Corrigé du TD 2

1. Dijkstra

La solution la plus simple consiste à effectuer deux appels à l'algorithme de Dijkstra sur le graphe G', construit à partir du graphe G dont on a enlevé l'arête négative, une fois avec comme point de départ l'origine x_0, une seconde fois avec comme point de départ l'extrémité de l'arête négative t. La complexité est bien de l'ordre de grandeur requis, et la terminaison est garantie, puisqu'il s'agit de deux appels à l'algorithme de Dijkstra dans un graphe n'ayant que des arêtes positives.

Preuve On remarque qu'il y a deux possibilités pour chaque sommet x :

- Le plus court chemin de s à x passe par l'arête négative ; dans ce cas, il se décompose en un plus court chemin de x_0 à s, l'arête (s,t), et un plus court chemin de t à x.
- Le plus court chemin de s à x ne passe pas par l'arête négative ; dans ce cas, il est trouvé dans le premier appel à Dijkstra, sur le graphe ne contenant pas l'arête négative.

Pour que ce raisonnement soit valide, il faut garantir qu'il existe un plus court chemin contenant au plus une fois l'arête négative. Soit c un plus court chemin contenant l'arête négative un nombre de fois minimal. S'il contient plusieurs fois l'arête, il contient un cycle $s \to t \to \cdots \to s$. La suppression de ce cycle construit un chemin de x_0 à x contenant l'arête négative une fois de moins que c, et qui est donc nécessairement strictement plus long que c ; dans ce cas, le cycle construit plus haut est de longueur strictement négative, et le plus court chemin est de longueur $-\infty$; le problème n'a donc pas de sens dans ce cas, sauf pour les sommets qui ne sont pas accessibles depuis le cycle de longueur négative.

Ce problème est détecté en cours d'algorithme simplement en remarquant que l'existence d'un cycle de longueur négative dans G revient à affirmer l'existence d'un chemin de t à s dans G' de longueur inférieure à $|l(s,t)|$; or le chemin le plus court de t à s dans G' est calculé lors du second appel à Dijkstra.

2. Roy-Warshall sur un exemple

On initialise avec la fiabilité de tous les arcs, les arcs absents étant considérés comme de fiabilité nulle (dans les matrices j'ai écrit les inverses des fiabilités pour alléger les notations) :

$$\begin{pmatrix} 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 & 2 \\ 0 & 3 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 & 3 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Ensuite les $n = 5$ matrices qu'on trouve sont successivement :

$$\begin{pmatrix} 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 & 2 \\ 0 & 3 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 & 3 \\ 1 & 0 & \mathbf{2} & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 & 2 \\ 0 & 3 & 0 & 1 & \mathbf{6} \\ 0 & 2 & \mathbf{6} & 0 & 3 \\ 1 & 0 & 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & \mathbf{6} & 2 & \mathbf{2} & 0 \\ 0 & 0 & 4 & \mathbf{4} & 2 \\ 0 & 3 & 0 & 1 & 6 \\ 0 & 2 & 6 & 0 & 3 \\ 1 & \mathbf{6} & 2 & \mathbf{2} & 0 \end{pmatrix} \begin{pmatrix} 0 & \mathbf{4} & 2 & 2 & \mathbf{6} \\ 0 & 0 & 4 & 4 & 2 \\ 0 & 3 & 0 & 1 & \mathbf{3} \\ 0 & 2 & 6 & 0 & 3 \\ 1 & \mathbf{4} & 2 & 2 & 0 \end{pmatrix} \begin{pmatrix} 0 & 4 & 2 & 2 & 6 \\ \mathbf{2} & 0 & 4 & 4 & 2 \\ \mathbf{3} & 3 & 0 & 1 & 3 \\ \mathbf{3} & 2 & 6 & 0 & 3 \\ 1 & 4 & 2 & 2 & 0 \end{pmatrix}$$

(en gras les entrées modifiées). A la fin les seules entrées nulles sont sur la diagonales puisque le graphe est fortement connexe.

On trouve ensuite le chemin de fiabilité maximale entre i et j en cherchant le dernier temps k auquel l'entrée (i, j) a été modifiée, ce qui nous dit que le chemin de i à j passe par k, puis en recherchant récursivement les meilleurs chemins de i à k et de k à j utilisant des sommets internes $\leq k$. Ceci se fait en temps linéaire si on a conservé toutes les matrices (espace n^3).

3. Algorithme de Viterbi

i. Il suffit de parcourir le graphe en largeur ; partant du sommet v_0, on suit toutes les arêtes d'étiquette s_1, et on construit la liste des sommets d'arrivée ; pour chacun de ces sommets, on suit alors les arêtes d'étiquette s_2, etc.

Si le graphe a $|S|$ sommets, la complexité est en $O(n|S|^2)$.

ii. Soit $c = v_0 v_1 \ldots v_n$ un chemin le plus probable partant de v_0 d'étiquette s ; alors le chemin $v_1 \ldots v_n$ est un chemin le plus probable partant de v_1 et d'étiquette $s_2 \ldots s_n$.

Posons alors $f(v, i)$ le chemin le plus probable issu de v et dont l'étiquette est $s[i..n]$. On a dans ce cas la relation de récurrence

$$f(v, i) = \max_{u ; \text{étiq}(v,u) = s_i} p(v, u) f(u, i + 1),$$

avec la condition initiale $f(v, n + 1) = 1$ pour tout v, qui nous donne un schéma de programmation dynamique. La complexité est dans ce cas $O(n|S|^2)$, car le calcul d'un terme de la table (qui est de taille $O(n|S|)$) prend un temps $O(|S|)$.