Algorithmique
Plus courts chemins et ARPM

Spé S4 Epita

Correction Examen B8

14 mai 2025

Solution 1 **(Longest-shortest –** ***9 points*****)**

1. 3 plus courts chemins (tous de coût 11) entre les sommets 0 et 7 dans le graphe `G2` :
   - Path 1 = $0 \to 3 \to 5 \to 7$
   - Path 2 = $0 \to 1 \to 6 \to 4 \to 7$
   - Path 3 (longest-shortest) = $0 \to 3 \to 5 \to 1 \to 6 \to 4 \to 7$

2. **Spécifications :** La fonction `longest_shortest(G, src, dst)` retourne, s'il existe, un *longest-shortest path* entre les deux sommets distincts `src` et `dst` dans le graphe orienté `G` à coûts strictement positifs, son coût et sa longueur. Si le chemin entre `src` et `dst` n'existe pas dans `G` la fonction retourne `None`.

   Graphe à coûts strictement positifs ⇒ Dijkstra.

```python
def __build_path(P, dst):
    # build the path to dst using P: predecessor/parent vector
    path = []
    while dst != -1:
        path.append(dst)
        dst = P[dst]
    path.reverse()
    return path

def longest_shortest(G, src, dst):
    dist = [infty] * G.order
    dist[src] = 0
    P = [None] * G.order
    P[src] = -1
    length = [0] * G.order
    H = heap.Heap(G.order)
    H.push(src, 0)
    while not H.is_empty():
        (_, x) = H.pop()
        if x == dst:
            return (__build_path(P, dst), dist[dst], length[dst])
        for y in G.adjlists[x]:
            c = dist[x] + G.costs[(x, y)]
            if dist[y] == c:
                if length[x] + 1 > length[y]:
                    length[y] = length[x] + 1
                    P[y] = x
            elif c < dist[y]:
                dist[y] = c
                P[y] = x
                length[y] = length[x] + 1
                H.update(y, c)
    return None
```

***Solution 2* (What is this ? – *3 points*)**

Lors de l'appel `mystery(`G_3`, 0, 7)` avec G_3 le graphe de la figure 3 :

1. Liste `L` construite par l'appel `aux(G, src)`, ligne 30 :

   `[7, 4, 6, 1, 5, 3, 2, 0]`

2. Vecteurs `dist1` et `P1` résultats du premier appel à `shortestpath` ligne 32 :

	0	1	2	3	4	5	6	7	8	9	10
`dist1`	0	2	8	1	7	3	3	8	∞	∞	∞
`P1`	-1	3	0	0	6	3	1	6	None	None	None

3. Vecteurs `dist2` et `P2` résultats du deuxième appel à `shortestpath` ligne 36 :

	0	1	2	3	4	5	6	7	8	9	10
`dist2`	0	4	∞	∞	10	∞	∞	12	∞	∞	∞
`P2`	-1	0	None	None	1	None	None	4	None	None	None

4. Résultat final retourné par `mystery(`G_3`, 0, 7)` : `(8, 12)`

***Solution 3* (ARM ou pas ? – *3 points*)**

Le principe énoncé ressemble à celui de Prim, mais **le graphe partiel T n'est pas un ARM de G** !

Justifications

- L'obligation de repartir de l'extrémité de la dernière arête ajoutée :
  - si un isthme est choisi, on ne pourra plus revenir en arrière : l'algorithme ne peut pas terminer, on ne pourra pas choisir $n-1$ arêtes ; (Pb 1)
  - l'algorithme peut également avoir à choisir une arête de poids élevé. (Pb 2)
- Il n'y a pas de gestion de deux ensembles de sommets complémentaires : celui des sommets déjà reliés à la source s et l'ensemble de ceux qui ne le sont pas encore.
  Le problème dans ce cas est de ne pas gérer l'acyclicité. (Pb 3)

Exemple

Si le sommet de départ est 3 dans le graphe ci-dessous :
- Après la première arête choisie (3-4), obligation de recommencer depuis 4 : on ne pourra plus revenir en arrière, les sommets 0, 1 et 2 ne seront jamais atteints (Pb 1).
- L'algorithme continue vers les sommets 5 puis 7. Après l'arête 7-6, c'est l'arête 6-4 qui sera choisie, alors qu'elle ne fait pas partie de la solution (Pb 2) et crée un cycle (Pb 3).

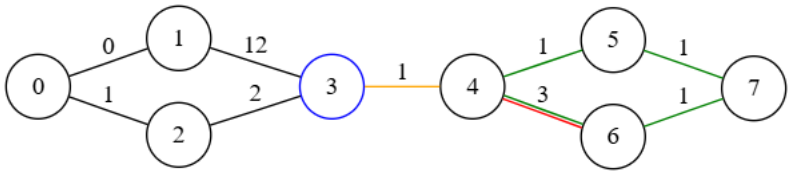

FIGURE 1 – Start from 3

***Solution 4* (ARPM via Kruskal – *2 points*)**

1. Les arêtes retenues sont :

   `{B,F,1},{B,G,2},{B,C,4},{G,H,5},{A,E,6},{A,D,7},{A,B,10}`

2. Les arêtes rejetées sont :

   `{F,G,3},{C,F,8},{D,E,9}`