1 Réponses attendues pour la méthode count

Pour la grammaire des arbres :

n	0	1	2	3	4	5	6	7	8	9	10
count(n, Tree)	0	1	1	2	5	14	42	123	429	1430	4862
count(n, Node)	0	0	1	2	5	14	42	123	429	1430	4862
count(n, Leaf)	0	1	0	0	0	0	0	0	0	0	0

Pour la grammaire des mots de Fibonacci :

n	0	1	2	3	4	5	6	7	8	9	10
count(n, Fib)	1	2	3	5	8	13	21	34	55	89	144
count(n, Cas1)	0	2	3	5	8	13	21	34	55	89	144
count(n, Cas2)	0	1	1	2	3	5	8	13	21	34	55
count(n, Vide)	1	0	0	0	0	0	0	0	0	0	0
count(n, CasAu)	0	1	2	3	5	8	13	21	34	55	89
count(n, AtomA)	0	1	0	0	0	0	0	0	0	0	0
count(n, AtomB)	0	1	0	0	0	0	0	0	0	0	0
count(n, CasBAu)	0	0	1	2	3	5	8	13	21	34	55

2 Grammaires

2.1 Alphabet A,B

$S = \mathcal{E} \mid AS \mid BS$
w est un mot de la grammaire A, B si :
— soit w est vide
— soit w est de la forme Au où u est un mot de la grammaire A, B
— soit w est de la forme Bu où u est un mot de la grammaire A, B

2.2 Mots de Dyck

$S = \mathcal{E} \mid (S) \mid S(S)$
w est un mot de Dyck si :
— soit w est vide
— soit w est de la forme (u) où u est un mot de Dyck de Dyck
— soit w est de la forme $u(v)$ où u et v sont des mots de Dyck

2.3 Mots sur l'alphabet A,B qui n'ont pas trois lettres consécutives égales

$S = \mathcal{E} \mid U \mid T$
$U = A \mid AA \mid AT \mid AAT$
$T = B \mid BB \mid BT \mid BBT$
w est un mot de cette grammaire si :
— soit w est vide
— soit w est de la forme A, AA, B, BB

— soit w est de la forme AT ou AAT où T est un mot de la grammaire qui commence par B
— soit w est de la forme BU ou BBU où U est un mot de la grammaire qui commence par A

2.4 Palindromes sur l'alphabet A, B

$S = \mathcal{E} \mid A \mid B \mid ASA \mid BSB$
w est un mot de la grammaire des palindrome sur A, B si :
— soit w est vide
— soit w est de la forme A ou B
— soit w est de la forme AuA où u est un palindrome.
— soit w est de la forme BuB où u est un palindrome.

2.5 Palindromes sur l'alphabet A, B, C

$S = \mathcal{E} \mid A \mid B \mid ASA \mid BSB \mid CSC$
w est un mot de la grammaire des palindrome sur A, B si :
— soit w est vide
— soit w est de la forme A ou B ou C
— soit w est de la forme AuA où u est un palindrome.
— soit w est de la forme BuB où u est un palindrome.
— soit w est de la forme CuC où u est un palindrome.

2.6 Mots sur l'alphabet A,B qui contiennent autant de A que de B

$S = \mathcal{E} \mid aTbS \mid bUaS$
$T = \mathcal{E} \mid aTbT$
$U = \mathcal{E} \mid bTaT$

3 Calcul de la valuation

3.1 Mots de Fibonacci

n	Fib	Cas1	Cas2	Vide	CasAu	AtomA	AtomB	CasBAu
règle	Vide ∪ Cas1	CasAU ∪ Cas2	AtomA ∪ AtomB	$\mathcal{E}$	AtomA*Fib	A	B	AtomB*CasAu
0	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	∞	∞	0	∞	1	1	∞
2	0	∞	1	0	1	1	1	∞
3	0	1	1	0	1	1	1	2
3	0	1	1	0	1	1	1	2

3.2 Mots de Dyck

n	Dyck	Casuu	Vide	AtomLPAR	AtomRPAR	Cas(u	Casu)
règle	Vide ∪ Casuu	Dyck * Cas(u	E	"("	")"	LPAR * Casu)	Dyck * RPAR
0	∞	∞	∞	∞	∞	∞	∞
1	∞	∞	0	1	1	∞	∞
2	0	∞	0	1	1	∞	∞
3	0	∞	0	1	1	∞	1
4	0	∞	0	1	1	2	1
5	0	2	0	1	1	2	1
6	0	2	0	1	1	2	1

3.3 Mots sur l'alphabet A,B

n	AB	AtomA	AtomB	CasAB	Vide	CasAu	CasBu
règle	Vide ∪ CasAB	A	B	CasAu ∪ CasBu	$\mathcal{E}$	AtomA * AB	AtomB * AB
0	∞	∞	∞	∞	∞	∞	∞
1	∞	1	1	∞	0	∞	∞
2	0	1	1	∞	0	1	1
3	0	1	1	1	0	1	1
4	0	1	1	1	0	1	1

3.4 Mots qui n'ont pas trois lettres consécutives égales sur A, B

n	Three	Vide	AtomA	AtomB	AA	BB	S	U	U1
règle	Vide ∪ S	$\mathcal{E}$	A	B	AtomA*AtomA	AtomB*AtomB	U ∪ T	AtomA ∪ U1	AA ∪ U2
0	∞	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	0	1	1	∞	∞	∞	∞	∞
2	0	0	1	1	2	2	∞	1	∞
3	0	0	1	1	2	2	1	1	2
4	0	0	1	1	2	2	1	1	2
5	0	0	1	1	2	2	1	1	2
6	0	0	1	1	2	2	1	1	2

n	U2	AT	AAT	T	T1	T2	BU	BBU
règle	AT ∪ AAT	AtomA * T	AtomA * AT	AtomB ∪ T1	BB ∪ T2	BU ∪ BBU	AtomB * U	AtomB * BU
0	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	1	∞	∞	∞	∞
3	∞	2	∞	1	2	∞	2	∞
4	2	2	3	1	2	∞	2	3
5	2	2	3	1	2	2	2	3
6	2	2	3	1	2	2	2	3

3.5 Palindromes sur A,B

n	Pal	Vide	AtomA	AtomB	S	S1
règle	Vide ∪ S	$\mathcal{E}$	A	B	AtomA ∪ S1	AtomB ∪ S2
0	∞	∞	∞	∞	∞	∞
1	∞	0	1	1	∞	∞
2	0	0	1	1	1	1
3	0	0	1	1	1	1
4	0	0	1	1	1	1
5	0	0	1	1	1	1
6	0	0	1	1	1	1

n	S2	ASA	ASA1	BSB	BSB1
règle	ASA ∪ BSB	AtomA * ASA1	Pal * AtomA	AtomB * BSB1	Pal * AtomB
0	∞	∞	∞	∞	∞
1	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞
3	∞	∞	1	∞	1
4	∞	2	1	2	1
5	2	2	1	2	1
6	2	2	1	2	1

3.6 Palindromes sur A,B,C

n	Pal	Vide	AtomA	AtomB	AtomC	S	S1	S2	S3
règle	Vide ∪ S	$\mathcal{E}$	A	B	C	AtomA ∪ S1	AtomB ∪ S2	AtomC ∪ S3	ASA ∪ S4
0	∞	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	0	1	1	1	∞	∞	∞	∞
2	0	0	1	1	1	1	1	1	∞
3	0	0	1	1	1	1	1	1	∞
4	0	0	1	1	1	1	1	1	∞
5	0	0	1	1	1	1	1	1	2
6	0	0	1	1	1	1	1	1	2

n	S4	ASA	ASA1	BSB	BSB1	CSC	CSC1
règle	BSB ∪ CSC	AtomA * ASA1	Pal * AtomA	AtomB * BSB1	Pal * AtomB	AtomC * CSC1	Pal * AtomC
0	∞	∞	∞	∞	∞	∞	∞
1	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞	∞	∞
3	∞	∞	1	∞	1	∞	1
4	∞	2	1	2	1	2	1
5	2	2	1	2	1	2	1
6	2	2	1	2	1	2	1

3.7 Palindromes sur A,B,C

n	Pal	Vide	AtomA	AtomB	AtomC	S	S1	S2	S3
règle	Vide ∪ S	$\mathcal{E}$	A	B	C	AtomA ∪ S1	AtomB ∪ S2	AtomC ∪ S3	ASA ∪ S4
0	∞	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	0	1	1	1	∞	∞	∞	∞
2	0	0	1	1	1	1	1	1	∞
3	0	0	1	1	1	1	1	1	∞
4	0	0	1	1	1	1	1	1	∞
5	0	0	1	1	1	1	1	1	2
6	0	0	1	1	1	1	1	1	2

n	S4	ASA	ASA1	BSB	BSB1	CSC	CSC1
règle	BSB ∪ CSC	AtomA * ASA1	Pal * AtomA	AtomB * BSB1	Pal * AtomB	AtomC * CSC1	Pal * AtomC
0	∞	∞	∞	∞	∞	∞	∞
1	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞	∞	∞
3	∞	∞	1	∞	1	∞	1
4	∞	2	1	2	1	2	1
5	2	2	1	2	1	2	1
6	2	2	1	2	1	2	1

3.8 Mots sur A,B qui contiennent autant de A que de B

n	Vide	AtomA	AtomB	S	S1	T	U	aTbS	TbS	bS
règle	$\mathcal{E}$	A	B	$\mathcal{E}$ ∪ S1	aTbS ∪ bUaS	$\mathcal{E}$ ∪ aTbT	$\mathcal{E}$ ∪ bUaU	A * TbS	T * bS	B * S
0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
1	0	1	1	∞	∞	∞	∞	∞	∞	∞
2	0	1	1	0	∞	0	0	∞	∞	∞
3	0	1	1	0	∞	0	0	∞	∞	1
4	0	1	1	0	∞	0	0	∞	1	1
5	0	1	1	0	∞	0	0	2	1	1
6	0	1	1	0	2	0	0	2	1	1
7	0	1	1	0	2	0	0	2	1	1

n	bUaS	UaS	aS	aTbT	TbT	bT	bUaU	UaU	aU
règle	B * UaS	U * aS	A * S	A * TbT	T * bT	B * T	B * UaU	U * aU	A * U
0	∞	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	∞	∞	1	∞	∞	1	∞	∞	1
4	∞	1	1	∞	1	1	∞	1	1
5	2	1	1	2	1	1	2	1	1
6	2	1	1	2	1	1	2	1	1
7	2	1	1	2	1	1	2	1	1

4 Fonction qui vérifie qu'une grammaire est correcte

Pour vérifier qu'une grammaire est correcte, on parcourt toutes les règles de la grammaire et pour chaque règle, s'il s'agit d'une règle de type ConstructorRule, on vérifie si les deux règles qu'elle possède en attributs sont définies dans la grammaire.

ADD STYLE

```
def checkDefinedRules (grammar):
    for ruleId in grammar:
    rule = grammar[ruleId]
    if isinstance(rule, R.ConstructorRule):
    fst, snd = rule.parameters
    if not (fst in grammar):
    raise IncorrectGrammar(fst+"_rule_not_defined_in_grammar")
    if not (snd in grammar):
    raise IncorrectGrammar(snd+"_rule_not_defined_in_grammar")
```

5 Structure du programme

6 Tests de cohérence génériques

Les propriétés suivantes doivent etres vérifiées par les grammaires :

- Pour toute règle r d'une grammaire, pour tout entier n positif ou nul, r.count(n) == len(r.list(n))
- ([rule.rank(i) for i in rule.list(n)] == list(range(rule.count(n))))
- ([rule.rank(i) for i in rule.list(n)] == list(range(rule.count(n))))
- Pour chaque règle d'une grammaire la valeur de la valuation calculée, correspond au plus petit mot produit par la règle
- Si r est une EpsilonRule alors count(n) avec n différent 0 retourne 0 sinon 1
- Si r est une SingletonRule alors count(n) avec n différent de 1 retourne 0 sinon 1
- Pour toutes les fonctions, pour toutes les règles n négatif provoque une exception
- Pour la fonction count on compare avec les suites connues (ajouter ref oeis)
- Lors du calcul de la valuation on vérifie qu'aucune valuation n'a la valeur ∞

7 Ajout count, random, list, unrank

on implémente les algos de l'énoncé en ajoutant vérification cas < 0

8 Ajout rank

Montrer le code
Expliquer la modification des paramètres avec valeur = None possible
Expliquer comment construire les fonctions avec 2 exemples
-> DyckGram
-> AB

9 Caching

Comparaison temps exec avec caching et sans
Comparaison temps en fonction des fonctions mémoisées
Comparaison mémoisation python2 vs python3 ?

10 Ajout des Grammaires Condensées

Expliquer traduction
Création de nom de règle
-> Grammaires condensées dans GrammarsC (pas totalement pour lisibilité)
On execute les test sur les GrammarsC
+ Eventuellement comparaison des résultats GrammairesCondensées vs GrammairesNonCondensées.

11 Constructeur Bound

Explication
- list concat
- count sum
- unrank et rank
Créer tests ?

12 Sequence

Montrer traduction
Montrer utilisation pour réécriture de DyckGram