Corrigé : Examen de Théorie des Graphes

Vayssade Jehan-Antoine

03 avril 2026

Durée : 1 heure 30 Note maximale : 20 points

1 Graphes et propriétés de base (5.0-9.0 points)

Soit le graphe non orienté $G = (V, E)$ donné.

1. **Dégré de chaque sommet** (0,5 pt)

   $\deg(A) = 3$, $\deg(B) = 3$, $\deg(C) = 3$, $\deg(D) = 3$, $\deg(E) = 3$, $\deg(F) = 3$.

2. **Le graphe est-il connexe ?** (0,5 pt)

   Oui, le graphe est connexe (on peut rejoindre n'importe quel sommet depuis n'importe quel autre). Dit autrement il n'existe pas de sommet non connecter au graphe. D'autant plus que les graphes est non-orienter ce qui facilite l'analyse.

   Dans le cas d'un graphe oriente un algorithme de type Kosaraju aurait été intéressant pour compter les nombres de composante connexes et automatiser l'analyse. Dans le cas présent, d'un graphe non-oriente, DFS ou BFS permettent également de l'automatiser (+0.5 pt).

3. **Le graphe est-il biparti ?** (0,5 pt)

   Non et la raison est simple : un graphe est biparti si et seulement s'il ne contient aucun cycle de longueur impaire. Or ici, on trouve immédiatement un cycle de longueur 3 (impair) : $A - E - F - A$.

   Ou plus simplement et vue brièvement : un graphe bipartis a un nombre chromatique qui est inférieur ou égal à 2. Plus simplement on peu avoir 1 ou 2 couleurs colorier tous les nœuds, donc si on attribue une couleur a A et un autre a E, F dois nécessairement prendre un troisième couleur. Ce qui invalide la propriétés (+0.5 pt).

   De façon plus rigoureuse, on peut encadrer le nombre chromatique par $\frac{n(G)}{\alpha(G)} \leq \varphi(G) \leq \Delta(G) + 1$. Avec (n(G)=6) (le graphe possède six sommets), $\alpha(G) = 2$ (on ne peut pas trouver un ensemble indépendant de plus de deux sommets) et $\Delta(G) = 3$ (chaque sommet est de degré 3), on obtient l'encadrement $\frac{6}{2} \leq \varphi(G) \leq 3 + 1$, soit $3 \leq \varphi(G) \leq 4$. Ainsi, le nombre chromatique du graphe est compris entre 3 et 4, et comme on a montré précédemment qu'une coloration avec 3 couleurs est impossible, on en déduit finalement que $\varphi(G) = 4$. (+1.5 pt)

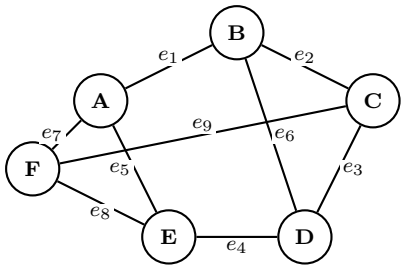
4. **Matrice d'adjacence de** G (1 pt)
(ordre des sommets : A, B, C, D, E, F)

$$M = \begin{bmatrix} & A & B & C & D & E & F \\ A & 0 & 1 & 0 & 0 & 1 & 1 \\ B & 1 & 0 & 1 & 1 & 0 & 0 \\ C & 0 & 1 & 0 & 1 & 0 & 1 \\ D & 0 & 1 & 1 & 0 & 1 & 0 \\ E & 1 & 0 & 0 & 1 & 0 & 1 \\ F & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Remarque : La matrice est symétrique car G est non-orienter. On constate que la diagonale comporte que des zéros, elles est donc aussi anti-réflexif on constate bien sur le graphes aucunes boucles. (+0.5 pt).

5. **Matrice d'incidence sommet-arcs** (1 pt)

La première étapes consiste a numéroter les arcs :



La matrice comporte 6 lignes (sommets) et 9 colonnes (arcs). Chaque colonne contient deux 1 pour les deux extrémités de l'arête non orientée :

$$A = \begin{bmatrix} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \\ A & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ B & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ C & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ D & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ E & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ F & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

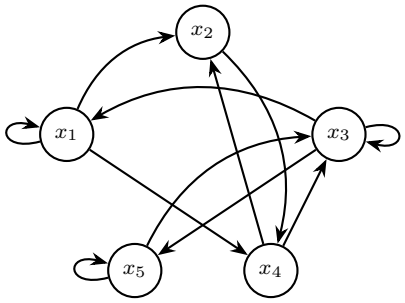
6. **Liste d'adjacence** (1 pt)
(Bonus +1 pt en version LP/LS)

$$A : \{B, E, F\} \qquad B : \{A, C, D\}$$
$$C : \{B, D, F\} \qquad D : \{B, C, E\}$$
$$E : \{A, D, F\} \qquad F : \{A, C, E\}$$

7. **Nombre de chemins de longueur 2 entre** A **et** D (0,5 pt)
Il existe **2** chemins de longueur 2 : $A - B - D$ et $A - E - D$.

2 Reconstruction d'un graphe (4.0-5.0 points)

1. **Reconstruction du graphe** (0,5 pt)
   L'application multivoque $\Gamma^{-1}(x_i)$ correspond a la liste des **predecesseurs**. On peut donc reconstruire les arcs suivants : $x_1 \to x_1$, $x_3 \to x_1$, $x_1 \to x_2$, $x_4 \to x_2$, $x_3 \to x_3$, $x_4 \to x_3$, $x_5 \to x_3$, $x_2 \to x_4$, $x_1 \to x_4$, $x_3 \to x_5$, $x_5 \to x_5$. Et donc le graphe suivant :



2. **Ensembles $W^+(x_i)$ et $W^-(x_i)$** (1 pt)
   Les applications $W^+(x_i)$ et $W^-(x_i)$ correspondent simplement aux listes des arcs u_i associees a x_i. Dit autrement, $W^+(x_i)$ contient tous les arcs qui commence par x_i, donc sous la forme $(\mathbf{x_i}, x_j)$ et $W^-(x_i)$ qui finissent par x_i donc sous la forme $(x_j, \mathbf{x_i})$.

   $$W^+(x_1) = \{(x_1, x_1), (x_1, x_2), (x_1, x_4)\} \qquad W^-(x_1) = \{(x_1, x_1), (x_3, x_1)\}$$
   $$W^+(x_2) = \{(x_2, x_4)\} \qquad W^-(x_2) = \{(x_1, x_2), (x_4, x_2)\}$$
   $$W^+(x_3) = \{(x_3, x_1), ((x_3, x_3), (x_3, x_5)\} \qquad W^-(x_3) = \{(x_3, x_3), (x_4, x_3), (x_5, x_3)\}$$
   $$W^+(x_4) = \{(x_4, x_2), (x_4, x_3)\} \qquad W^-(x_4) = \{(x_1, x_4), (x_2, x_4)\}$$
   $$W^+(x_5) = \{(x_3, x_5), (x_5, x_5)\} \qquad W^-(x_5) = \{(x_3, x_5), (x_5, x_5)\}$$

   Donc si vous avez écris $W^+(x_1) = \{x_1, x_2, x_4\}$ ... rigoureusement ... c'est faux ...

3. **Demi-degrés** (0.5 pt)
   Les demi-degrees sont donnees par les applications δ^+ et δ^- respectivement entrant et sortant. Ils sont simple a calculer via le cardinal des applications W et Γ associees. Par exemple $\delta^+(x_i) = |W^+(x_i)| = |\Gamma^+(x_i)|$ et idem pour $\delta^-(x_i)$.

   $$\delta^+(x_1) = 3,\ \delta^-(x_1) = 2$$
   $$\delta^+(x_2) = 1,\ \delta^-(x_2) = 2$$
   $$\delta^+(x_3) = 3,\ \delta^-(x_3) = 3$$
   $$\delta^+(x_4) = 2,\ \delta^-(x_4) = 2$$
   $$\delta^+(x_5) = 2,\ \delta^-(x_5) = 2$$

   Pour rappel, $\delta(x_i) = \delta^+(x_i) + \delta^-(x_i)$, dans le cas de boucle on s'appercois donc qui la boucle compte pour 2 dans $\delta(x_i)$ car elle cnotribue a la fois a $\delta^+(x_i)$ et a $\delta^-(x_i)$ (+0.5 pt).

4. **Matrice d'adjacence** A (1 pt)

   $$A = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

5. **Matrice diagonale des demi-degrés et matrice laplacienne** (0.5 pt)
   $D = \text{diag}(3, 1, 3, 2, 2)$ et Matrice laplacienne $L = D - A$. (+0.5 si matrices complete)

3 Parcours et détection de cycles (4.0-7.0 points)

1. **Parcours en profondeur (DFS) depuis 1** (2 pt)
   Un ordre de découverte possible version simple :
   — On commence avec $S = [1]$
   — $U = 1; S[]; \Gamma(1) = [2, 5] \implies S = [2, 5]$
   — $U = 5; S[2]; \Gamma(5) = [2, 4] \implies S = [2, 2, 4]$
   — $U = 4; S[2, 2]; \Gamma(4) = [2] \implies S = [2, 2, 2]$
   — $U = 2; S[2, 2]; \Gamma(2) = [3] \implies S = [2, 2, 3]$
   — $U = 3; S[2, 2]; \Gamma(3) = [1] \implies S = [2, 2, 1]$
   — $U = 1; S[2, 2]$ déjà visité (arrête arrière ou avant)
   — $U = 2; S[2]$ déjà visité (arrête arrière ou avant)
   — $U = 2; S[]$ déjà visité (arrête arrière ou avant)
   — 6 n'est pas visité on relance l'algo $S = [6]$
   — $U = 6; S[]; \Gamma(6) = [3] \implies S = [3]$
   — $U = 3; S[]$ déjà visité (arrête transverse car reinit juste avant)
   On a donc le parcours suivant : $1 - 5 - 4 - 2 - 3 - 6$.

   Dans cette version simple on ne peux pas caractériser correctement les types d'arcs. Pour cela il faut utiliser la version complète de l'algorithme avec les notions de couleurs pour garder la trace des états. On ne peu pas non plus obtenir les temps de découvertes d et de fins f également associées aux notions de couleurs (passage $blanc \to gris$ et $gris \to noire$).

   Pour la version longue je met simplement un résultat intermediaire et final ici :

Sommet	d	f	π	couleur	commentaire
1	1	-	-	gray	detection avant 1→2 a t=10
5	2	-	1	gray	detection avant 5→2 a t=9
4	3	-	5	gray	
2	4	7	4	black	
3	5	6	2	black	
6	-	-	-	white	

TABLE 1 – Résultats du DFS avec timestamps d'entrée (d) et de fin (f), prédécesseur (π), couleur finale et type d'arc. L'ordre de découverte est $1 \to 5 \to 4 \to 2 \to 3 \to 6$.

Sommet	d	f	π	couleur
1	1	10	-	black
5	2	9	1	black
4	3	8	5	black
2	4	7	4	black
3	5	6	2	black
6	11	12	-	black

TABLE 2 – Résultats du DFS avec timestamps d'entrée (d) et de fin (f), prédécesseur (π), couleur finale et type d'arc. L'ordre de découverte est $1 \to 5 \to 4 \to 2 \to 3 \to 6$.

2. **Caractérisation des arcs** (1 pt)

   Definition : (+1 point)
   — Tree / Arbre : l'arc mène vers un sommet blanc (nouveau sommet).
   — Back / Arriere : l'arc mène vers un sommet gris (ancêtre actuellement dans la pile de récursion).
   — Forward / Avant : l'arc mène vers un sommet noir déjà découvert, avec $d[u] < d[v]$ (descendant dans l'arbre DFS).
   — Cross / Transverse : l'arc mène vers un sommet noir déjà découvert, avec $d[u] > d[v]$ (entre deux sous-arbres déjà terminés).

d	Arc ($u \to v$)	Type d'arc	Explication (selon la couleur de v au moment de l'examen)
1	$1 \to 5$	Arbre	v blanc
2	$5 \to 4$	Arbre	v blanc
3	$4 \to 2$	Arbre	v blanc
4	$2 \to 3$	Arbre	v blanc
5	$3 \to 1$	Arriere	v gris (ancêtre dans la pile)
9	$5 \to 2$	Avant	v noire ($d[u] = 2 < d[v] = 4$)
10	$1 \to 2$	Avant	v noire ($d[u] = 1 < d[v] = 4$)
11	$6 \to 3$	Transverse	v noire ($d[u] = 11 > d[v] = 5$)

TABLE 3 – Classification des arcs rencontrés durant le DFS avec l'ordre de découverte.

3. **Le graphe contient-il un cycle ?** (0,5 pt)
   Oui, exemple : **1 → 2 → 3 → 1**. Justification : présence d'un **arc arrière** (3→1) lors du DFS avec coloration (gris). Voir même plus directe, $1 \to 5 \to 4 \to 2 \to 3 \to 1$ en suivant le parcours DFS.
4. **Ordre de découverte et de finition** (0,5 pt)
   La découverte des nœuds : $1 - 5 - 4 - 2 - 3 - 6$ correspond au temps de découverte suivant $1 - 2 - 3 - 4 - 5 - 11$. L'ordre de finition est donc $10 - 9 - 8 - 7 - 6 - 12$. Donc on recopie directement les colonnes du tableaux.
5. Écrivez une version simple de l'algorithme DFS en python (bonus +2 pts)

```python
# Pour simplifier L ici est une liste d'adjacence
def dfs_iterative(L, start):
    visited = [False] * len(L)
    stack   = [start]
    while stack:
        u = stack.pop()
        if not visited[u]:
            visited[u] = True
            stack = stack + L[u]
            # ou list(reversed(L[u])) pour l'ordre
    return visited
```

Ici avec 3 exercices maîtrisés vous pouviez avoir entre 13 et 20.

Temps estimer ±45min, le temps de sécuriser quelques points en plus.

4 Algorithmes de plus court chemin (4.0-5.0 points)

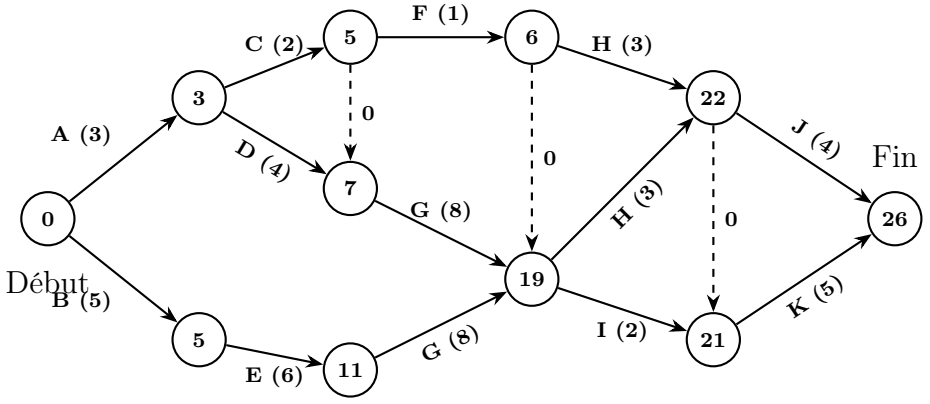
1. **Application de Dijkstra depuis** S (3,5 pts)
   L'algorithme de Dijkstra explore les sommets par ordre de distance croissante. Dans le tableau ci-dessous, chaque cellule contient la valeur $d(v)$ (distance depuis S) et le prédécesseur entre parenthèses. Le sommet choisi à chaque étape est indiqué en **gras**.

Étape	**Visité**	S	A	B	C	D	T
Initialisation	$\emptyset$	**0 (-)**	∞	∞	∞	∞	∞
1	$\{S\}$	0	4 (S)	**2 (S)**	∞	∞	∞
2	$\{S, B\}$	0	**4 (S)**	2	4 (B)	7 (B)	∞
3	$\{S, B, A\}$	0	4	2	**4 (B)**	7 (B)	∞
4	$\{S, B, A, C\}$	0	4	2	4	7 (B)	**5 (C)**
5	$\{S, B, A, C, T\}$	0	4	2	4	**7 (B)**	5

TABLE 4 – Évolution des distances et sélection des sommets.

2. **Plus court chemin de** S **à** T (0,5 pt)
   Prédécesseurs finaux : T ← C ← B ← S. Ce qui donne le chemin : $S \to B \to C \to T$ pour une longueur de **5** (dernier poid).

1. L'objectif de cet exercice est de valider la cohérence du réseau fourni. Une analyse rigoureuse montre que les valeurs inscrites dans les nœuds de l'énoncé comportent des erreurs de calcul. Si vous êtes passer a coté vous perdez juste les points de la question.



Par contre les lien fictifs manques les valeurs associées ! Ce qui poseras probleme pour le calcule des datas au plus tard. Dans un réseau PERT, les tâches fictives (dummies) ont par définition une durée de 0 (Ou aussi noté en France par exemple $D'(4)$ a la place de 0 pour l'arc $5 \to 7$). Cependant, elles imposent des contraintes de précédence qui modifient radicalement le calcul des dates au plus tard et les marges si elles sont ignorées. (+1.0 pts)

Modifications apportées pour la correction :

(a) **Tâches fictives (Dummies) :** Elles ont maintenant une valeur de **0** explicitée. Sans cela, le calcul du "plus tard" au nœud 8 par exemple serait faussé car il doit tenir compte du fait que le nœud 9 dépend de lui via une durée nulle ($T_8 = \min(T_{10} - 4, T_9 - 0)$).

(b) **Nœud 7 :** Rectifié à **19**. L'erreur de l'énoncé (15) ignorait la branche $B \to E \to G$.

(c) **Nœud 9 :** Rectifié à **21** $(19 + 2)$ car il est sur le chemin critique.

(d) **Nœud 10 :** Rectifié à **26** $(21 + 5)$. C'est la durée minimale réelle du projet.

2. Effectuer le calcul des dates au plus tard (1 pts) On utilise la date de fin corrigée ($T_{10} = 26$). La formule est : $T_i = \min_j(T_j - d_{ij})$.

Nœud	10	9	8	7	6	5	4	3	2	1	0
E_i **(corrigé)**	26	21	22	19	6	11	7	5	5	3	0
T_i **(calculé)**	26	21	21	19	16	11	11	11	5	7	0

3. Calculer les marges (0.5 pt) La marge totale d'une tâche est $MT = T_{succ} - E_{prec} - \text{durée}$.

Tâche	**Début**	**Fin**	**Durée**	**Calcul (MT)**	**Marge**	**Critique**
A	0	1	3	$7 - 0 - 3$	4	
B	0	2	5	$5 - 0 - 5$	**0**	**Oui**
C	1	3	2	$11 - 3 - 2$	6	
D	1	4	4	$11 - 3 - 4$	4	
E	2	5	6	$11 - 5 - 6$	**0**	**Oui**
F	3	6	1	$16 - 5 - 1$	10	
G (5-7)	5	7	8	$19 - 11 - 8$	**0**	**Oui**
H (7-8)	7	8	3	$22 - 19 - 3$	**0**	**Oui**
I	7	9	2	$21 - 19 - 2$	**0**	**Oui**
J	8	10	4	$26 - 22 - 4$	**0**	**Oui**
K	9	10	5	$26 - 21 - 5$	**0**	**Oui**

4. Identifier le(s) chemin(s) critique(s) (0.5 pt)

L'analyse montre qu'il existe deux chemins critiques car les deux branches partant du nœud 7 (19) (via H-J et via I-K) présentent une durée identique de 7 unités.

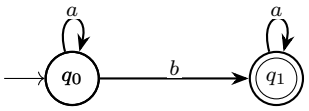
— **Chemin Critique 1 :** $0 \xrightarrow{B} 2 \xrightarrow{E} 5 \xrightarrow{G} 7 \xrightarrow{H} 8 \xrightarrow{J} 10$

— **Chemin Critique 2 :** $0 \xrightarrow{B} 2 \xrightarrow{E} 5 \xrightarrow{G} 7 \xrightarrow{I} 9 \xrightarrow{K} 10$

5. Calculer la durée totale minimale du projet (0.5 pt) La durée totale minimale du projet, après correction des erreurs de calcul cumulées dans l'énoncé, est de **26 unités de temps**. Lié a la question 1.

5 Expressions régulières (2.0 points)

1. Les expressions régulières et les automates finis peuvent être modélisés par des graphes orientés où (1 pt) :
   — les nœuds représentent les **états** (début, transitoires, acceptants)
   — les arcs étiquetés représentent les **transitions** sur les symboles de l'alphabet
   — l'état initial a une flèche entrante sans étiquette
   — les états acceptants sont souvent doublés ou marqués
   — bien-sur il était accepté des représentations similaire moins strictes
2. **Graphe de l'automate pour** $r = a^*ba^*$ (1 pt)
   États : q_0 (initial), q_1 (vu le b), q_2 (acceptant).
   Transitions : $q_0 \xrightarrow{a} q_0$, $q_0 \xrightarrow{b} q_1$; $q_1 \xrightarrow{a} q_2$; $q_2 \xrightarrow{a} q_2$ et q_2 est l'état acceptant.
   Graphiquement :



6 Diffusion discrète : analyse critique (3.0-5.0 points)

L'équation $\frac{\partial f}{\partial t} = -Lf$ décrit un processus de **diffusion** (ou chaleur) sur le graphe. La matrice laplacienne L encode la structure locale du graphe (différences entre un sommet et ses voisins). Au fil du temps, les valeurs $f(t)$ s'homogénéisent : elles tendent vers la moyenne globale (consensus).

1. **Analysez la formule suivante qui fait intervenir des notions de propriétés spectrales et de fermeture transitives.** (x pts)
   La formule proposée semble tenter de combiner diffusion et fermeture transitive (somme des puissances de A). Cependant, elle est problématique car :
   — le signe est incorrect (devrait être $-L$ pour la diffusion) dit autrement cela renforce les dissensus ; vers la création d'aggloméras et donc le graphes s'hétérogènéise.
   — la somme $I + A + \cdots + A^{n-1}$ est la fermeture transitive (chemins de toute longueur), ce qui ne correspond pas à un opérateur de diffusion local ;
   — elle mélange des notions spectrales et de connectivité globale de manière non justifiée pour cette PDE (équation des dérivées partielles).

   Dit autrement il n'y avais aucun sens a chercher (petit easter egg pour un étudiant) mais visiblement personne n'a trouvé. Donc simplement une question bonus et de remise en question (+2 pt).

7 Algorithme de Bellman-Ford et cycle absorbant (5.0-5.5 points)

1. **Premières itérations de Bellman-Ford depuis** S (3 pts)
   Voici les premières itérations de l'algorithme à partir du sommet source S. On initialise $d(S) = 0$ et $d(v) = \infty$ pour les autres. Concernant le nombre d'itération, ce dernier est borner par la définition suivante : Dans un graphe à n sommets sans cycle absorbant, tout plus court chemin contient au plus $n - 1$ arcs, donc ici $7 - 1 = 6$ itérations.

Itération	$d(S)$	$d(A)$	$d(B)$	$d(C)$	$d(D)$	$d(E)$	$d(T)$
Initialisation	0	∞	∞	∞	∞	∞	∞
Itération 1	0	4	2	∞	∞	∞	∞
Itération 2	0	4	2	0	∞	∞	∞
Itération 3	0	4	2	0	-4	2	∞
Itération 4	0	4	1	0	-4	-1	8
Itération 5	0	4	1	-1	-5	-1	5

   À l'itération 4 : Le cycle commence à réduire les distances : $d(B)$ devient $\min(2, d(D)+5) = 1$. $d(E)$ est mis à jour via la nouvelle valeur de D : $\min(2, -4+3) = -1$. $d(T)$ est mis à jour via l'ancienne valeur de E ($2+6 = 8$). **À partir de l'itération 5 :** on constate que $d(C)$ diminue encore ($d(B) - 2 = -1$). Comme le poids du cycle $B \to C \to D \to B$ est de -1, chaque itération supplémentaire réduira les distances de B, C, D, E et T de 1 unité.

2. **Itération supplémentaire** (1 pt)
   Après $|V|-1$ itérations, si on effectue une itération de plus et qu'au moins une distance diminue encore, alors il existe un **cycle absorbant** (de poids négatif) accessible depuis S. Ici, le cycle C → D → B (via l'arc -4) est absorbant et accessible. La somme total des transition est $-2 - 4 + 5 = -1$, donc si on tourne 5 fois dans le cycle on obtient un chemin de longueur -5 ce qui est possible mais probablement incohérent. Donc on pourrais prendre n'importe quel chemin, très long, puis tomber dans le cycle pour réduit d'autant, pour n'importe quelles longueurs.

3. **Explication et pseudo-solution** (1 pt)
   En présence d'un cycle absorbant, les distances peuvent être rendues arbitrairement petites (on peut boucler indéfiniment sur le cycle négatif). Il n'existe donc plus de « plus court chemin » bien défini et fini, car on peut réduire indéfiniment le coût en bouclant. Cela invalide la notion même de distance minimale pour certaines paires de sommets. **Pseudo-solution** : exécuter Bellman-Ford et détecter le cycle négatif à la $|V|$-ième itération. On peut ensuite supprimer les arcs du cycle ou avertir l'utilisateur qu'aucun réel plus court chemin n'existe. **Détournement** : Par ailleurs, l'utilisation de poids négatifs peut être exploitée pour influencer le chemin calculé et favoriser le passage par certaines parties du graphe. Cependant, il est essentiel d'éviter la création de cycles négatifs, qui rendent le problème mal défini (+0.5).

Cette seconde partie offrais entre 14 et 16.5 points (ou plus en fonction des réponsses).