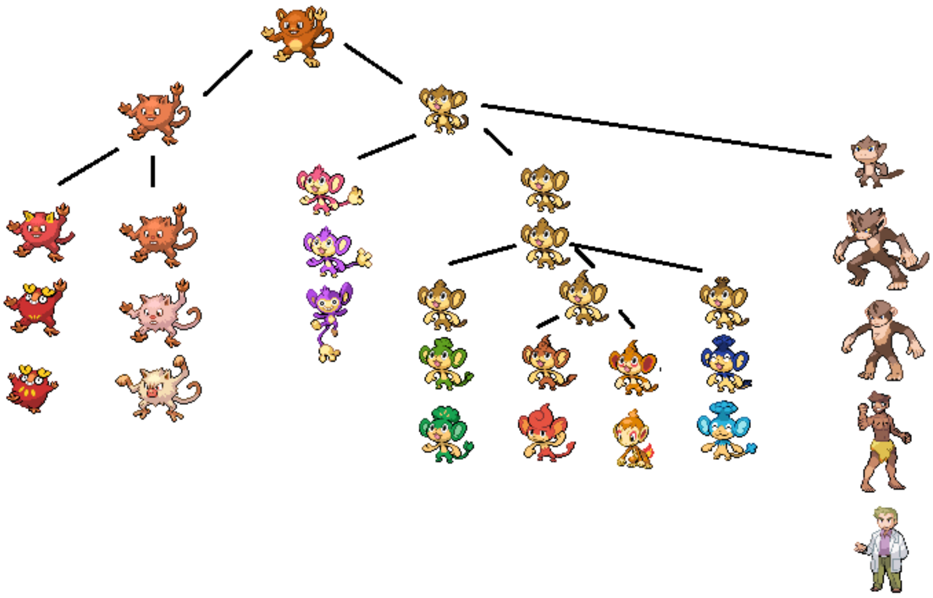
TD n° 2 - Structures de données

Les exercices qui suivent traitent du problème du ***Première Ancêtre Commune*** (PAC) dans les arbres enracinés. Etant donné un arbre enraciné T, il s'agit de répondre efficacement aux requêtes suivantes : soient u, v deux nœuds quelconques de T, calculer PAC(u, v) l'ancêtre commune à u et v le plus profond dans l'arbre. La figure ci-dessous illustre le résultat de requêtes de ce type.



PAC(, )=    **PAC(**, )= 

Dans tous les exercices, l'arbre enraciné sera donné en entrée à la fois par :

- `mère`$[x]$ qui associe à tout nœud x son père (en temps $\mathcal{O}(1)$), avec la convention `mère`$[r]$=r pour la racine r.
- `fille`$[x]$ qui associe à tout nœud x la liste de ses filles (accès au début de liste en temps $\mathcal{O}(1)$), avec la convention `fille`$[f]$=$\emptyset$ pour toute feuille f.

Remarquer que les données `mère` et `fille` sont redondantes : connaître l'une de ces deux informations pour tout nœud x, permet de recalculer l'autre en temps $\mathcal{O}(n)$ où n est le nombre total de nœuds.

Notation : une *solution en temps* $\langle f(n), g(n)\rangle$ pour traiter un type de requête sur une structure de données (p.ex. PAC sur les arbres enracinés) est un algorithme qui, pour toute entrée à n éléments, effectue un précalcul en temps $f(n)$ (en construisant éventuellement des structures de données annexes) qui permet ensuite de répondre à toute requête en temps $g(n)$. Noter que, dans tous les exercices, on considèrera la complexité pire cas.

Exercice 1. *Premiers algorithmes pour résoudre PAC*

1. Donner un algo simple qui résout PAC en temps $\langle f(n), \mathcal{O}(1)\rangle$ où $f(n)$ est polynomial en n.

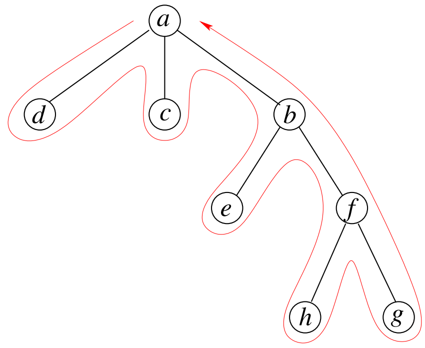
2. Donner un algo simple qui résout PAC en temps $\langle \mathcal{O}(1), g(n)\rangle$ où $g(n)$ est polynomial en n.

3. Montrer que dans le cas particulier des arbres T réduits à une seule branche, PAC admet une solution en temps $\langle \mathcal{O}(n), \mathcal{O}(1)\rangle$.

Exercice 2. *De PAC à IMIN*

Cet exercice propose une réduction de PAC à IMIN, le problème des minimum de sous-tableaux. Etant donné un tableau $A[1..n]$ de n entiers, il s'agit de répondre efficacement aux requêtes suivantes : soient $1 \leq i \leq j \leq n$, donner l'indice $\mathrm{IMIN}(i, j)$ du (ou d'un) minimum du sous-tableau $A[i..j]$.

Soit T un arbre enraciné, un *Tour Eulérien* de T est une promenade fermée qui part de la racine et passe exactement deux fois par chaque arête de l'arbre. On peut stocker cette promenade ainsi que les changements de profondeur dans un tableau $E[1..(2n-1)]$ où $E[i]$ contient le i-ème nœud rencontré et sa profondeur. Dans ce tableau, $E[1]$ et $E[2n-1]$ contiennent la racine avec la profondeur 0, et chaque nœud apparaît plusieurs fois en fonction de son degré. La figure ci-dessous donne une illustration d'un Tour Eulérien et son stockage dans un tableau (avec les profondeurs des nœuds).



Un Tour Eulerien avec les profondeurs

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Noeud	*a*	*d*	*a*	*c*	*a*	*b*	*e*	*b*	*f*	*h*	*f*	*g*	*f*	*b*	*a*
Profondeur	*0*	*1*	*0*	*1*	*0*	*1*	*2*	*1*	*2*	*3*	*2*	*3*	*2*	*1*	*0*

1. Ecrire un algorithme qui calcule un Tour Eulérien de T et le stocke dans un tableau avec les profondeurs, en temps $\mathcal{O}(n)$.

2. Une fois le Tour Eulérien stocké dans un tableau $E[1..(2n-1)]$ avec les profondeurs des nœuds, associer à chaque nœud x de l'arbre l'indice $R[x]$ de sa première occurence dans $E[1..(2n-1)]$. Montrer alors que toute requête PAC sur T peut se ramener à une requête IMIN sur E (via les indices stockés dans R).

Exercice 3. *IMIN et IMIN$\pm$1 pour conclure sur PAC*

Dans le précédent exercice, le tableau des profondeurs qui suit un Tour Eulérien a la propriété que deux profondeurs consécutives varient toujours de $+1$ ou -1. Le problème IMIN restreint à ce type de tableau en entrée sera noté IMIN$\pm$1.

1. Montrer qu'il existe une solution en temps $\langle \mathcal{O}(n \log n), \mathcal{O}(1) \rangle$ à IMIN.

On va chercher une solution en temps $\langle \mathcal{O}(n), \mathcal{O}(1) \rangle$ pour IMIN $\pm$ 1. Soit $A[1..n]$ est une instance où deux valeurs consécutives diffèrent toujours de exactement $+1$ ou -1, on découpe A en blocs de largeur $\log_2(n)/2$ (arrondir comme il faut) et on construit un nouveau tableau $\tilde{A}[1..2n/\log_2(n)]$ où $\tilde{A}[i]$ est le minimum du i-ème bloc de A. Un bloc $A[i..j]$ est dit *normalisé* si $A[i] = 0$, sinon on peut le *normaliser* en soustrayant $A[i]$ à toutes les valeurs du bloc.

2. Est-il toujours vrai que les réponses aux requêtes IMIN sont les mêmes pour un bloc et sa version normalisée ?

3. Donner un majorant sous-linéaire sur le nombre de blocs normalisés de largeur $\log_2(n)/2$. Quelle est la complexité pour précalculer et stocker toutes les réponses aux requêtes IMIN pour tous les blocs possibles ?

4. En appliquant la Question 1 au tableau $\tilde{A}$ et la Question 3 aux blocs de A, montrer comment obtenir une solution en temps $\langle \mathcal{O}(n), \mathcal{O}(1) \rangle$ pour IMIN $\pm$ 1.

5. En déduire qu'il existe une solution en temps $\langle \mathcal{O}(n), \mathcal{O}(1) \rangle$ pour PAC.