TP 4

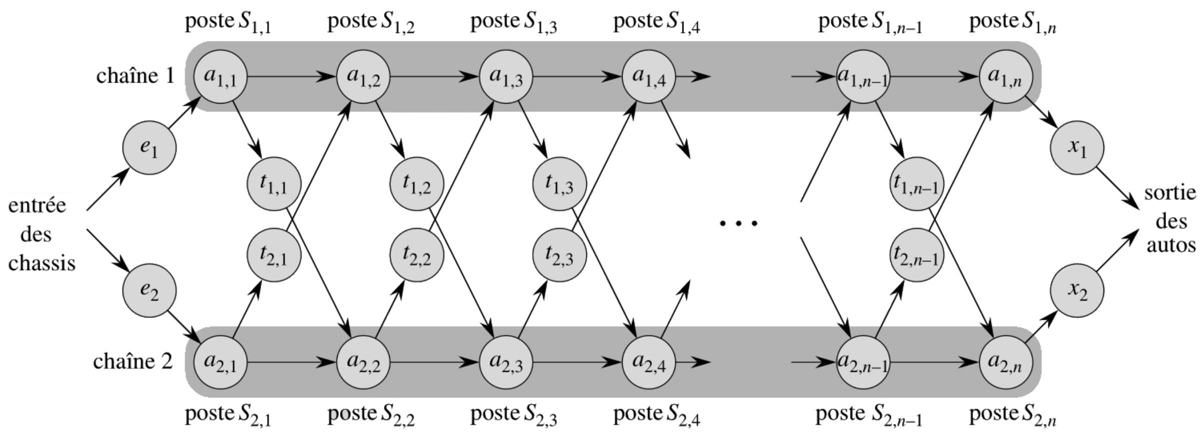
PROGRAMMATION DYNAMIQUE

Exercice 1

Un constructeur automobile possède un atelier avec deux chaînes de montage comportant chacune n postes. Chaque véhicule doit passer par les n postes dans l'ordre. Le constructeur cherche à déterminer quels sont les postes à sélectionner sur la chaîne 1 et sur la chaîne 2 pour minimiser le délai de transit d'une voiture à travers l'atelier. On note, pour tout $i \in [\![1, 2]\!]$ et tout $j \in [\![1, n]\!]$,

- $\mathrm{S}_{i,j}$ le j-ème poste de la chaîne i ;
- e_i le temps d'entrée d'un véhicule sur la chaîne i ;
- $a_{i,j}$ le temps de montage pour le poste j sur la chaîne i ;
- $t_{i,j}$ le temps de transfert d'un véhicule de la chaîne i vers l'autre chaîne après le poste $\mathrm{S}_{i,j}$;
- x_i le temps de sortie d'un véhicule de la chaîne i.

On peut représenter ces valeurs à l'aide d'un graphe



Chaque solution de ce problème d'optimisation est définie par le sous-ensemble de postes de la chaîne 1 utilisés (les postes restant sont choisis dans la chaîne 2). Il y a donc 2^n solutions possibles *i.e.* le nombre de sous-ensembles d'un ensemble à n éléments. Par conséquent, l'approche naïve consistant à considérer tous les chemins possibles est inefficace.
La programmation dynamique permet de résoudre ce problème efficacement.

▷ On identifie des sous-problèmes dont les solutions optimales vont nous permettre de reconstituer une solution optimale du problème initial.

  Les sous-problèmes à considérer ici consistent à calculer un itinéraire optimal jusqu'à chacun des postes $\mathrm{S}_{i,j}$.

  Par exemple, considérons un itinéraire optimal jusqu'au poste $\mathrm{S}_{1,j}$, celui-ci est

  - un chemin direct de l'entrée au poste $\mathrm{S}_{1,1}$ (si $j = 1$),
  - soit un chemin optimal jusqu'au poste $\mathrm{S}_{1,j-1}$ suivi du poste $\mathrm{S}_{1,j}$ (si $j > 1$),
  - soit un chemin optimal jusqu'au poste $\mathrm{S}_{2,j-1}$ suivi d'un changement de chaîne et du poste $\mathrm{S}_{1,j}$ (si $j > 1$).

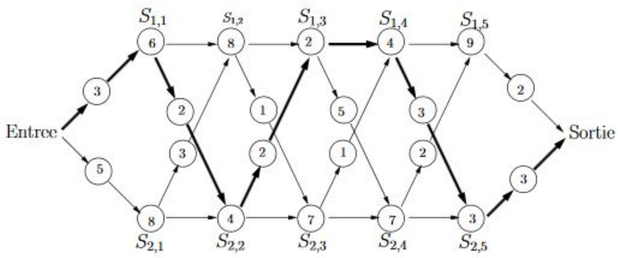
▷ On définit la valeur optimale de manière récursive à partir des valeurs des solutions optimales des sous-problèmes. Notons $f_i(j)$ le délai optimal jusqu'à $S_{i,j}$ et f^* le délai optimal total. Pour traverser l'atelier, il faut atteindre ou bien $S_{1,n}$ ou bien $S_{2,n}$ et sortir de l'atelier. Par conséquent :

$$f^* = \min(f_1(n) + x_1, f_2(n) + x_2) \qquad \text{avec } \begin{cases} f_1(1) = e_1 + a_{1,1} \\ f_2(1) = e_2 + a_{2,1} \end{cases}$$

et, de manière récursive, pour tout $j \in [\![2, n]\!]$:

$$f_1(j) = \min\big(f_1(j-1) + a_{1,j}, f_2(j-1) + t_{2,j-1} + a_{1,j}\big),$$
$$f_2(j) = \min\big(f_2(j-1) + a_{2,j}, f_1(j-1) + t_{1,j-1} + a_{2,j}\big).$$

Écrire une fonction itérative `chaine(a,t,e,x)` d'arguments quatre listes correspondant aux données de l'énoncé et qui renvoie le délai optimal f^*.
Par exemple, on a :



j	1	2	3	4	5
$f_1(j)$	9	17	19	23	32
$f_2(j)$	13	15	22	29	29

$f^* = 32.$

Exercice 2

On dispose d'un tableau indiquant les tarifs des vols reliant n villes d'Europe. Celui-ci est codé à l'aide d'une liste D de n^2 cases où n est le nombre de ces villes (numérotées de 0 à $n-1$). On convient que la case $D[n \times i + j]$ contient le prix du vol direct au départ de la ville numéro i et à l'arrivée dans la ville numéro j.

1. On note, $W^0 = D$ et, pour tout $k \in [\![0, n-1]\!]$, W^{k+1} la liste donnant le coût minimal pour connecter les principales villes d'Europe en autorisant des correspondances dans les villes numérotées de 0 à k.
   Écrire la relation de récurrence reliant les coefficients de W^{k+1} à ceux de W^k.
2. En exploitant une stratégie de programmation dynamique sous forme itérative, écrire une fonction efficace `FW(D)` renvoyant le liste des coûts optimaux pour des vols entre ces n villes.
3. Écrire une fonction `CircuitOpt(D)` renvoyant la liste des trajets optimaux pour des vols entre ces n villes (chaque case est donc une liste contenant les numéros des villes traversées).
4. Traduire cet exercice avec le vocabulaire des graphes.

Exercice 3

En partant du sommet du triangle ci-dessous et en se déplaçant vers les nombres adjacents de la ligne inférieure, le total maximum que l'on peut obtenir pour relier le sommet à la base est égal à 23 :

$$\begin{array}{ccccccc} & & & \mathbf{3} & & & \\ & & \mathbf{7} & & 4 & & \\ & 2 & & \mathbf{4} & & 6 & \\ 8 & & 5 & & \mathbf{9} & & 3 \end{array} \qquad 3 + 7 + 4 + 9 = 23.$$

On modélise ce triangle à l'aide d'une liste de listes.

1. Pour $(i, j) \in [\![0, n-1]\!]^2$ avec $i \leqslant j$, on note $S_{i,j}$ le plus grand total en partant du coefficient à la ligne i colonne j.
   Que vaut $S_{n-1,j}$? Donner une relation pour $0 \leqslant j \leqslant i \leqslant n-2$, une formule reliant $S_{i,j}$, $S_{i+1,j}$ et $S_{i+1,j+1}$.
2. Programmer une fonction calculant le total maximal pour un chemin reliant le sommet du triangle à sa base.
3. Modifier cette fonction pour renvoyer un chemin qui réalise cette somme.
   On codera le chemin par une liste des colonnes visitées depuis la ligne 1 jusqu'à la ligne n.