Licence Informatique — Algorithmique — Juin 2007
Durée 3 heures — documents et calculatrices interdits

Prenez le temps de faire les calculs au brouillon avant de les recopier sur votre copie d'examen. Efforcez–vous de traiter les questions dans l'ordre. Indiquez bien le numéro de chaque question avant la réponse. Les copies mal présentées seront pénalisées.

1 Programmation linéaire et AMPL (12 points)

Un organisme académique envisage d'affecter certains membres de son personnel à deux services A et B. Un membre du personnel a une charge hebdomadaire différente, suivant qu'il est affecté au service A ou au service B, dans les trois domaines : enseignement, recherche et administration. L'organisme a estimé les besoins à satisfaire approximativement dans ces trois domaines. Le tableau suivant récapitule la charge de travail hebdomadaire d'un membre du personnel dans chacun des services ainsi que les besoins hebdomadaires en personnes–jours de l'organisme :

	Enseignement (en jours)	Recherche (en jours)	Administration (en jours)
Service A	1	3	1
Service B	3	2	0
Besoins (en personnes–jours)	8	13	5

L'organisme souhaite déterminer le nombre x_1 de membres du personnel à affecter au service A et le nombre x_2 de membres du personnel à affecter au service B qui répondent le mieux aux besoins. Pour être précis, l'organisme souhaite minimiser le maximum x_3 des écarts, en valeur absolue, pour les trois domaines, entre le nombre de personnes–jours estimé et le nombre de personnes–jours réalisé.

1.1 Modélisation mathématique

Question 1 [2 pts]. Modéliser le problème ci–dessus sous la forme d'un programme linéaire en les trois variables réelles fournies dans l'énoncé. Ne pas hésiter à présenter le programme sous une forme différente de celle habituellement adoptée en cours, pour mieux faire ressortir sa structure.

SOLUTION. Programme linéaire (sous une forme inhabituelle) :

$$\begin{cases} x_3 = z[\min] \\ -x_3 \leq x_1 + 3\,x_2 - 8 \leq x_3 \\ -x_3 \leq 3\,x_1 + 2\,x_2 - 13 \leq x_3 \\ -x_3 \leq x_1 - 5 \leq x_3 \\ x_1,\, x_2,\, x_3 \geq 0. \end{cases}$$

Question 2 [2 pts]. Mettre le programme linéaire sous forme standard. Expliquer en une phrase pourquoi il pose un problème de démarrage.

SOLUTION. Les variables d'écart sont les variables $x_4, \ldots, x_9$. Il y a un problème de démarrage parce que l'origine (pour le programme précédent) n'est pas une solution réalisable ou encore (c'est équivalent), parce que la base formée des variables d'écart n'est pas réalisable.

$$\begin{cases} -x_3 = z[\max] \\ x_1 + 3\,x_2 + x_3 - x_4 = 8 \\ x_1 + 3\,x_2 - x_3 + x_5 = 8 \\ 3\,x_1 + 2\,x_2 + x_3 - x_6 = 13 \\ 3\,x1 + 2\,x_2 - x_3 + x_7 = 13 \\ x1 + x3 - x_8 = 5 \\ x_1 + x_3 + x_9 = 5 \\ x_1, \ldots, x_9 \geq 0. \end{cases}$$

Question 3 [1 pt]. Donner le programme artificiel associé au programme de la question précédente (laisser l'objectif artificiel sous une forme simple). Ne pas résoudre ce programme.

SOLUTION. On introduit trois variables artificielles $x_{10},\ x_{11},\ x_{12}$ dans les contraintes 1, 3 et 5.

$$\begin{cases} x_{10} + x_{11} + x_{12} = z[\min] \\ x_1 + 3\,x_2 + x_3 - x_4 + x_{10} = 8 \\ x_1 + 3\,x_2 - x_3 + x_5 = 8 \\ 3\,x_1 + 2\,x_2 + x_3 - x_6 + x_{11} = 13 \\ 3\,x1 + 2\,x_2 - x_3 + x_7 = 13 \\ x1 + x3 - x_8 + x_{12} = 5 \\ x_1 + x_3 + x_9 = 5 \\ x_1, \ldots, x_{12} \geq 0. \end{cases}$$

1.2 Modélisation en AMPL

On souhaite réaliser un modèle AMPL qui abstraie le problème précédent. En particulier, on souhaite pouvoir changer le nombre de services et de domaines sans réécrire le modèle.

Question 4 [1 pt]. Déclarer le ou les ensembles nécessaires.

Question 5 [1 pt]. Déclarer le ou les paramètres.

Question 6 [1 pt]. Déclarer la ou les variables nécessaires.

Question 7 [1 pt]. Écrire l'objectif économique du modèle.

Question 8 [1 pt]. Écrire la ou les contraintes du modèle.

Question 9 [2 pts]. Donner la partie *data* correspondant à l'énoncé.

SOLUTION. Voici une solution complète.

```
set SERVICES;
set DOMAINES;

param charge {SERVICES, DOMAINES} >= 0;
param besoin {DOMAINES} >= 0;

var effectif {SERVICES} >= 0;
var ecart >= 0;

minimize ecart_va : ecart;

subject to ecart_estime_realise_inf {d in DOMAINES} :
    sum {s in SERVICES} effectif [s] * charge [s,d] - besoin [d] >= - ecart;

subject to ecart_estime_realise_sup {d in DOMAINES} :
    sum {s in SERVICES} effectif [s] * charge [s,d] - besoin [d] <= ecart;

data;

set SERVICES := A B;
set DOMAINES := enseignement recherche administration;

param charge :
    enseignement recherche administration :=
  A      1           3           1
  B      3           2           0;

param besoin :=
  enseignement      8
  recherche        13
  administration    5;
```

2 Comptage de chemins dans un graphe acyclique (6 points)

On se donne un graphe orienté sans circuit $G = (S, A)$ et un sommet distingué s à partir duquel tous les autres sommets de G sont accessibles. On veut mettre au point un algorithme qui, pour chaque sommet $x \in S$, donne le nombre $n(x)$ de chemins possibles de s vers x (on ne veut pas les chemins, seulement leur nombre).

On rappelle un schéma algorithmique généralement applicable à tous les graphes sans circuit : *avant de traiter un sommet x, on s'arrange pour traiter tous les prédecesseurs de x.*

Question 10 [1 pt]. Nommer un algorithme vu en cours qui applique ce schéma algorithmique pour résoudre un autre problème. Quel est le problème résolu par cet algorithme ?

SOLUTION. L'algorithme de Bellman. Il détermine un chemin de valeur minimale entre le sommet s et tous les autres sommets du graphe.

Question 11 [1 pt]. Que vaut $n(s)$?

SOLUTION. $n(s) = 1$.

Question 12 [1 pt]. Exprimer $n(x)$ en fonction des $n(y)$ où y balaie l'ensemble des prédecesseurs de x.

SOLUTION. $n(x)$ vaut la somme des $n(y)$ où y balaie l'ensemble des prédecesseurs de x.

Question 13 [1 pt]. Donner en pseudocode, à la façon du cours, un algorithme fondé sur les idées précédentes, qui calcule $n(x)$ pour tous les sommets x du graphe. L'algorithme ne doit pas dépasser douze lignes.

SOLUTION. Voici le pseudocode

```
procédure compter (G, s)
début
  Trier topologiquement le graphe
  n(s) := 1
  pour tout sommet x (après s) suivant l'ordre topologique faire
    n(x) := 0
    pour y balayant les prédecesseurs de x faire
      n(x) := n(x) + n(y)
    fait
  fait
fin
```

Question 14 [2 pts]. Déterminer la complexité en temps, dans le pire des cas, de l'algorithme de la question précédente. On peut supposer que le graphe est implanté avec des listes de prédecesseurs et des listes de successeurs. Si votre algorithme comporte plusieurs étapes successives, indiquer la complexité en temps, dans le pire des cas, de chacune des étapes.

SOLUTION. Le tri topologique a une complexité en $O(n + m)$ c'est–à–dire en $O(m)$ puisque le graphe est connexe. On a utilisé le fait que le graphe est donné par des listes de successeurs. Le coût total de la première instruction de la boucle extérieure (initialisations de $n(x)$) est en $O(n)$. Le coût total du corps de la boucle intérieure est en $O(m)$ puisque la boucle parcourt tous les prédecesseurs de tous les sommets et le graphe est donné par des listes de prédecesseurs. Au total, on obtient une complexité en temps, dans le pire des cas, en $O(m) + O(n) + O(m)$ qui se simplifie en $O(m)$ puisque le graphe est connexe.

3 Chemins de valeur minimale et tas binaires (6 points)

3.1 Chemins de valeur minimale

Question 15 [3 pts]. Appliquer l'algorithme de Dijkstra sur le graphe de figure 1 à partir du sommet A. Utiliser la feuille jointe à l'énoncé. À chaque itération,

1. colorier en rouge les sommets déjà traités, en vert les sommets en cours de traitement et en bleu les sommets pas encore rencontrés ;
2. indiquer dans le tas binaire représenté à droite du graphe un état possible de la file avec priorité ;
3. reporter sur le graphe et sur le tas binaire, la valeur *pi*.

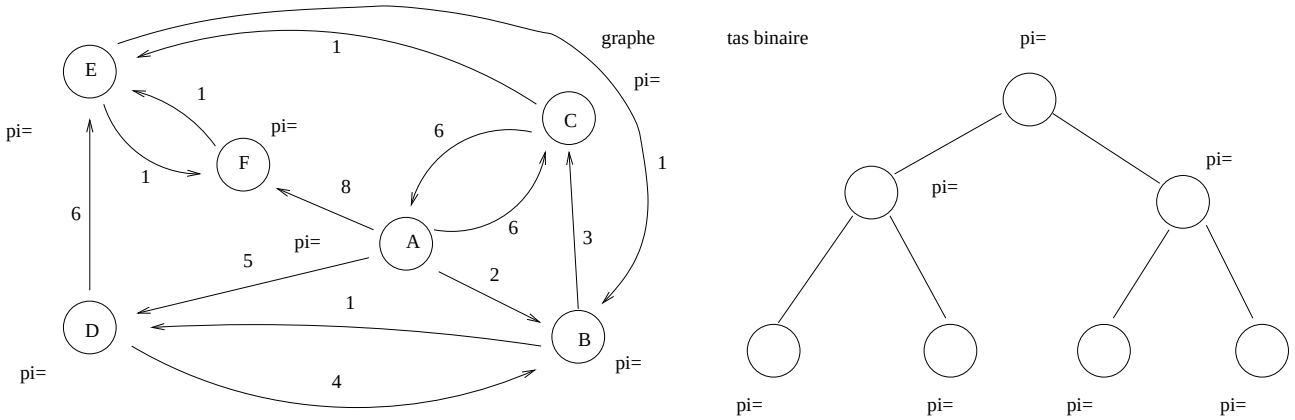


FIG. 1 – Graphe et tas binaire

3.2 Tas binaires

On considère un tas binaire, contenant n entiers, où chaque élément est inférieur ou égal à ses fils gauche et droit. On s'intéresse au calcul du k–ième plus petit élément du tas avec $k < n$.

Question 16 [1 pt]. Donner en une ou deux phrases un algorithme de complexité en temps, dans le pire des cas, en $O(k \log n)$ qui permette de déterminer le k–ième plus petit élément du tas. Cet algorithme peut éventuellement modifier le tas. Justifier la complexité.

> SOLUTION. Il suffit d'extraire k fois de suite le plus petit élément du tas. Comme chaque extraction a une complexité en temps, dans le pire des cas, en $O(\log n)$, on trouve une complexité totale en $O(k \log n)$.

Question 17 [1 pt]. Rappeler comment il est possible d'implanter ce tas au moyen d'un tableau T d'entiers. Préciser en particulier comment accéder aux fils gauche et droit de l'élément situé à l'indice i.

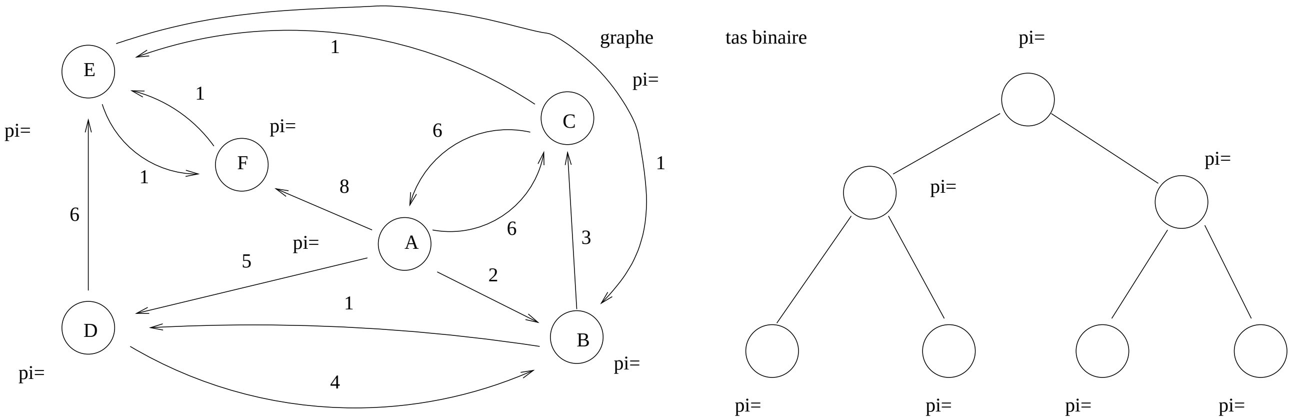
SOLUTION. On indice les éléments de T à partir de 1. Les fils gauche et droit de T_i sont T_{2i} et T_{2i+1}.

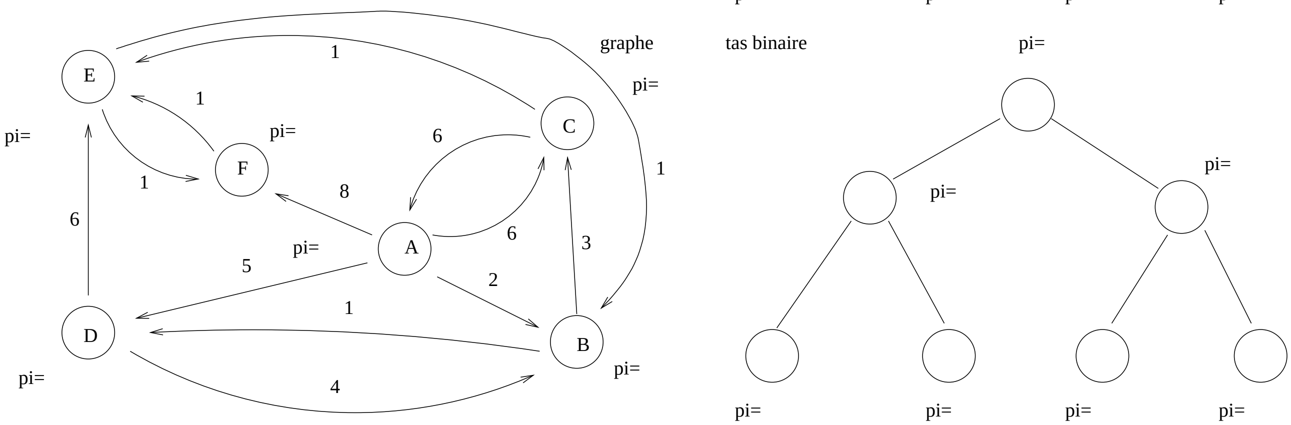
Question 18 [1 pt]. On suppose le tas implanté au moyen d'un tableau T d'entiers. Le k–ième plus petit élément du tas est situé dans T à un certain indice i. Donner une borne supérieure pour i. Ne pas hésiter à expliquer, au moyen d'un dessin, comment sont placés les k premiers éléments du tas, dans le pire des cas.

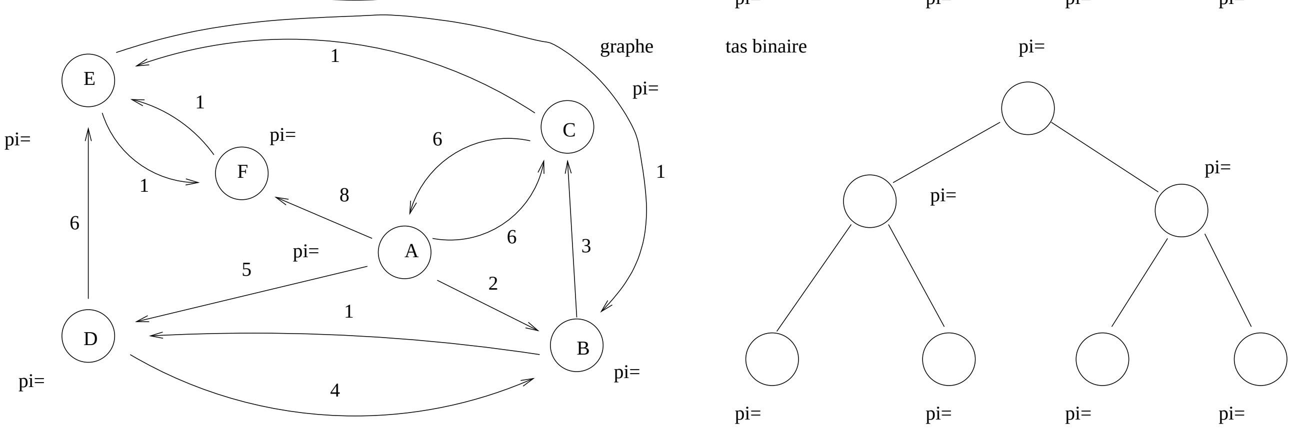
SOLUTION. Dans le pire des cas, le i–ème élément du tas est le fils droit de $(i-1)$–ème pour $2 \leq i \leq k$. Dans ce cas, le k–ème se trouve à l'indice $2^k - 1$.

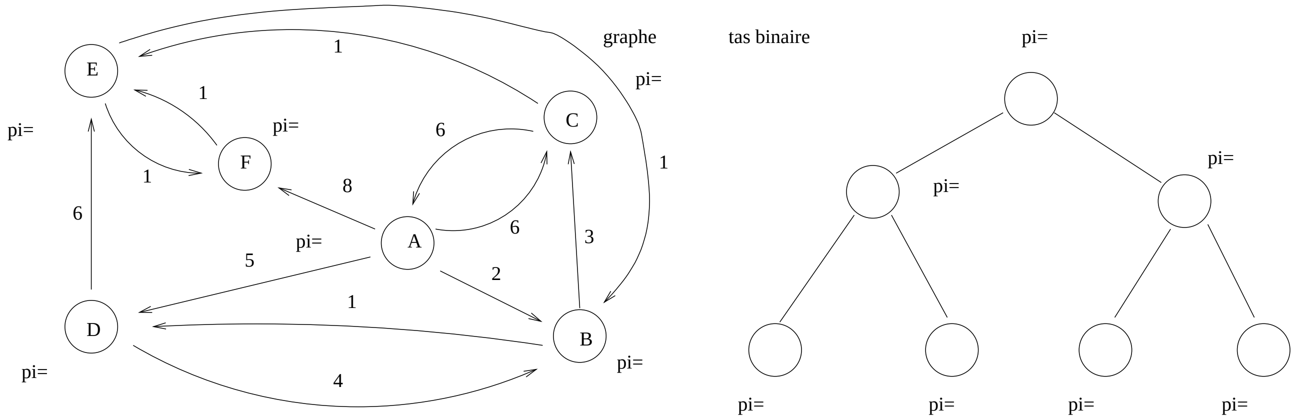
Annexe pour la question sur l'algorithme de Dijkstra

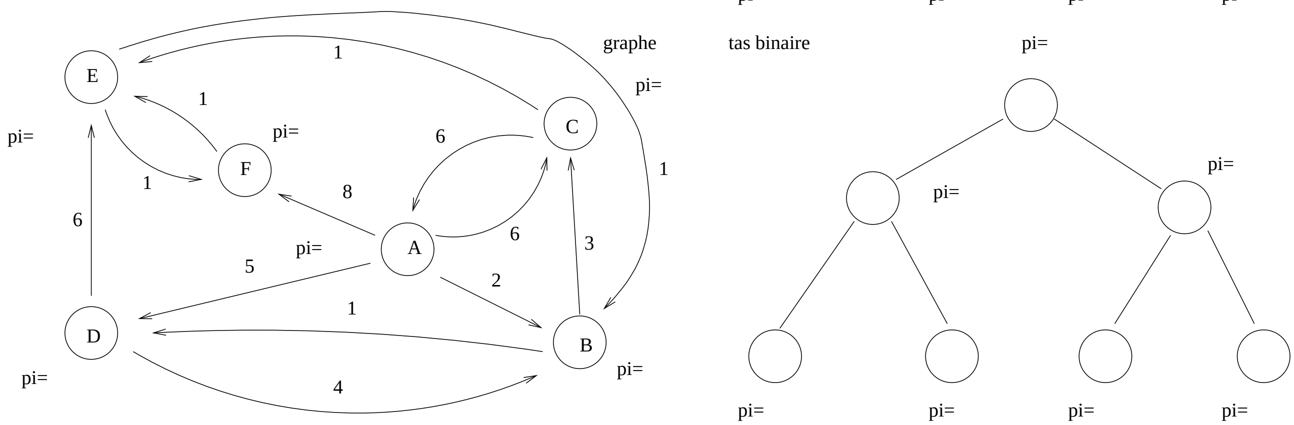
Numéro de place (seul indice de recherche en cas de perte) :

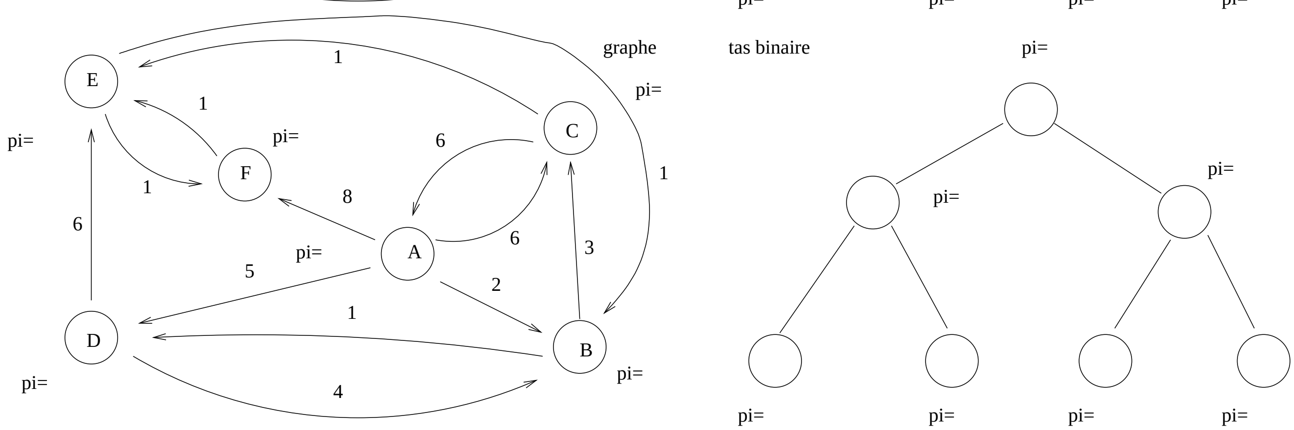


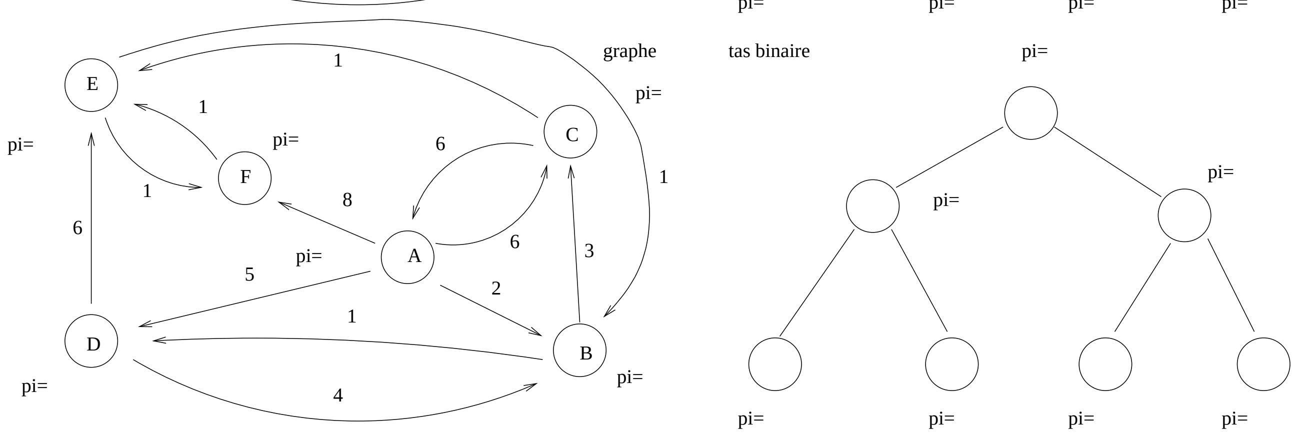





graphe
tas binaire
pi=
E F C A D B
1 1 1 6 8 6 6 1 3 5 2 1 4


graphe
tas binaire
pi=
E F C A D B
1 1 1 6 8 6 6 1 3 5 2 1 4


graphe
tas binaire
pi=
E F C A D B
1 1 1 6 8 6 6 1 3 5 2 1 4


graphe
tas binaire
pi=
E F C A D B
1 1 1 6 8 6 6 1 3 5 2 1 4