Algo - Séance 3 de TD

Exercice 5 (de la feuille de TD2)

1) tab = 8 7 12 5 15 4 | 3 9

Push(tab, 6)

$j=2$ 7 8 ______

$j=3$ inchangé

$j=4$ 7 8 5 12 ______

$j=5$ inchangé

$j=6$ 7 8 5 12 4 15 ______

2) boucle effectuée $k-1$ fois, et ses instructions ~~sont~~ élémentaires. Donc Push termine.

3) Invariant : $\Pi(j)$ = " - $tab_j[j ; n-1]$ inchangé "
($j \in [1, k]$) - $tab_j[0, j-1]$ contient son élément max en position $[j-1]$

Base : pour $j=1$, $tab_1 = tab$ donc $\begin{cases} [1 \dots n-1] \text{ inchangé} \\ [0] = \max [0 \dots 0] \end{cases}$

Induction : Soit $j \in [1, k-1]$, suppose $\Pi(j)$,

- alors tab_j n'a pas modifié $[j \dots n-1]$ et la boucle ~~su~~ivante ne va modifier que $tab_{j+1}[j]$ et $tab_{j+1}[j-1]$
  Donc $[j+1 \dots n-1]$ sera inchangée
- On sait que max $tab_j[0, \dots j-1]$ est en position $j-1$
  - si $tab_j[j-1] \leq tab_j[j]$, alors on aura

$tab_{j+1}[j] = tab_j[j] \geqslant tab_j[j-1] \geqslant tab_j[i] \quad \forall i < j-1$

$\phantom{tab_{j+1}[j] = tab_j[j] \geqslant tab_j[j-1] \geqslant} \overset{\shortparallel}{tab_{j+1}[i]}$

- sinon (on entre dans le if)

  $tab_j[j] < tab_j[j-1]$ donc $tab_j[j-1]$ est le max de $tab_{j+1}[0, .., j]$.

  Le programme inverse $tab_j[j]$ et $tab_j[j-1]$

  si bien que $\begin{cases} tab_{j+1}[j] > tab_{j+1}(j-1) \\ tab_{j+1}[j] = tab_j[j-1] \geqslant tab_j[i] \; \forall i < j-1 \end{cases}$

Donc par récurrence, la propriété est vraie $\forall j \in [1, k]$.

4) On sort de la boucle quand $j = k$, et $\Pi(k)$ est donc vérifiée : $\begin{cases} tab[k .. n-1] \text{ inchangé} \\ tab[k-1] \text{ contient le max de } tab[0..k-1] \end{cases}$

D'où la validité de Push.

Exo 6 (de la feuille de TD 2)

1) Trie le tableau par ordre croissant : à chaque tour de boucle, on met l'élément le plus grand à la fin et on réduit le tableau sur lequel on travaille.

tab = [18, 17, 4, 12, 1, 2], n=len(tab)=6
BubbleSort(tab)

i=1, on a fait push(tab, n), ça a mis 18 à la fin
i=2, push(tab, n-1), donc ça a mis 17 en position n-2
i=3, ...
i=6, on a le tableau trié et on sort de la boucle

2) On a n tours de boucle, et dans chaque boucle, on a des instructions élémentaires et Push. On a vu dans l'exo 5 que Push termine, donc BubbleSort termine aussi.

3) Invariant : $\Pi^*(i) = \begin{cases} tab^*_i[n-i \,..\, n-1] \text{ contient les } i \text{ plus grands éléments de tab, triés croissants} \\ tab^*_i[0 \,..\, n-i-1] \text{ contient les autres éléments de tab} \end{cases}$

Base : $\Pi^*(1)$. après un tour de boucle, l'élément max est en position n-1 grâce à push.
Le reste des éléments est au début.

Alternative : $\Pi^*(0)$, $\begin{cases} tab[n \,..\, n-1] = \emptyset \text{ donc vrai} \\ tab[0,..\, n-0-1] = tab \text{ donc vrai} \end{cases}$

Induction : Soit $i \in [\![0, n-1]\!]$, supposons $\Pi^*(i)$

On entre dans la boucle et on fait
$Push(tab^*_i, n-i-1)$

D'après Push, $tab^*_{i+1}[n-i-1, ..., n-1] = tab^*_i[n-i-1 .. n-1]$

" la fin du tableau reste inchangée "

Au début du tableau, Push met le max en position
$n-i-1$.

Par hypothèse de récurrence, tout ce qu'il y a dans le début du tableau est plus petit que ce qu'il y a dans la fin. Donc $tab^*_{i+1}[n-i-1] \leq tab^*_{i+1}[n-i, ..., n-1]$

(par hyp de rec)

$\geq tab^*_{i+1}[0, ..., n-i-2]$

(par propriété de Push)

Les éléments n-i à n-1 étaient déjà triés, et on ajoute le maximum des autres éléments en position n-i-1. Donc le tableau de n-i-1 à n-1 est trié.
De plus, Push conserve les mêmes éléments, donc tous les éléments restants sont entre 0 et n-i-2.

D'où $\Pi^*(i+1)$

Par principe de récurrence $\Pi^*(i)$ est vraie $\forall i \in [\![0, n]\!]$

4) On sort de la boucle quand i=n, et la propriété $\Pi^*(n)$ est vérifiée :

- tab*[n-i ... n-1] = tab* entier   est trié et contient tous les éléments de tab initial
- tab*[0 ... n-n-1] est vide

Donc BubbleSort a bien trié les éléments du tableau.

Question bonus (vous avez pas encore appris ça) : Combien de tours de boucle y a-t-il au total ?

Complexité : "nombre de tours de boucle"

Bubble

```
i=0
while i<n
    j=1
    while j<n-i
        élémentaire
```

Push

élémentaire $\big\} O(1)$ $\Big\} O(n-i)$ $\Bigg\}$ $\displaystyle\sum_{i=0}^{n} n-i = n^2 - \frac{n(n-1)}{2}$

On sait qu'on a environ $\frac{n^2}{2}$ tours de boucle. $\underset{n\to+\infty}{\approx} n^2 - \frac{n^2}{2} \approx \frac{n^2}{2}$

On parle donc d'un tri en $\Theta(n^2)$.

TD 3 : Algorithmes récursifs

Rappel de cours

Algo itératif

Fait une **boucle** un certain nb de fois défini par des paramètres

ex
```
def fac(n):
    r = 1
    for i from 1 to n:
        r = r x i
    return r
```

Terminaison : ∀ param, le nb de boucles est fini

Validité : invariant de boucle dont dépend la valeur retournée

Intérêt : intuitif
programmation

Attention : la boucle termine pas

Algo récursif

S'appelle lui-même en changeant ses paramètres (avec un cas de base)

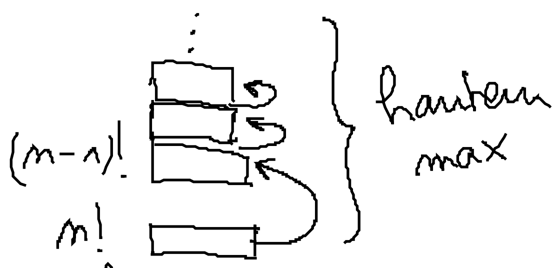
```
def fac(n):
    if n == 0 : return 1
    (else) : return n x fac(n-1)
```

Par récurrence, le nb d'appels est fini ∀ param

Par récurrence, la fonction est valide ∀ param

concision
mathématique

Soit (u_n) une suite telle que
- $u_0 = 1$, $\forall n \in \mathbb{N}$, $u_{n+1} = u_n \times (n+1)$



Stack overflow

Exercice 4 : fonction puissance récursive

1) 1/ $u_{10}(2) = (u_5(2))^2 = (u_2^2(2) \times 2)^2$

$$= ((u_1(2)^2)^2 \times 2)^2$$

$$= (((\underbrace{u_0(2)}_{1}{}^2 \times 2)^2)^2 \times 2)^2$$

$$= ((2^2)^2 \times 2)^2 = (4^2 \times 2)^2 = (16 \times 2)^4$$

$$= 32 \times 32 = \underline{1024}$$

1
2
4
16
32
64
128
256
512
$2^{10} = 1024$
2048

2/ Base : $u_0(x) = 1 = x^0 \quad \forall x \in \mathbb{R}$

Induction :

Supposons que $\forall m \leq n$, $u_m(x) = x^m$

Mq $u_{n+1}(x) = x^{n+1}$.

Si $n+1$ est pair, $u_{n+1}(x) = \left(u_{\frac{n+1}{2}}(x)\right)^2$

$$= \left(x^{\frac{n+1}{2}}\right)^2 = x^{n+1}$$

Sinon, $u_{n+1}(x) = \left(u_{\frac{n}{2}}\right)^2 \times x = \left(x^{n/2}\right)^2 \times x = x^n \cdot x = x^{n+1}$

2) Puissance(2, 10)

"n = 10"
"n = 5"
"n = 2"
"n = 1"
"n = 0"

empilement

⚠ Stack overflow

$$10 \to 5 \to 2 \to 1 \to 0$$

Puissance(2, 0) = 1
Puissance(2, 1) = 2
Puissance(2, 2) = 4
Puissance(2, 5) = 32
Puissance(2, 10) = 1024

3) Montrer par récurrence que ça termine et que c'est valide.

<u>Base</u> : la fonction n'a que des opérations élémentaires et renvoie $1 = x^0$

<u>Induction</u> : Suppose que Puissance(x, m) termine et renvoie x^m $\forall m \leq n$

Prenons Puissance(x, n+1)

On a des opérations élémentaires, et Puissance(x, $\frac{n+1}{2}$) qui (par hypothèse) termine. Donc ça termine en renvoyant u_{n+1} comme défini ci-dessus.

Donc Puissance(x, n) termine et renvoie x^n $\forall n \in \mathbb{N}$

Exo 1 : Magenta, Jaune
Exo 2 : Vert, Rouge
Exo 3 : Bleu

} jusqu'à 12h10

Exo 1 : calcul récursif du nombre d'occurrences d'un élément dans un tableau

1) nb_occ([3,6,7,6,2,6,3], 6) "combien de fois y a-t-il 6 dans ce tableau ?"

Déroulement :

tab = tab[0:k] = [3 6 7 6 2 6 3] x = 6
[3 6 7 6 2 6]
[3 6 7 6 2]
...
[ ] x = 6
[ ] occ = 0
[3] occ = 0
[3 6] occ = 1
[3 6 7] occ = 1
[3 6 7 6] occ = 2
...
[3 6 7 6 2 6 3] occ = 3

2) Base : $k=0$, nb_occ pour $k=0$ n'a que des opérations élémentaires
On renvoie 0 car dans le tableau vide, il n'y a pas x

Induction : Si ça termine pour k, Mq ça se termine pour $k+1$

si tab[k-1] == x, on appelle nb_occ pour k, qui se termine par hypothèse

si tab[k-1] $\neq$ x, même chose. Donc ça termine

Validité : si nb_occ pour k renvoie le nb d'occ entre tab[0, ..., k-1], alors,

soit tab[k] vaut x, donc on ajoute 1 occurrence au nb précédent (correct par hypothèse)

soit tab[k] $\neq x$ donc on renvoie le nb d'occ précédent (valide par hypothèse).

Conclusion : OK

Nous corrigerons rapidement les exos 2 et 3 en début de séance prochaine.