

Algorithmique M1

2013--2014

A

.

D

G

P

B

A

:

F ,

CFC.

C

(U -F)

F

C

+

TD

: ,
, ...
(P R)

C : 9 30 ! 11 30

TD1: R. M , 8 30! 10 30;

TD2: S. L , 10 30! 12 30;

@
:// . . - - . / / 1 .

C

$$U + \\ CC = (1 + 1)$$

P

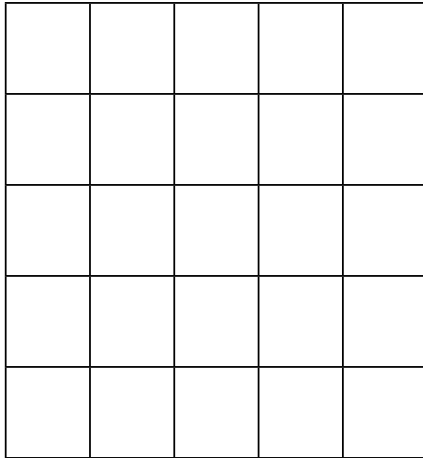
P , ...

R
CORRECTS EFFICACES !

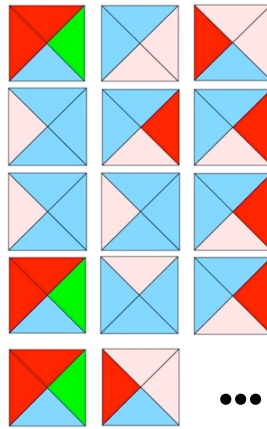
C !

Un puzzle...

Une grille 5x5



25 dominos



Questions/objections

Sur le problème du puzzle:

- les ordinateurs sont de plus en plus rapides...
- les informaticiens sont incompetents... et incapables de trouver un bon algorithme.
- nous n'avons pas montré qu'il n'existe pas d'algorithme efficace.
- le Puzzle est très très spécial... et sans intérêt !

Puzzle

On essaie toutes les possibilités ?

25 cartes à placer sur 25 cases:

- 25 possibilités pour la première,
- 24 pour la seconde,
- 23 pour la troisième,
- ...

$25 \times 24 \times 23 \times \dots \times 2$ possibilités !

Avec un ordinateur qui évalue 1 milliard de cas par seconde, il faudrait ...

490 millions d'années !

(25! s'écrit avec 26 chiffres...)

Attendre un ordinateur plus rapide ?

En 10 ans, la vitesse des ordinateurs a été multipliée par 50...

Supposons qu'aujourd'hui, on puisse résoudre un problème de taille K en une heure...

Si l'algorithme a une complexité n , alors...

- un ordinateur 100 fois plus rapide, résoudra des pb de taille $100 \times K$.
- un ordinateur 1000 fois plus rapide, résoudra des pb de taille $1000 \times K$.

Si l'algorithme a une complexité n^2 , alors...

- un ordinateur 100 fois plus rapide, résoudra des pb de taille $10 \times K$.
- un ordinateur 1000 fois plus rapide, résoudra des pb de taille $32 \times K$.

Si l'algorithme a une complexité 2^n , alors...

- un ordinateur 100 fois plus rapide, résoudra des pb de taille $7 + K$.
- un ordinateur 1000 fois plus rapide, résoudra des pb de taille $10 + K$.

La course est perdue d'avance !

Encore un exemple...

Imaginons un véhicule dont la consommation d'énergie est une fonction linéaire (n) ou exponentielle (2^n) de la distance à parcourir.

Si l'on peut faire 50 kilomètres avec un réservoir V , alors un réservoir 1000 fois plus gros, permettra de parcourir...

- 50000 km si la consommation est linéaire, et
- 60 km si la consommation est exponentielle !

Les problèmes NP-complets

Tous ces problèmes sont aussi difficiles les uns que les autres.

Si on a un algorithme efficace pour un, on peut l'adapter pour les autres.

Aujourd'hui, on ne sait pas si il existe des algorithmes efficaces pour tous ces problèmes !

Mais on sait que d'autres problèmes sont durs...

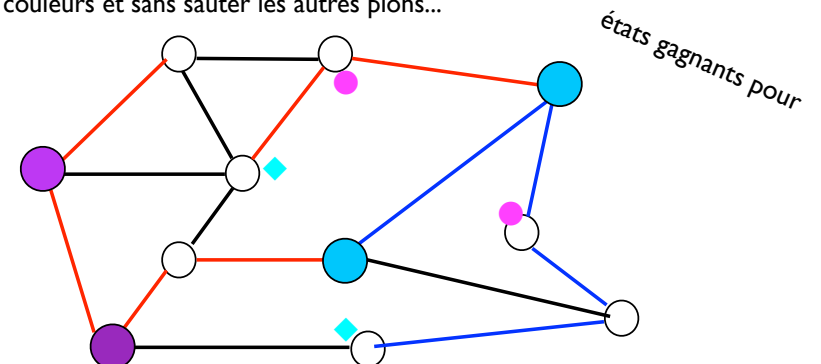
Le puzzle n'est pas isolé !

Beaucoup de problèmes lui ressemblent...

- Recherche de chemins hamiltoniens (qui passent une et une seule fois par chaque sommet) dans un graphe
- Problème du voyageur de commerce
- Planification (emploi du temps)
- Colorier une carte avec 3 couleurs (2: facile, 4: toujours possible)
- ...

De "vrais" problèmes difficiles

2 joueurs avec 2 pions chacun (◆ et ●) cherchent à atteindre des états gagnants en déplaçant leurs pions en suivant les couleurs et sans sauter les autres pions...



De “vrais” problèmes difficiles

Question: est-ce que le joueur 1 (ou 2) a une stratégie gagnante ?

Ce “problème” est exponentiel !

(Il existe un $K > 1$ tel que tout algorithme pour ce problème prend un temps qui augmente au moins aussi vite que K^n où n est le nombre d'intersections.)

Et au delà: doublement exponentiel (2^{2^n}), triplement...

Problème de correspondance de Post

Un ensemble de type de cartes:

abb	a	bab	baba	aba
bbab	aa	ab	aa	a

Peut-on écrire le même mot en haut et en bas ?

a	abb	abb	baba	abb	aba
aa	bbab	bbab	aa	bbab	a

Oui !

Toujours plus dur...

On a vu des problèmes que l'on peut résoudre efficacement...

Et d'autres, beaucoup plus difficiles.

Il y a même des problèmes **sans algorithme**.

Problème de correspondance de Post

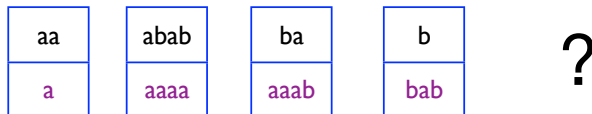
bab	bb	a
a	ba	abb

?

Il n'y a pas de solution !!

(sur aucune carte, les deux mots ne terminent par la même lettre !)

Problème de correspondance de Post



Oui !

Mais la plus petite solution utilise 781 cartes !!
(voir [PCP@home contest](#))

Le problème PCP est indécidable...

Problème de l'arrêt

1. Tant que $x \neq 1$ Faire :
2. Si x est pair Alors $x := x/2$
3. Sinon $x := 3x + 1$
4. Stop

Essai avec 9:

9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

On ne sait pas si cela termine
pour tout entier !!

Problème de l'arrêt

Est-ce que ce programme termine ?

1. Tant que $x \neq 1$ Faire $x := x-2$
2. Stop

Termine si x est impair.

Problème de l'arrêt

Donnée: un programme P , une entrée x

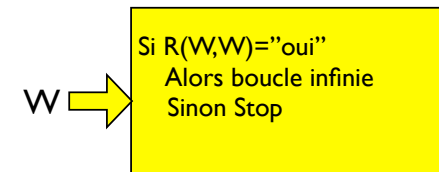
Question: est-ce $P(x)$ s'arrête ?

Ce problème est indécidable.

Soit R une procédure qui résout le problème.

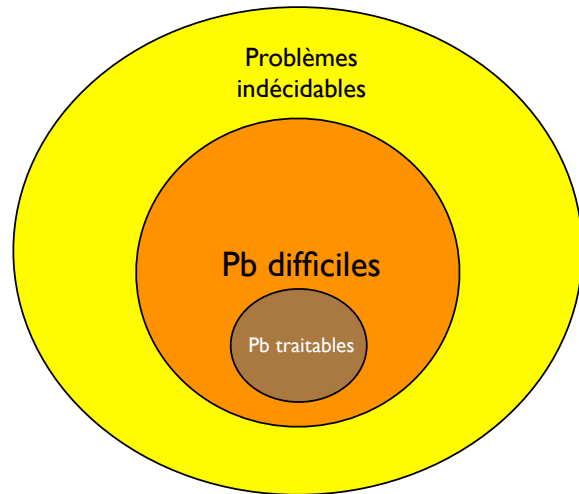
($R(P,x)$ = "oui" ssi $P(x)$ s'arrête)

Soit P le programme suivant:



Que répond $P(P)$? R n'existe pas !!!

Vue globale



E

,

!

Classes de complexité

- Dans chaque groupe, il y a de très nombreuses **classes de complexité** contenant des problèmes "aussi difficiles les uns que les autres".
- Etant donné un problème, chercher :
 - des algorithmes (leurs coûts en temps et en espace)
 - sa complexité (ou une borne inférieure)

PUB !

Allez au cours de calculabilité et complexité !

E

U

E

:

,

8	16	24	32	40	0	5	-9	4	2	3	78	7	24	6	9	-18	7	2
---	----	----	----	----	---	---	----	---	---	---	----	---	----	---	---	-----	---	---

:= 140

= 115



L

...

8	0	10	4	19	40	0	5	9	14	2	3	78	7	2	6	9	8	7	2
---	---	----	---	----	----	---	---	---	----	---	---	----	---	---	---	---	---	---	---

?

... 223

-8	-10	-10	-4	-19	-40	-10	-5	-9	-14	-2	-3	-78	-7	-24	-6	-9	-18	-7	-2
----	-----	-----	----	-----	-----	-----	----	----	-----	----	----	-----	----	-----	----	----	-----	----	----

?

0 ! (. . - 0)

L

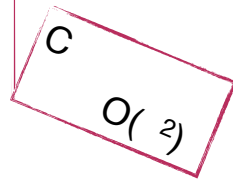
D : T 0.. -1

R : M , 0 ≤ , ≤ -1

, = $\sum_{\epsilon \in \mathcal{E}} T$:

: , +1, ...,

A 2



```
def algo2(T[0..n-1]):
    maxsofar = 0
    for i = 0,...,n-1:
        sum = 0
        for j = i,...,n-1:
            sum += T[j]
            maxsofar = max(maxsofar, sum)
    return maxsofar
```



-1

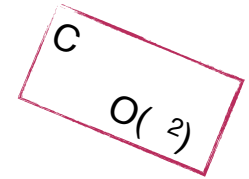
-

-1

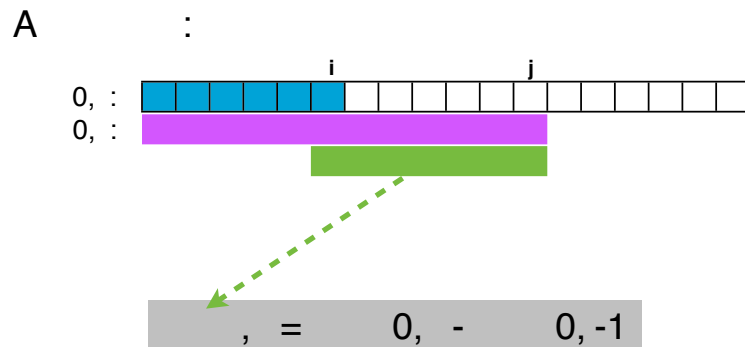
A 3

```
def algo3(T[0..n-1]):
    sum[i] = 0  ∀ i = -1, 0, ..., n
    for i = 0..n-1:
        sum[i] = sum[i-1] + T[i]
    maxsofar = 0
    for i = 0..n-1:
        for j = i..n-1:
            sumij = sum[j] - sum[i-1]
            maxsofar = max(maxsofar, sumij)
    return maxsofar
```

sum[i] = 0,

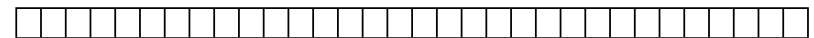


A 3

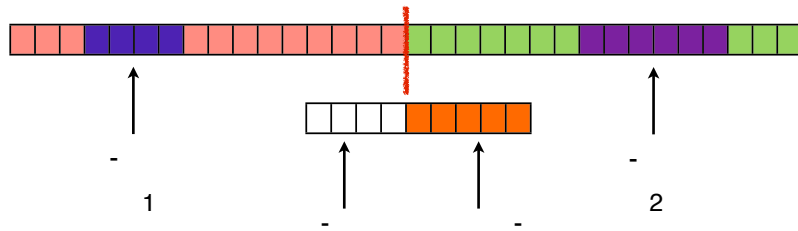


, = 0, - 0, -1

A 4



A 4



$$= (1, 2, +)$$

C : - -1.
C : - .

```
def algo4(T,l,u) :
    if (l>u) : return 0
    if (l==u) : return max(0,T[l])
    m = (l+u)/2

    lmax = sum = 0
    for i = m ... l : // l ≤ m
        sum += T[i]
        lmax = max(lmax,sum)
    rmax = sum = 0
    for i = m+1 ... u : // m ≤ u
        sum += T[i]
        rmax = max(rmax,sum)

    return max(lmax+rmax, algo4(T,l,m),algo4(T,m+1,u))
```

A 4

B

A 1	A 2 A 3	A 4	A 5
O(3)	O(2)	O(.)	O()
...		- -	

- -

?

A 80 ?



A

nm

cs