Ce que l'informatique ne pourra jamais faire ?

Arnaud Casteigts
Université de Genève

Rencontres de Venise :
Epistémologie

Jan 8, 2025.

Disclaimer...

Dana Angluin (1947 -)

Professeure honoraire à Yale

Nombreuses contributions en informatique fondamentale.

Types de preuves :

Proof by example :
The author gives only the case n=2 and suggests that it contains most of the ideas of the general proof.

Proof by intimidation :
'Trivial.'

Proof by vigorous handwaving :
Works well in a classroom or seminar setting.

Proof by cumbersome notation :
Best done with access to at least four alphabets and special symbols.

Proof by exhaustion :
An issue or two of a journal devoted to your proof is useful.

Proof by omission :
'The reader may easily supply the details.'
'The other 253 cases are analogous.'

Proof by obfuscation :
A long plotless sequence of true and/or meaningless syntactically related statements.

Proof by funding :
How could three different government agencies be wrong ?

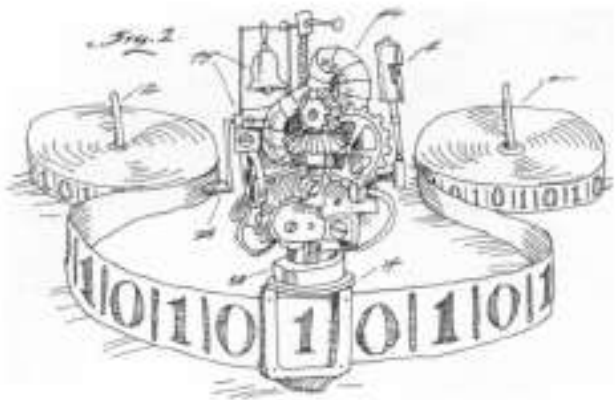
Proof by eminent authority :
'I saw Karp in the elevator and he said it was probably NP-complete.'

...
`https://users.cs.northwestern.edu/~riesbeck/proofs.html`

→ *Mon exposé risque d'utiliser un mix de ces techniques...*

(Partie I)

Décidabilité



Les limites de la connaissance humaine ?

“Ignoramus et ignorabimus” (1872)

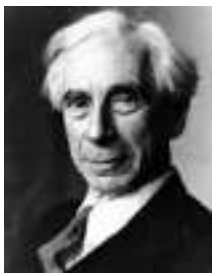
Emil du Bois-Reymond (1818-1896)
(physiologiste)

“Wir müssen wissen, wir werden wissen” (1930)

David Hilbert (1862-1943)
(mathématicien)

La crise des fondements

“Wir müssen wissen, wir werden wissen”



Pas avec la théorie des ensembles !

Soit $R = \{x \mid x \notin x\}$, alors $R \in R \iff R \notin R$

Bertrand Russell (1872-1970)

Quel que soit le système, en fait !

Tout système axiomatique est soit incomplet, soit incohérent.

Kurt Gödel (1906-1978)

Et même des questions naturelles !

Comme savoir si un algorithme s’arrêtera.

Alan Turing (1912-1954)

Traitement de l'information

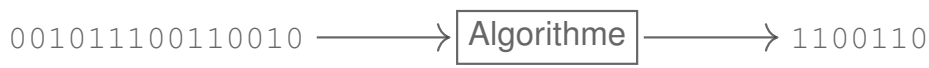
Tri de données :

Calcul d'itinéraire :

Reconnaissance d'image :

Mathématiques :

Plus généralement

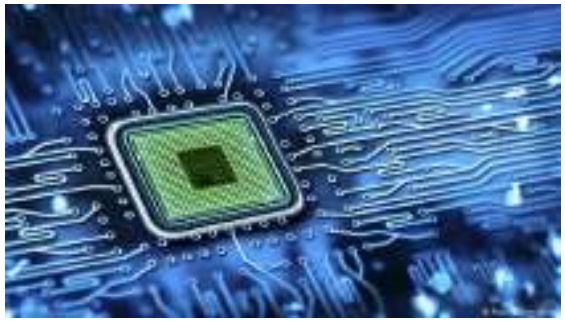


Un algorithme prend en entrée une suite de **symboles** et produit en sortie une autre suite de **symboles**. Il réalise un **traitement**.

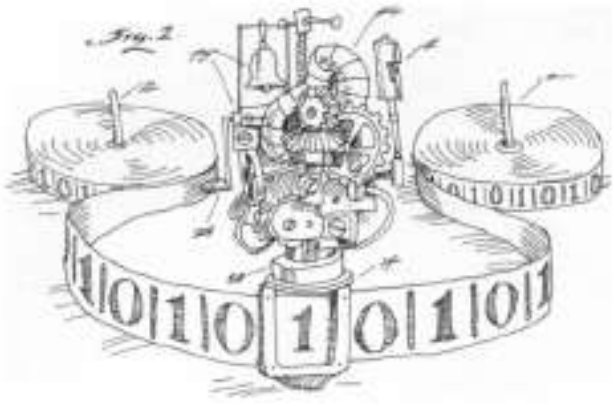
Problème de **décision** : cas particulier dont la sortie est `1` ou `0` (Oui ou Non).

Mais en fait, qu'est-ce qu'un algorithme ?

Programme ? Algorithme ?



Machine de Turing (1936)



230 A. M. TURING [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

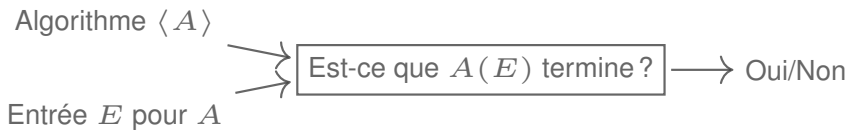
[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable numbers, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are,

→ Modèle **mathématique** d'algorithme, présumément capable de représenter **n'importe quel traitement** d'information physiquement réalisable (Conjecture de Church-Turing).

Le problème de l'arrêt

Problème de l'arrêt (Halting problem) :

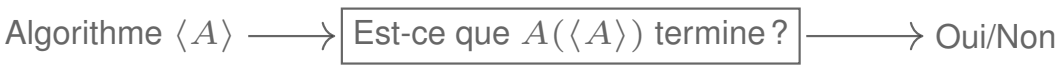


Version universelle (pour info) :

Existe-t-il un algorithme $A_H(\langle A \rangle, E)$ capable de répondre à cette question dans tous les cas ?

Raisonnement par l'absurde

Supposons que A_H existe. On peut alors facilement créer un algorithme A_P qui résoud le cas particulier suivant :



$A_P(\langle A \rangle)$:
  Si $A_H(\langle A \rangle, \langle A \rangle)$ dit oui :
    Dire oui
  Sinon
    Dire non

On peut aussi (soyons tordus) utiliser A_H pour créer un algorithme A_T qui prend en entrée un algorithme $\langle A \rangle$ et adopte le comportement opposé de $A(\langle A \rangle)$, à savoir boucler si $A(\langle A \rangle)$ termine et terminer si $A(\langle A \rangle)$ boucle.

$A_T(\langle A \rangle)$:
  Si $A_H(\langle A \rangle, \langle A \rangle)$ dit oui :
    Boucler à l'infini
  Sinon
    Terminer

Que se passe-t-il si l'on exécute $A_T(\langle A_T \rangle)$?

- Si A_H nous dit que $A_T(\langle A_T \rangle)$ termine, alors $A_T(\langle A_T \rangle)$ boucle
- Si A_H nous dit que $A_T(\langle A_T \rangle)$ boucle, alors $A_T(\langle A_T \rangle)$ termine

$\rightarrow A_H$ ne peut pas exister.

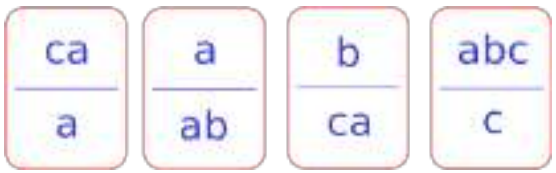
(et toc !)

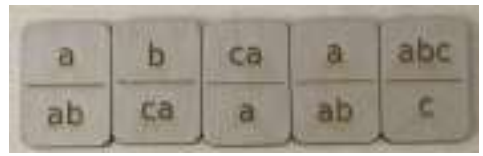
Autres problèmes indécidables

Tous ces problèmes sont indécidables **dans le cas général**. Cela n'empêche pas des cas particuliers d'être décidable, bien sûr.

- Résolution d'équations diophantiennes
  (équations polynomiales à coefficients entiers et solutions entières)
  P. ex. : $x^2 + y^2 = 3z^2$.

- Problème de correspondance de Post





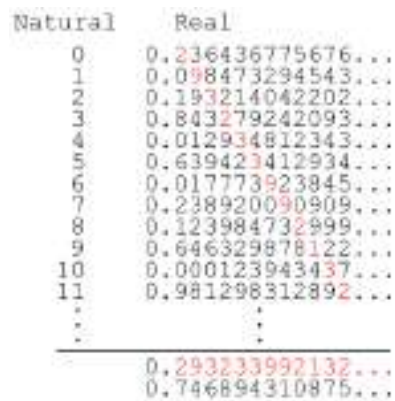
- Décider si un algorithme répond toujours OUI.
- Nombreuses propriétés de programme... (Théorème de Rice)

En fait :

- Le nombre d'**algorithmes** a la cardinalité des entiers ($\aleph_0$)
- Le nombre de **problèmes** a la cardinalité des réels ($2^{\aleph_0}$)

$\rightarrow$ L'écrasante majorité des problèmes est indécidable.

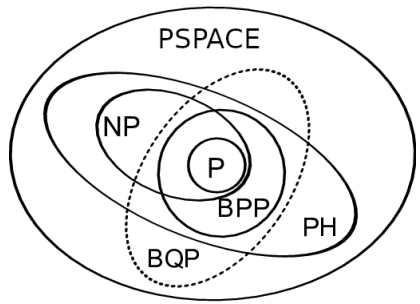
(fort heureusement, beaucoup n'ont aucun intérêt)



Argument diagonal (Cantor)

(Partie 2)

Complexité algorithmique



(désormais, on reste dans le monde décidable)

La lettre de Gödel à Von Neumann (1956)

Princeton, 20/III. 1956

Lieber Herr v. Neumann!

Ich habe mit größtem Bedauern von Ihrer Erkrankung gehört. Die Nachricht kam mir ganz unerwartet. Vorgestern hatte mir Bernays noch im Sommer von einem Schwächeanfall erzählt, den Sie einmal hatten, aber er meinte damals, dass dem keine größere Bedeutung beizumessen sei. Wie ich höre, haben Sie sich in den letzten Monaten einer radikalen Behandlung unterzogen, welche Ihnen, wie mir scheint, den gewünschten Erfolg hatte. Ich hoffe u. wünsche Ihnen, dass Ihr Zustand sich bald noch weiter bessert u. dass die neuesten Errungenschaften der Medizin, wenn möglich, zu einer vollständigen Heilung führen mögen.

Da Sie sich, wie ich höre, jetzt häufig [illegible] möchte ich mir erlauben, Ihnen über ein mathematisches Problem zu schreiben, über das mich Ihre Ansicht sehr interessieren würde: Man kann offenbar leicht eine Turingmaschine konstruieren, welche von jeder Formel F des engeren Funktionalkalküls u. jeder natürl. Zahl n zu entscheiden gestattet, ob F einen Beweis der Länge n hat [Länge = Anzahl der Symbole]. Sei $\psi(F,n)$ die Anzahl der Schritte, die die Maschine dazu benötigt u. sei $\varphi(n) = \max_F \psi(F,n)$. Die Frage ist wie rasch $\varphi(n)$ für eine optimale Maschine wächst. Man kann zeigen $\varphi(n) \geq Kn$. Wenn es wirklich eine Maschine mit ~~$\varphi(n) \sim Kn$~~ $\varphi(n) \sim Kn$ (oder auch nur $\sim Kn^2$) gäbe, hätte das Folgerungen von der größten Tragweite. Es würde nämlich offenbar bedeuten, dass man trotz der Unlösbarkeit des Entscheidungsproblems die Denkarbeit der Mathematiker bei ja-oder-nein Fragen vollständig durch Maschinen ersetzen könnte. [illegible]

Gödel's letter to Von Neumann (1956)

I would like to allow myself to write you about a mathematical problem, of which your opinion would very much interest me. One can obviously construct a Turing machine, which for every formula F in first order predicate logic and every natural number n, allows one to decide if there is a proof of F of length n (length = number of symbols). Let $\Psi(F, n)$ be the number of steps the machine requires for this and let $\varphi(n) = \max_F \Psi(F, n)$.

The question is how fast $\varphi(n)$ grows for an optimal machine. (...) If there really were a machine with $\varphi(n) \sim n$ (or even $\sim n^2$), this would have consequences of the greatest importance. Namely, it would mean that (...) the mental work of a mathematician concerning Yes-or-No questions could be completely replaced by a machine.

(...) It would be interesting to know, for instance, the situation concerning the determination of primality of a number and how strongly in general the number of steps in finite combinatorial problems can be reduced with respect to simple exhaustive search.

Et voilà le domaine de la complexité algorithmique lancé !

Complexité algorithmique ?

Ressources nécessaires pour résoudre un problème (décidable).

Quel type de ressources ?

- Temps (nombre d'opérations)
- Espace (quantité de mémoire)
- Impact du non-déterminisme ?
- Impact de l'aléa ?
- ...

Point de vue **asymptotique**

- **Évolution** de ces quantités en fonction de la **taille** n de l'entrée, quand $n \to \infty$
- Notations $O(\cdot), \Omega(\cdot), \Theta(\cdot)$ (ignore les facteurs constants et les termes dominés)

  Intuition $\leq \quad \geq \quad =$ Ex : $3n^2 + 5n + 4 = \Theta(n^2)$
- Quelques adjectifs :

Constant	$\Theta(1)$	Quadratique	$\Theta(n^2)$
Logarithmique	$\Theta(\log n)$	Exponentiel	$\Theta(2^n)$
Linéaire	$\Theta(n)$	Factoriel	$\Theta(n!)$
Quasi-linéaire	$\Theta(n \log n)$	Polynomial	$O(n^c) = n^{O(1)}$

En général, on s'intéresse au **pire cas** (maximum sur toutes les instances possibles du problème).

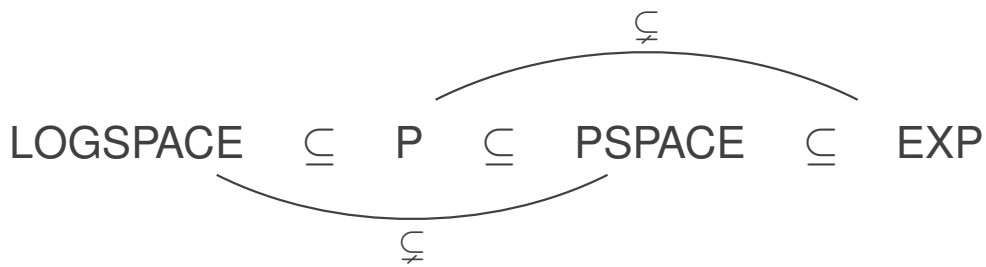
L'espace et le temps

Classes génériques de problèmes

- TIME$(f(n))$: Problèmes que l'on peut résoudre en temps $O(f(n))$.
- SPACE$(f(n))$: Problèmes que l'on peut résoudre en espace $O(f(n))$.

Cas particuliers les plus connus

Nom commun	Problèmes résolubles en...	Définition
LOGSPACE	espace logarithmique	SPACE$(\log n)$
P	temps polynomial	TIME$(n^{O(1)})$
PSPACE	espace polynomial	SPACE$(n^{O(1)})$
EXP	temps exponentiel	TIME(2^n)



La plus importante est la classe P

Problèmes que l'on peut résoudre "rapidement" (en temps $n^{O(1)}$). (+ robuste / composable / appropriée)

Classes de problèmes NP et coNP

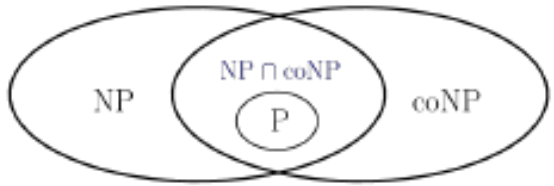
Plusieurs définitions équivalentes, les plus simples :

NP : $\exists$ **preuve courte** que la réponse est OUI (si la réponse est OUI) – certificat positif

coNP : $\exists$ **preuve courte** que la réponse est NON (si la réponse est NON) – certificat négatif

Preuve courte = **vérifiable** en temps polynomial

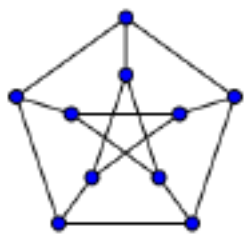
Observation : P $\subseteq$ NP et P $\subseteq$ coNP (l'algorithme lui-même permet de vérifier, sans autre certificat)

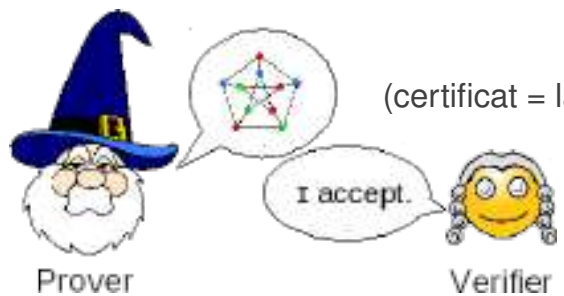


Quelques problèmes dans NP (présumément pas dans P) : 3-COLORING, CLIQUE, TSP, FACTORISATION, SAT, ...

Exemple : 3-COLORING

Ce graphe peut-il être colorié avec 3 couleurs ?



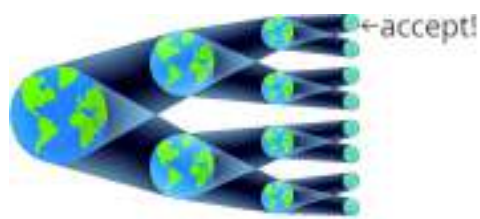


(certificat = la coloration elle-même)

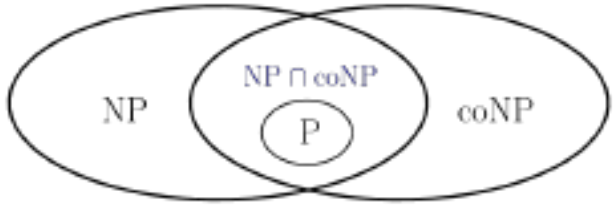
Définition historique

NP = Non-deterministic Polynomial time

Intuition : capacité à "**deviner**" le certificat (qu'il suffit alors de vérifier).



Intermède (exercices)



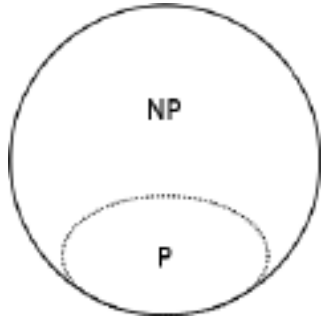
Classer les problèmes de décision suivants

- Entrée : Une liste de mot.
  Question : est-elle triée alphabétiquement ?
- Entrée : Un graphe.
  Question : est-il connexe ?
- Entrée : Un nombre N.
  Question : est-il premier ?
- Entrée : Deux nombres N et k.
  Question : N admet-il un facteur $\leq k$?
- Entrée : Deux graphes G_1 et G_2.
  Question : sont-ils isomorphes ?
- Entrée : Un graphe G et un entier k.
  Question : G contient-t-il un sous-graphe complet de taille k ?
- Entrée : Un plateau d'échecs de taille $N \times N$ (règles généralisées).
  Question : Y a-t-il une stratégie gagnante pour le joueur 1 ?
- Entrée : Un énoncé mathématique.
  Question : est-il vrai ?

P versus NP

Les questions faciles à vérifier sont-elles faciles à résoudre ?

(Est-ce que P $=$ NP ?)



Importance de la question

- La majorité des problèmes informatiques de la vie courante sont dans NP.
  Si P $=$ NP, on peut résoudre tous ces problèmes rapidement.
- Serait-ce une bonne nouvelle ? Oui et non (cryptographie).
- Un des 7 “problèmes du millénaire” (fondation Clay, \$1 M / pb), au même titre que la conjecture de Riemann.

Portée philosophique ?

- Tout énoncé mathématique ayant une preuve humainement vérifiable peut-il être résolu “rapidement”.
- Les connaissances que l’on est capable de vérifier sont-elles “facilement” découvrable ?
- Je sais reconnaître une symphonie, donc j’aurais pu la composer moi-même ?
- L’intuition est-elle automatisable ?
- *etc.*

bémols : formalisation + quid de $O(n^{100})$?

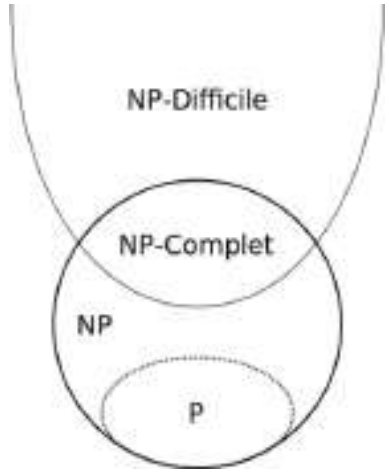
Statut de la question ? À ce jour, on ne connaît pas la réponse.

L’écrasante majorité des experts pensent que P $\neq$ NP.

NP-complétude

Difficulté et complétude

- NP-difficile : problèmes qui sont **au moins aussi difficile** que tout problème de NP
  Cela se montre par des **réductions** entre problèmes.
- NP-complet : à la fois dans NP et NP-difficile



Comment montrer qu'un problème est NP-difficile ?

$\rightarrow$ trouver un problème déjà NP-difficile et le réduire à ce problème.

Exemples de problèmes NP-complets

- SAT : $(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_3 \vee \overline{x_4}) \vee \ldots$ (Cook, Levin'71)
- 3-COLORING, CLIQUE, SET COVER, HAMILTONIAN CYCLE, TSP,
- Des milliers d'autres problèmes ...

Si **un seul** de ces problèmes est résoluble en temps polynomial, alors ils le sont **tous** et P $=$ NP.

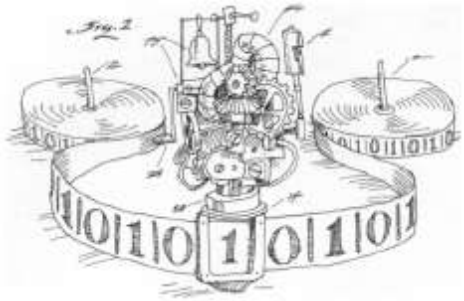
Si un seul de ces problèmes requiert un temps super-polynomial, alors P $\neq$ NP.

Rappel important

Cette théorie s'intéresse au **pire cas**, c'est à dire aux instances les plus dures à résoudre. Beaucoup de ces problèmes sont résoluble dans certains cas pratiques.

Les instances issues du monde réel sont souvent sympatiques.

Et l'IA dans tout ça ?



V.S.

Toutes les limitations évoquées s'appliquent à l'IA, sans distinction.

De nombreux problèmes, y compris NP-complets, sont **faciles dans le cas moyen**, l'IA (apprentissage profond) peut en théorie les résoudre dans la plupart des cas.

L'IA est particulièrement adaptée pour les problèmes difficiles à formaliser/définir.

Les problèmes qui sont **difficiles dans le cas moyen** resteront inaccessibles pour l'IA. Présumément, FACTORING en fait partie.

Toutefois, il est possible que l'IA nous aide à découvrir des algorithmes plus efficaces que ceux que l'on connaît, même pour ces problèmes. Aucun résultat théorique n'exclut que l'IA surpasse les algorithmiciens !

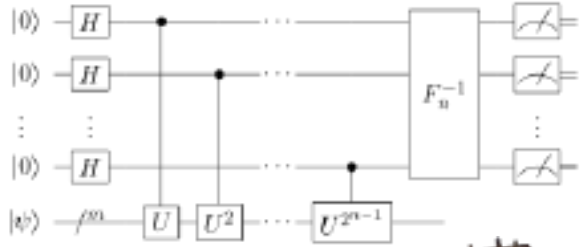
Et le quantique ?

BPP : *Bounded-error probabilistic polynomial time*

- Problèmes résolubles en temps polynomial par un **algorithme probabiliste** (peut tirer à pile ou face) avec une probabilité d'erreur strictement inférieure à 1/2

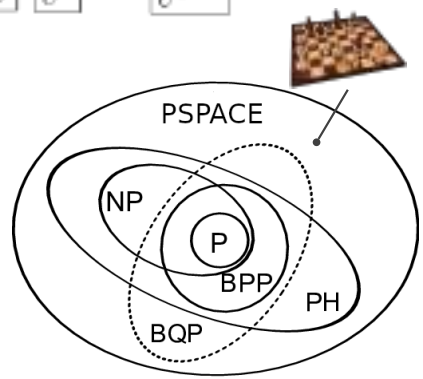
BQP : *Bounded error quantum polynomial time*

- Problèmes résolubles en temps polynomial par un **ordinateur quantique** avec une probabilité d'erreur strictement inférieure à 1/2



Que sait-on ?

- P $\subseteq$ BPP $(0 < 1/2)$
- BPP $\subseteq$ BQP (non-réversibilité simulable en temps poly)
- BQP $\subseteq$ PSPACE (Bernstein et Vazirani'97)
- FACTORING $\in$ BQP (Shor'94)
- Quid de BQP *versus* NP ? (pressentie incomparables)



(structure pressentie)

Un ordinateur quantique peut-il résoudre des problèmes NP-complets ?

$\rightarrow$ C'est très improbable (invaliderait des conjectures très vraisemblables).

Conclusion

L'informatique peut-elle tout faire ?

- Problèmes indécidables
- Problèmes dont la complexité est trop importante
- L'IA hérite des mêmes limitations
- Le quantique va possiblement plus loin (mais a des limitations aussi)
- De nombreux problèmes (mêmes difficiles) sont résolubles dans le cas moyen

Autres pistes de discussions

- Les humains sont-ils soumis à la thèse de Church-Turing ?
- Peuvent-ils puiser dans d'autres sources ? (machines à oracles / théorie de Penrose)
- Ces résultats sont plausiblement les mêmes à l'autre bout de l'univers

Merci !