

Informatique et Programmation

Cours 9

Jean-Jacques Lévy

`jean-jacques.levy@inria.fr`

`http://jeanjacqueslevy.net/prog-py`

Plan

- algorithme glouton (**Dijkstra**)
- algorithme glouton (marche du cavalier)
- exploration exhaustive (sac à dos)
- exploration exhaustive (les 8 reines)
- programmation dynamique (plus courts chemins, **Warshall**)
- programmation dynamique (plus longue chaîne commune)

dès maintenant: **télécharger Python 3 en** `http://www.python.org`

Recap

- mots clés en Python (déjà vus en rouge)

```
>>> help()
help> keywords
```

Here is a list of the Python keywords. Enter any keyword to get more help.

False	class	from	or
None	continue	global	pass
True	def	if	raise
and	del	import	return
as	elif	in	try
assert	else	is	while
async	except	lambda	with
await	finally	nonlocal	yield
break	for	not	

Python ++

- rupture de boucle

- break arrête la boucle
- continue saute à l'itération suivante

```
for i in range (10):  
    if i * 2 == 4 :  
        break  
    print (i, end = " ")
```

➡ 0 1

```
for i in range (10):  
    if i * 2 == 4 :  
        continue  
    print (i, end = " ")
```

➡ 0 1 3 4 5 6 7 8 9

- instruction vide

- pass ne fait rien

```
if 4 * 2 == 4 :  
    pass  
else :  
    print ("8 is not 4")
```

➡ 8 is not 4

Python ++

- traitement des exceptions

- try: début d'un bloc avec exception possible
- except IOError: récupère l'exception IOError
- except: récupère toutes les exceptions
- finally: pour le traitement normal **et** le traitement exceptionnel

- alias de module

- as déclare un alias pour un module (par exemple si le nom est trop long)

```
def lire_lignes (nom) :  
    try:  
        f = open (nom, 'r')  
        return f.read().splitlines()  
    except IOError:  
        print("Fichier '%s' inexistant." % nom)  
lire_lignes('abc')
```

➡ Fichier 'abc' inexistant.

```
import random as r  
r.choice(['a', 'b', 'c'])
```

➡ 'a'

Recap

- mots clés en Python (déjà vus en rouge)

```
>>> help()
help> keywords
```

Here is a list of the Python keywords. Enter any keyword to get more help.

False	class	from	or
None	continue	global	pass
True	def	if	raise
and	del	import	return
as	elif	in	try
assert	else	is	while
async	except	lambda	with
await	finally	nonlocal	yield
break	for	not	

Devoir maison

- question 1

```
def nb_base (c, s) :  
    r = 0  
    for ch in s :  
        if c == ch :  
            r = r + 1  
    return r
```

- question 2

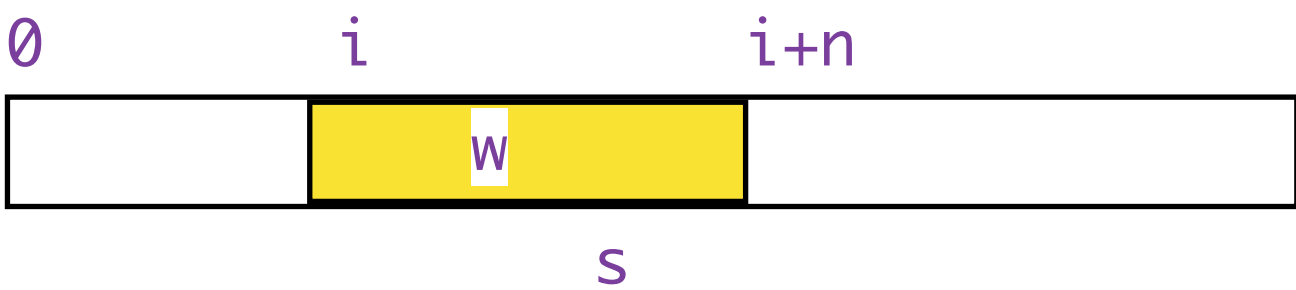
- question 3

```
def eq_seq_at (w, i, s) :  
    n = len(w)  
    for j in range (n):  
        if w[j] != s[i+j] :  
            return False  
    return True
```

- question 4

```
def nb_seq (w, s) :  
    r = 0  
    for i in range (len(s) - len(w) + 1) :  
        if eq_seq_at (w, i, s) :  
            r = r + 1  
    return r
```

- question 5



Détection de virus dans l’ADN

Projet du fin du cours Algorithmes et Programmation

École EEEA – 2022

L’acide désoxyribonucléique (ADN) est une séquence de plusieurs milliards de bases adénine (A), guanine (G), cytosine (C), and thymine (T). Voici un exemple de bout d’ADN humain :

```
TATTTACCATATCAGATTCACATTCAGTCCICAGCAAAATGAAGGGCTCCATTTTCACTCIGTTTTTATT  
TATTTACCATATCAGATTCACATTCAGTCCICAGCAAAATGAAGGGCICCATTTTCACTCTGTTTTTATT  
CTCTGTCCTATTGCCATCTCAGAAGTGCGGAGCAAGGAGTCTGTGAGACTCTGTGGGCTAGAATACATA  
CGGACAGTCATCTATATCIGTGCTAGCTCCAGGTGGAGAAGGCATCAGGAGGGGATCCCICAAGCTCAGC  
AAGCTGAGACAGGAAACTCCTTCCAGCICCCACATAAACGTGAGTTTTCTGAGGAAAATCCAGCGCAAAA  
CCTTCCGAAGGTGGATGCCTCAGGGGAAGACCGTCITTGGGGIGGACAGATGCCACTGAAGAGCTTIGG  
AAGTCAAAGAAGCATTTCAGTGATGTCAAGACAAGATTTACAACTTTGTGTTGCACTGATGGCTGTTCCA  
TGACTGATTTGAGTGCICTTTGCTAAGACAAGAGCAAATACCCAATGGGTGGCAGAGCTTTATCACATGT  
TTAATTACAGTGTTTTACTGCCTGGTAGAACACTAATATTGTGTTATTAATAATGATGGCTTTTGGGTAGG  
CAAACTICTTTTCTAAAAGGTATAGCIGAGCGGTIGAAACCACAGIGATCICTATTTTCICCCTTIGCC  
TAAAAATGCTATAAACCA
```

En Python, on représentera les bouts d’ADN par des chaînes de caractères ’A’, ’T’, ’G’, ’C’. Par exemple `s = ’TATTTACCATATCAGAT’`. Rappel : une expression Python sur plusieurs lignes s’écrit en finissant chaque ligne (sauf la dernière) par le caractère `\`.

- ****
- Question 1

Écrire la fonction `nb_base(c, s)` qui compte le nombre de base `c` dans le bout d’ADN `s`.
- Question 2

Quels sont les nombres de base A, C, T, G dans l’exemple d’ADN précédent.

- ****
- On recherche les nombres de motifs dans des bouts d’ADN. Par exemple, on compte 10 motifs CTA dans l’exemple d’ADN précédent.

Question 3

Écrire la fonction `eq_seq_at(w, i, s)` qui répond vrai si le motif `w` figure en position `i` dans `s`, et faux s’il n’y figure pas.

Question 4

Écrire la fonction `nb_seq(w, s)` qui compte le nombre d’occurences du motif `w` dans le bout d’ADN `s`.

Question 5

Quels sont les nombres des motifs GCA et CATCTCAGA dans l’exemple d’ADN précédent.

Devoir maison

- expression sur plusieurs lignes

```
print (2 + \  
      3)
```



```
s = 'TATTTACCATATCAGATTCACATTCAGTCCICAGCAAAATGAAGGGCTCCATTTTCACTCIGTTTTTATT\  
TATTTACCATATCAGATTCACATTCAGTCCICAGCAAAATGAAGGGCICCATTTTCACTCTGTTTTTATT\  
CTCTGTCCTATTTGCCATCTCAGAAGTGCGGAGCAAGGAGTCTGTGAGACTCTGTGGGCTAGAATACATA\  
CGGACAGTCATCTATATCIGTGCTAGCTCCAGGTGGAGAAGGCATCAGGAGGGGATCCCICAAGCTCAGC\  
AAGCTGAGACAGGAAACTCCTTCCAGCICCCACATAAACGTGAGTTTTCTGAGGAAAATCCAGCGCAAAA\  
CCTTCCGAAGGTGGATGCCTCAGGGGAAGACCGTCITTGGGGIGGACAGATGCCCACTGAAGAGCTTIGG\  
AAGTCAAAGAAGCATTTCAGTGATGTCAAGACAAGATTTACAACTTTGTGTTGCACTGATGGCTGTTCCA\  
TGACTGATTTGAGTGCICTTTGCTAAGACAAGAGCAAATACCCAATGGGTGGCAGAGCTTTATCACATGT\  
TTAATTACAGTGTTTTACTGCCTGGTAGAACACTAATATTGTGTTATTAAAATGATGGCTTTTGGGTAGG\  
CAAACTICTTTTCTAAAAGGTATAGCIGAGCGGTIGAAACCACAGIGATCICTATTTTCICCCTTIGCC\  
TAAAAATGCTATAAACC'
```



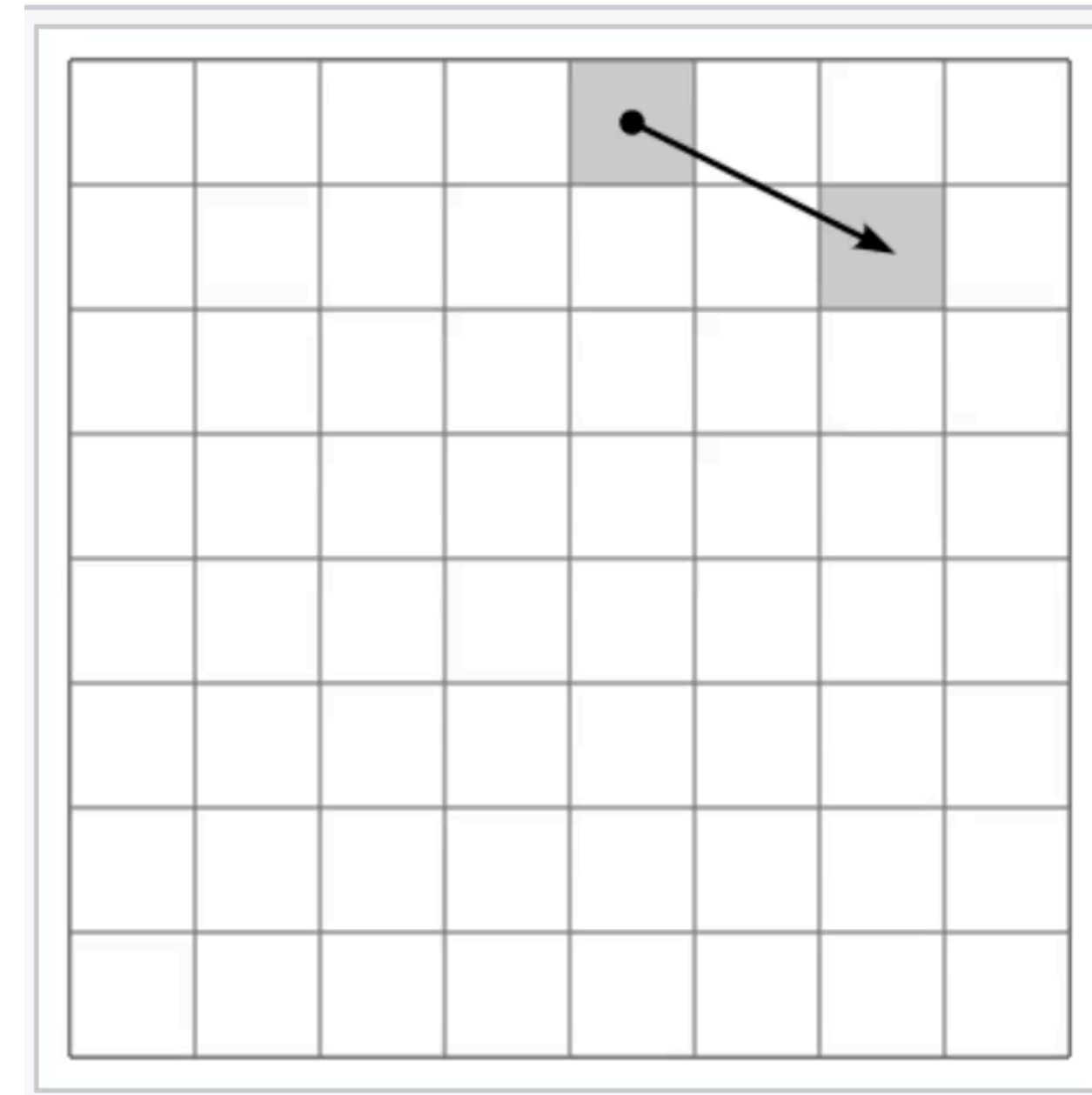
Exploration

on distingue 3 méthodes d'exploration

- algorithmes gloutons
 - un choix local permet d'obtenir la solution globale
- exploration exhaustive
 - on parcourt les solutions globales jusqu'à trouver la bonne solution
 - retours arrière possibles (*backtracking*)
- programmation dynamique
 - on mémorise tous les résultats partiels pour obtenir la solution globale
 - demande de la mémoire supplémentaire

Algorithme glouton

- plus court chemin dans un graphe (**Dijkstra**)
[on cherche le minimum local — cf cours 8]
- arbre de recouvrement minimal dans un graphe non-orienté valué (**Kruskal**, **Prim**)
- allocation de ressources, etc
- marche du cavalier pour couvrir toutes les cases d'un échiquier



Marche du cavalier

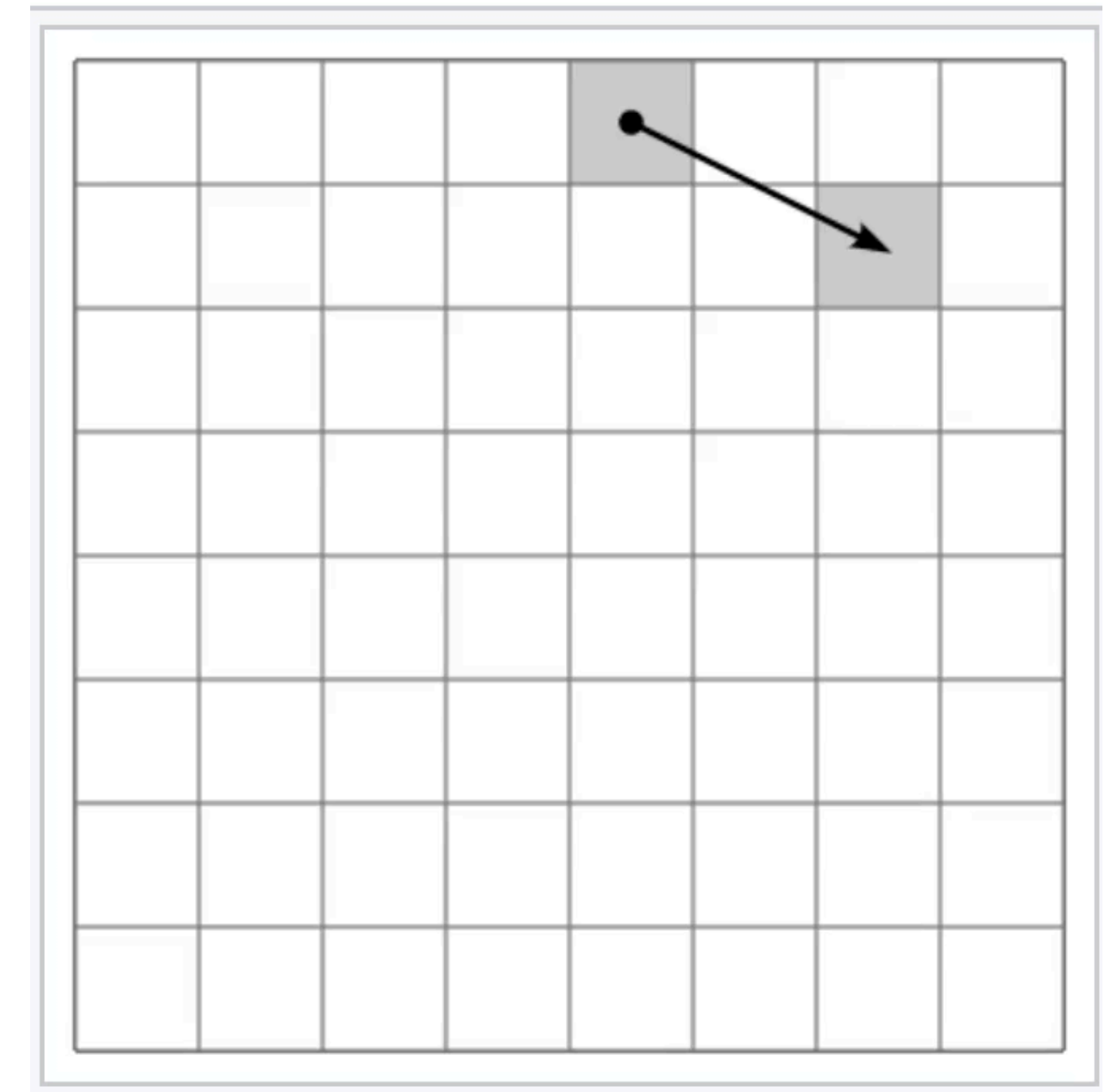
- marche du cavalier pour couvrir toutes les cases d'un échiquier
- on se déplace vers la case où il y aura le moins de déplacements possibles pour le cavalier

```
LIBRE = -1
dX = [ 2, 1, -1, -2, -2, -1, 1, 2 ]
dY = [ 1, 2, 2, 1, -1, -2, -2, -1 ]
import sys; infini = sys.maxsize

def marche (a, i, j, k) :
    a[i][j] = k
    (i1, j1) = caseMinCoupsJouables (a, i, j)
    if (i1, j1) == (i, j) :
        return k
    else:
        return marche (a, i1, j1, k+1)

def une_solution (n, i, j) :
    a = new_matrix (n, n, LIBRE)
    if marche (a, i, j, 1) < n*n :
        print ("pas de solution!")
    print_matrix (a)
```


`une_solution (8, 0, 4)`



47	14	61	32	1	16	19	34
64	31	46	15	60	33	2	17
13	48	57	62	45	18	35	20
30	63	42	53	56	59	40	3
49	12	55	58	41	44	21	36
26	29	52	43	54	39	4	7
11	50	27	24	9	6	37	22
28	25	10	51	38	23	8	5

Marche du cavalier

- marche du cavalier pour couvrir toutes les cases d'un échiquier
- on se déplace vers la case où il y aura le moins de déplacements possibles pour le cavalier

```
def jouable (a, i, j) :  
    return 0 <= i < len(a) and \  
           0 <= j < len(a[0]) and \  
           a[i][j] == LIBRE  
  
def nbDeCoupsJouables (a, i, j) :  
    if not jouable (a, i, j) :  
        return infini  
    r = 0  
    for k in range (len(dX)) :  
        if jouable (a, i + dX[k], j + dY[k]) :  
            r = r + 1  
    return r  
  
def caseMinCoupsJouables (a, i, j) :  
    i1 = i; j1 = j;  
    min = infini  
    for k in range(len(dX)) :  
        nk = nbDeCoupsJouables (a, i+dX[k], j+dY[k])  
        if nk < min :  
            i1 = i+dX[k]; j1 = j + dY[k]  
            min = nk  
    return (i1, j1)
```

Conclusion temporaire

- autres problèmes résolus par exploration exhaustive
- exemple de programmation dynamique
- problèmes NP
- exploration et parallélisme