Algorithme de Dijkstra — Étapes détaillées

But : Déterminer les plus courts chemins qui relient A et les autres sommets

Principe général de l'algorithme

— On commence avec une distance de 0 pour le sommet de départ, et ∞ (infinie) pour les autres.
— À chaque étape, on traite le sommet non visité le plus proche.
— On met à jour les distances de ses voisins si on trouve un chemin plus court.
— On répète jusqu'à ce que tous les sommets soient traités.

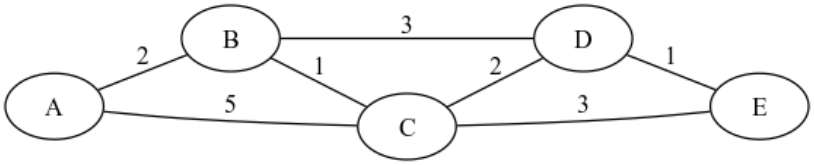

FIGURE 1 – Graphe pondéré utilisé pour l'algorithme de Dijkstra

Étape 1 – Traitement du sommet A

Initialisation

— `distances = {'A': 0}`
— `predecesseurs = {'A': None}`
— `a_traiter = [(0, 'A')]`

On extrait `(0, 'A')`, donc :

— `distance = 0`, `sommet = 'A'`
— `a_traiter = []`
— On ajoute `'A'` à `deja_affectes`
— `graphe['A'] = [(2, 'B'), (5, 'C')]`

Traitement du voisin B (poids 2)

— `nouvelle_distance (AB) = 0 (AA) + 2 (AB) = 2`
— `distances['B'] (AB) = 2`
— `predecesseurs['B'] = 'A'`
— `a_traiter.append((2, 'B'))`

Traitement du voisin C (poids 5)

— `nouvelle_distance (AC) = 0 (AA) + 5 (AC) = 5`
— `distances['C'] = 5`

- predecesseurs['C'] = 'A'
- a_traiter.append((5, 'C'))

Fin de l'étape 1

- a_traiter = [(2, 'B'), (5, 'C')] (triée)
- distances = {'A': 0, 'B': 2, 'C': 5}
- predecesseurs = {'A': None, 'B': 'A', 'C': 'A'}

Étape 2 – Traitement du sommet B

- a_traiter = [(2, 'B'), (5, 'C')] → on extrait (2, 'B')
- distance = 2, sommet = 'B'
- On ajoute 'B' à deja_affectes
- graphe['B'] = [(2, 'A'), (1, 'C'), (3, 'D')]

Voisin A (déjà affecté)

- On ignore A

Voisin C (poids 1)

- nouvelle_distance (AC) = 2 (AB) + 1 (BC) = 3
- 3 (AC) < 5 (ancienne valeur de AC) donc :
  - distances['C'] = 3
  - predecesseurs['C'] = 'B'
  - a_traiter.append((3, 'C'))

Voisin D (poids 3)

- nouvelle_distance (AD) = 2 (AB) + 3 (BD) = 5
- D est nouveau :
  - distances['D'] = 5
  - predecesseurs['D'] = 'B'
  - a_traiter.append((5, 'D'))

Fin de l'étape 2

- a_traiter = [(3, 'C'), (5, 'C'), (5, 'D')] (triée)
- distances = {'A': 0, 'B': 2, 'C': 3, 'D': 5}
- predecesseurs = {'A': None, 'B': 'A', 'C': 'B', 'D': 'B'}

Étape	**Sommet traité**	`a_traiter`	`distances`	`prédecesseurs`
0	—	`[(0, 'A')]`	`'A': 0`	`'A': None`

1	A	`[(2, 'B'), (5, 'C')]`	`'A': 0`
			`'B': 2`
			`'C': 5`
				`'A': None`
				`'B': 'A'`
				`'C': 'A'`

2	B	`[(3, 'C'), (5, 'C'), (5, 'D')]`	`'A': 0`
			`'B': 2`
			`'C': 3`
			`'D': 5`
				`'A': None`
				`'B': 'A'`
				`'C': 'B'`
				`'D': 'B'`

3	C	`[(5, 'D'), (6, 'E'), (5, 'C')]`	`'A': 0`
			`'B': 2`
			`'C': 3`
			`'D': 5`
			`'E': 6`
				`'A': None`
				`'B': 'A'`
				`'C': 'B'`
				`'D': 'B'`
				`'E': 'C'`

4	D	`[(6, 'E'), (6, 'E')]`	*inchangé*	*inchangé*
5	E	`[(6, 'E')]`	*inchangé*	*inchangé*
6	E (déjà vu)	`[]`	*inchangé*	*inchangé*

TABLE 1 – Étapes de l'algorithme de Dijkstra avec les valeurs ligne par ligne