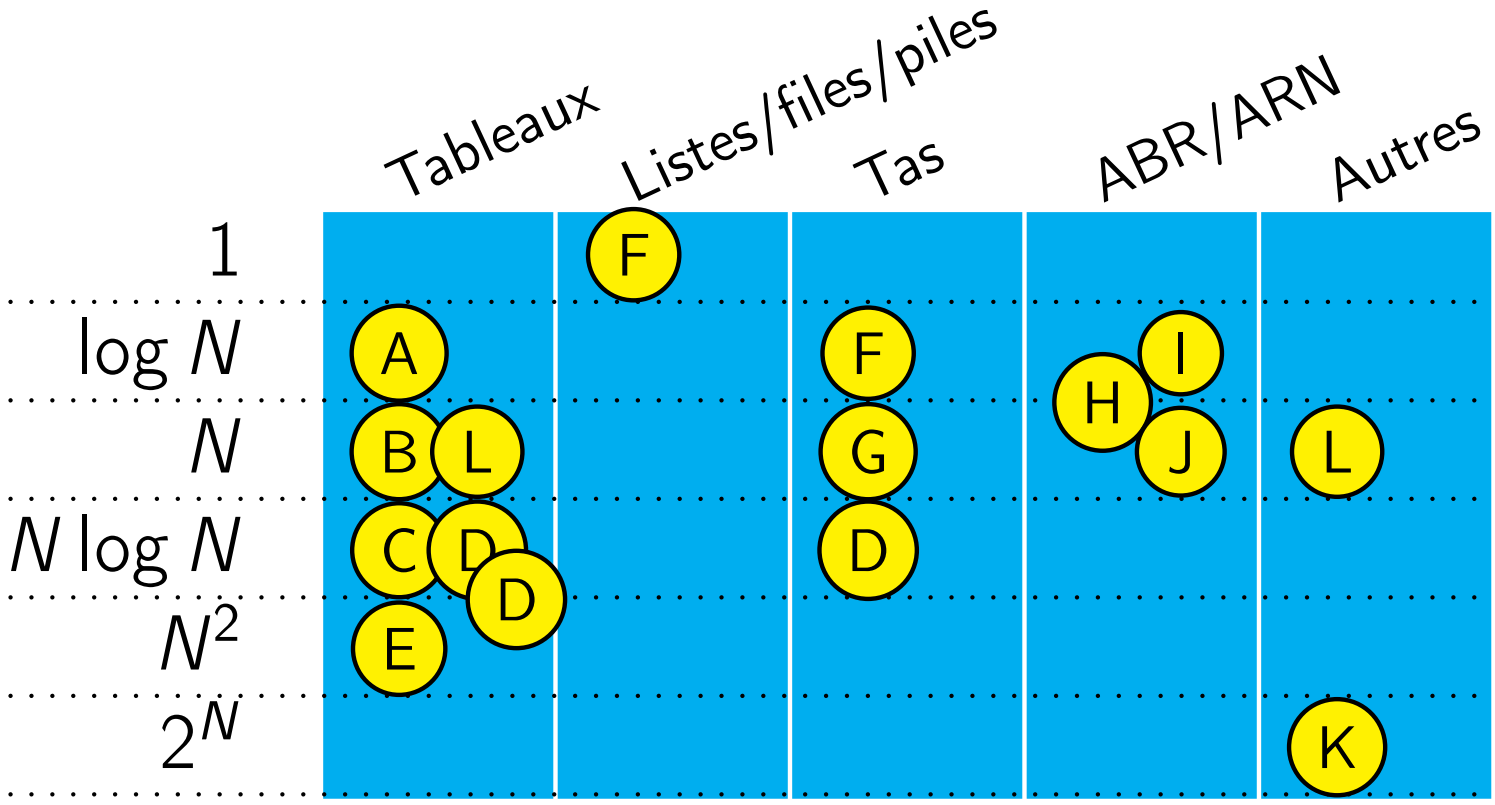
H Ajout/retrait/recherche dans les ABR.

I Ajout/retrait/recherche dans les ABR rouge noir.

J Parcours.

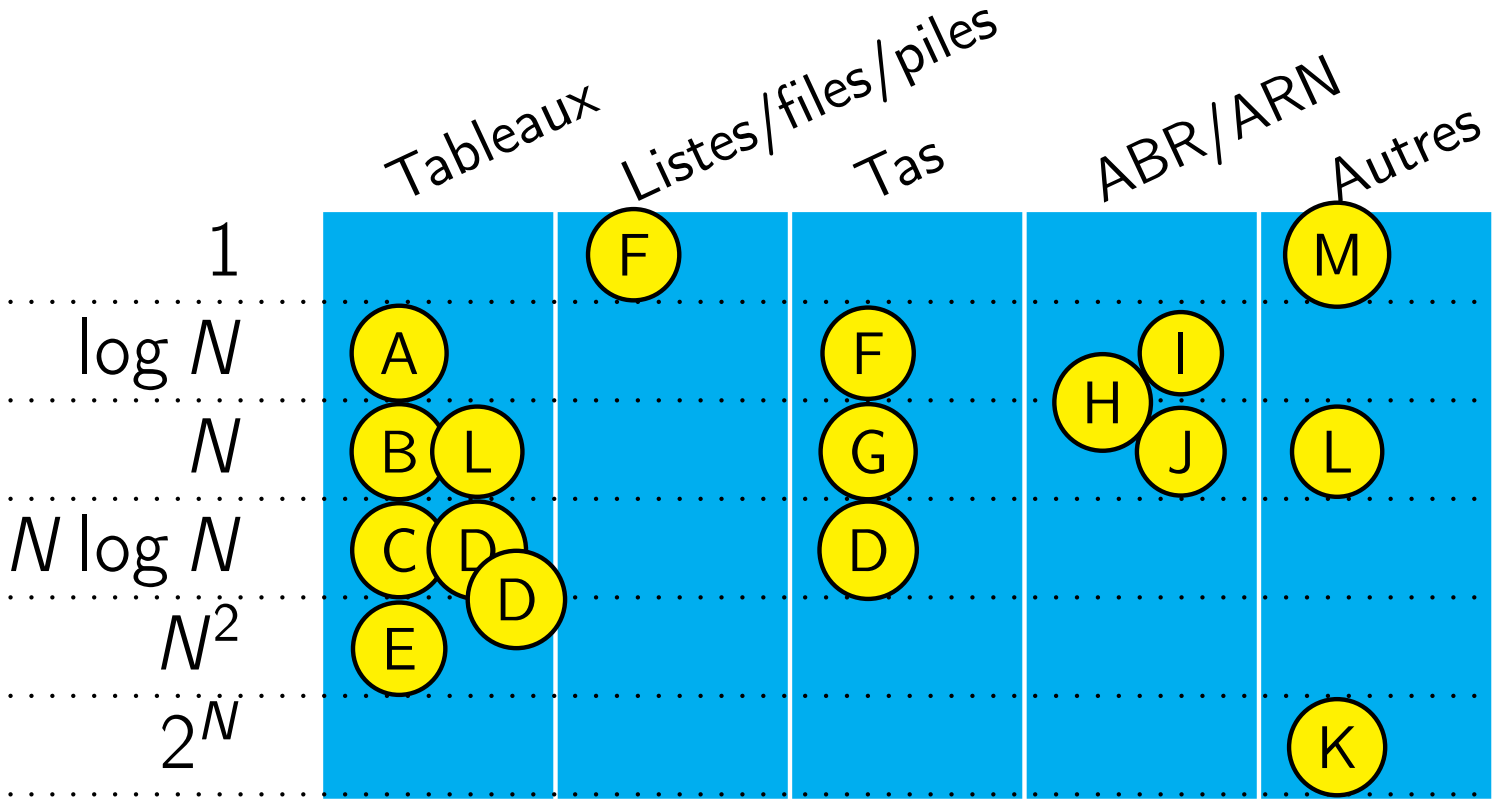


K Tours de Hanoï (deux appels récursifs pour passer de N à $N-1$).



K Tours de Hanoï (deux appels récursifs pour passer de N à $N-1$).

L Médian et rang (moyenne). Rue $\mathbb{Z}$.



K Tours de Hanoï (deux appels récursifs pour passer de N à $N-1$).

L Médian et rang (moyenne). Rue $\mathbb{Z}$.

M Ajout/retrait/recherche dans les tables de hachage (en moyenne, si la distribution est uniforme et que le taux de remplissage est raisonnable).

One more thing

L'indécidabilité de l'arrêt

- On peut énumérer, par les entiers, tous les programmes.

L'indécidabilité de l'arrêt

- On peut énumérer, par les entiers, tous les programmes.
- Un programme prend un entier en entrée et s'arrête (1) ou boucle éternellement (0).

L'indécidabilité de l'arrêt

- On peut énumérer, par les entiers, tous les programmes.
- Un programme prend un entier en entrée et s'arrête (1) ou boucle éternellement (0).

Donnée :	0	1	2	...
Programme 0	1	1	1	.....
Programme 1	1	0	0	.....
Programme 2	0	0	1	.....
⋮	⋮	⋮	⋮	⋮

L'indécidabilité de l'arrêt

- On peut énumérer, par les entiers, tous les programmes.
- Un programme prend un entier en entrée et s'arrête (1) ou boucle éternellement (0).

Donnée :	0	1	2	...
Programme 0	1	1	1	.....
Programme 1	1	0	0	.....
Programme 2	0	0	1	.....
⋮	⋮	⋮	⋮	⋮

- **On suppose** qu'il existe un programme $f(p, q)$ qui renvoie 1 si le programme n° p s'arrête sur la donnée q et **renvoie** 0 **sinon**.

L'indécidabilité de l'arrêt

- On peut énumérer, par les entiers, tous les programmes.
- Un programme prend un entier en entrée et s'arrête (1) ou boucle éternellement (0).

Donnée :	0	1	2	...
Programme 0	1	1	1	.....
Programme 1	1	0	0	.....
Programme 2	0	0	1	.....
⋮	⋮	⋮	⋮	⋮

- **On suppose** qu'il existe un programme $f(p, q)$ qui renvoie 1 si le programme n° p s'arrête sur la donnée q et **renvoie** 0 **sinon**.
- On construit un programme g qui prend en entrée un entier p et s'arrête si $f(p, p)$ vaut 0 ou boucle éternellement sinon. Le programme g a lui même un numéro, n.

L'indécidabilité de l'arrêt

- On peut énumérer, par les entiers, tous les programmes.
- Un programme prend un entier en entrée et s'arrête (1) ou boucle éternellement (0).

Donnée :	0	1	2	...
Programme 0	1	1	1	.....
Programme 1	1	0	0	.....
Programme 2	0	0	1	.....
	⋮	⋮	⋮	.....
Programme n				.....
	⋮	⋮	⋮	

- **On suppose** qu'il existe un programme $f(p, q)$ qui renvoie 1 si le programme n° p s'arrête sur la donnée q et **renvoie** 0 **sinon**.
- On construit un programme g qui prend en entrée un entier p et s'arrête si $f(p, p)$ vaut 0 ou boucle éternellement sinon. Le programme g a lui même un numéro, n.

L'indécidabilité de l'arrêt

- On peut énumérer, par les entiers, tous les programmes.
- Un programme prend un entier en entrée et s'arrête (1) ou boucle éternellement (0).

Donnée :	0	1	2	...
Programme 0	1	1	1	.....
Programme 1	1	0	0	.....
Programme 2	0	0	1	.....
⋮	⋮	⋮	⋮	⋮
Programme n	0			.....
⋮	⋮	⋮	⋮	⋮

- **On suppose** qu'il existe un programme $f(p, q)$ qui renvoie 1 si le programme n° p s'arrête sur la donnée q et **renvoie** 0 **sinon**.
- On construit un programme g qui prend en entrée un entier p et s'arrête si $f(p, p)$ vaut 0 ou boucle éternellement sinon. Le programme g a lui même un numéro, n.

L'indécidabilité de l'arrêt

- On peut énumérer, par les entiers, tous les programmes.
- Un programme prend un entier en entrée et s'arrête (1) ou boucle éternellement (0).

Donnée :	0	1	2	...
Programme 0	1	1	1	.....
Programme 1	1	0	0	.....
Programme 2	0	0	1	.....
⋮	⋮	⋮	⋮	⋮
Programme n	0	1		.....
⋮	⋮	⋮	⋮	⋮

- **On suppose** qu'il existe un programme $f(p, q)$ qui renvoie 1 si le programme n° p s'arrête sur la donnée q et **renvoie** 0 **sinon**.
- On construit un programme g qui prend en entrée un entier p et s'arrête si $f(p, p)$ vaut 0 ou boucle éternellement sinon. Le programme g a lui même un numéro, n.

L'indécidabilité de l'arrêt

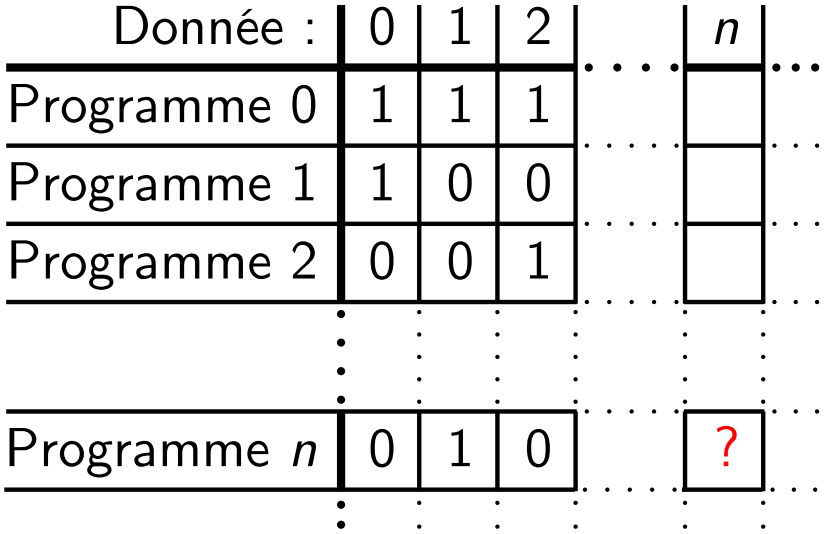
- On peut énumérer, par les entiers, tous les programmes.
- Un programme prend un entier en entrée et s'arrête (1) ou boucle éternellement (0).

Donnée :	0	1	2	...
Programme 0	1	1	1	.....
Programme 1	1	0	0	.....
Programme 2	0	0	1	.....
	⋮	⋮	⋮	.....
Programme n	0	1	0	.....
	⋮	⋮	⋮	⋮

- **On suppose** qu'il existe un programme $f(p, q)$ qui renvoie 1 si le programme n° p s'arrête sur la donnée q et **renvoie** 0 **sinon**.
- On construit un programme g qui prend en entrée un entier p et s'arrête si $f(p, p)$ vaut 0 ou boucle éternellement sinon. Le programme g a lui même un numéro, n.

L'indécidabilité de l'arrêt

- On peut énumérer, par les entiers, tous les programmes.
- Un programme prend un entier en entrée et s'arrête (1) ou boucle éternellement (0).



- **On suppose** qu'il existe un programme $f(p, q)$ qui renvoie 1 si le programme n° p s'arrête sur la donnée q et **renvoie** 0 **sinon**.
- On construit un programme g qui prend en entrée un entier p et s'arrête si $f(p, p)$ vaut 0 ou boucle éternellement sinon. Le programme g a lui même un numéro, n.

Est-ce que $g(n)$ s'arrête ? Que vaut $f(n, n)$?

L'indécidabilité de l'arrêt

- On peut énumérer, par les entiers, tous les programmes.
- Un programme prend un entier en entrée et s'arrête (1) ou boucle éternellement (0).

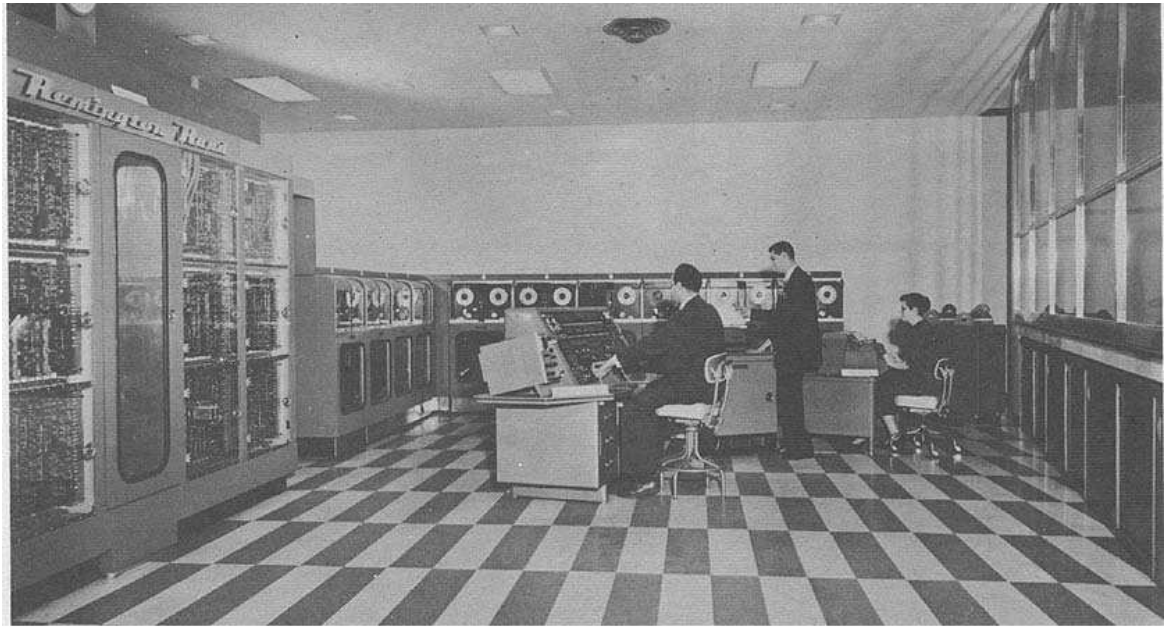
Donnée :	0	1	2	...	n	...
Programme 0	1	1	1	...		...
Programme 1	1	0	0	...		...
Programme 2	0	0	1	...		...
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Programme n	0	1	0	...	?	...

- **On suppose** qu'il existe un programme f qui renvoie 1 si le programme n° p s'arrête sur la donnée p et **renvoie** 0 **sinon**.
- On construit un programme g qui prend en entrée un entier p et s'arrête si $f(p, p)$ vaut 0 ou boucle éternellement sinon. Le programme g a lui même un numéro, n.

Contradiction

Est-ce que $g(n)$ s'arrête ? Que vaut $f(n, n)$?

Une petite tranche d'histoire



1952

Dans les années 50, les ordinateurs entrent peu à peu dans l'industrie.
En 1952, un ordinateur UNIVAC I est utilisé par la chaîne de télévision CBS pour prédire le vainqueur de l'élection présidentielle opposant Enseinhower à Stevenson.

1954

Le 7 juin 1954, mort à 41 ans de Alan Turing.
En 1936, il publiait *On Computable Numbers, with an Application to the Entscheidungsproblem* reformulant les résultats de Kurt Gödel comme indécidabilité du problème de l'arrêt. Il introduit pour cela une notion de machine à ruban infini, les machines de Turing. Il travailla pendant la guerre pour le chiffre et le contre-espionnage allié avec de nombreux succès, mêlant découvertes mathématiques et machines électromécaniques. Les premiers ordinateurs programmables naissaient dans la foulée. Alan Turing est considéré comme l'un des inventeurs, sinon le père, de l'informatique.

« L'informatique n'est pas plus la science des ordinateurs que l'astronomie n'est celle des télescopes. »

E. W. Diskstra

1955

Au début du printemps 1952, le jeune Edsger W. Dijkstra découvre la programmation.
Trois ans plus tard, nous sommes en 1955, Dijkstra hésite entre poursuivre une carrière de physicien théorique ou se lancer dans l'informatique, discipline naissante dont il est l'un des tout premiers représentants aux Pays-Bas. Est-ce une profession respectable que d'être programmeur se demande t'il ? Son patron au centre mathématique d'Amsterdam lui conseille de continuer dans l'informatique en lui disant qu'il n'appartient qu'à Dijkstra d'en faire une discipline respectable.

1956

Sur les écrans de cinéma, les américains découvrent avec *Gog*, en 1954, un superordinateur devenu assassin suite à un sabotage russe. Et en 1956, avec *La planète interdite*, un ordinateur gigantesque construit par une civilisation extraterrestre . . . ainsi que ce qu'il y a de mieux comme robot à tout faire.

1957

Le 8 février 1957, mort à 53 ans de John von Neumann.
Il fît des contributions, souvent majeures, dans différents champs des mathématiques, ainsi qu'en économie, en mécanique quantique, et ne fût pas en reste en matières de bombes et d'armes nucléaires.
Il est l'inventeur du tri fusion et de l'architecture moderne des ordinateurs. Il meurt d'un cancer, probablement contracté à la suite de son implication dans le projet Manahattan.

1957

Les premiers compilateurs voient le jour. Celui du langage FORTRAN développé par l'équipe de John Backus chez IBM est considéré comme le premier compilateur complet. Le compilateur LISP, 1962, est le premier à se compiler lui-même.

1958

Naissance d'ALGOL, ALGOrithmic Language (au départ *international algorithmic language*). Ce langage qui connaîtra plusieurs versions, dont le fameux ALGOL 60, va avoir un grand succès dans les milieux académiques, et il deviendra notamment le standard pour l'écriture d'algorithmes. Il inspirera de nombreux autres langages tels que le Pascal. L'absence de standard bien défini pour les entrées/sorties limitera son adoption dans l'industrie.
C'est la première fois qu'apparaît la paire `begin`/`end` pour le parenthésage des blocs d'expressions.

« We should forget about small efficiencies, say about 97% of the time : premature optimization is the root of all evil »

Tony Hoare.

1960

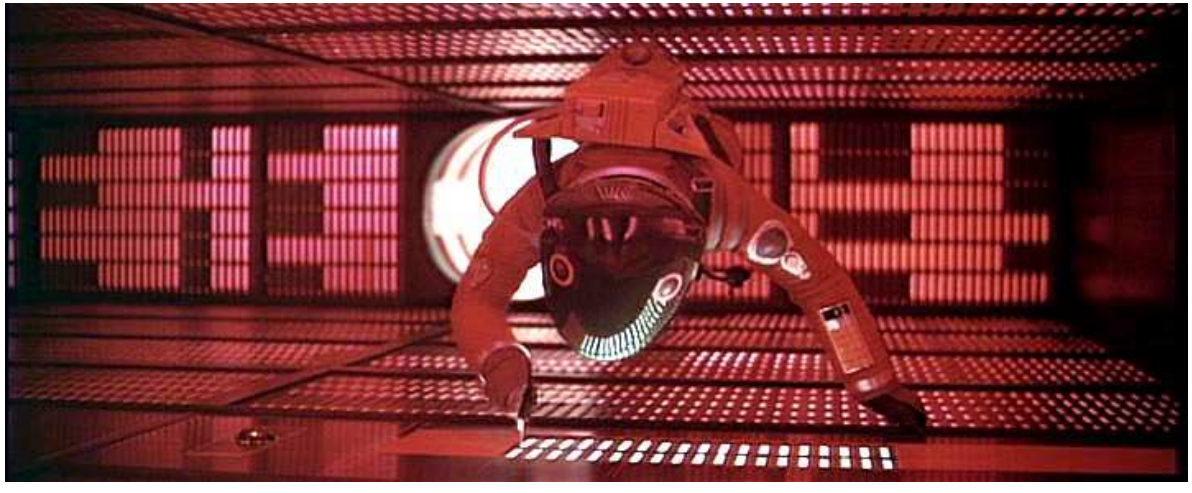
Sir Charles Antony Richard Hoare invente le Quicksort.

1963

En 1963, Donald Knuth soutien sa thèse de mathématiques et entame son travail sur son œuvre majeure *The Art of Computer Programming*. Avant cela il avait travaillé sur les compilateurs. Il vient également de faire une contribution importante concernant l'adressage ouvert (n objets pour m tiroirs, $n \leq m$).
TAoCP, dont le premier volume sortira en 1968, fonde l'analyse mathématique des algorithmes et popularise la notation asymptotique.
Accessoirement Knuth créera le logiciel TEX pour la publication du troisième volume.

1968

En 1968, sur les écrans de cinéma, HAL (CARL en vf) apprend en lisant sur les lèvres des astronautes Dave et Frank qu'ils vont essayer de le déconnecter.
HAL (*Heuristically programmed ALgorithmic Computer*), est un super-ordinateur considéré comme le sixième membre de l'équipage du vaisseau spatiale *Discovery*.
En 2001, un ordinateur comme HAL reste de la science fiction.



1968

Dijkstra publie la lettre *Goto statement considered harmful* qui marque le point de départ de la programmation structurée. En 1972, Dijkstra, Hoare et Ole-Johan Dahl publieront le livre *Structured programming*.

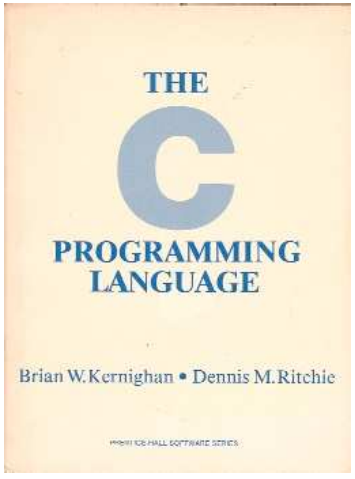
1970

Le gouvernement socialiste de Salvador Allende, nouvellement élu, lance CYBERSYN un système informatique de planification temps réel de l'économie chilienne. Il engage pour cela le cybernéticien anglais Stafford Beer et achète un premier IBM 360. Des Telex servent à transmettre les données entre des usines réparties dans tout le Chili et l'IBM 360. Ce dernier permet de visualiser les données dans un salle de contrôle aux allures futuristes, de laquelle en retour partent les directives de productions. L'expérience n'en est qu'à ses débuts lors du coup d'État de Pinochet, plus adepte de Milton Friedman en matière d'économie.

Quelque chose comme `QWERTYUIOP`

1971

Premier mail sur le réseau ARPANET.



1973

Les premières versions d'UNIX sortent début 70. En 1971, sortie du premier manuel du programmeur UNIX. En 1973, unix est réécrit en langage C, créé pour l'occasion. Le livre *The C Programming Language* sortira en 1978.

En 1983, Ken Thomson et Dennis M. Ritchie reçoivent le Turing award pour leur travail sur UNIX. À cette occasion, Ken Thomson prononcera le discours *reflections on trusting trust*.