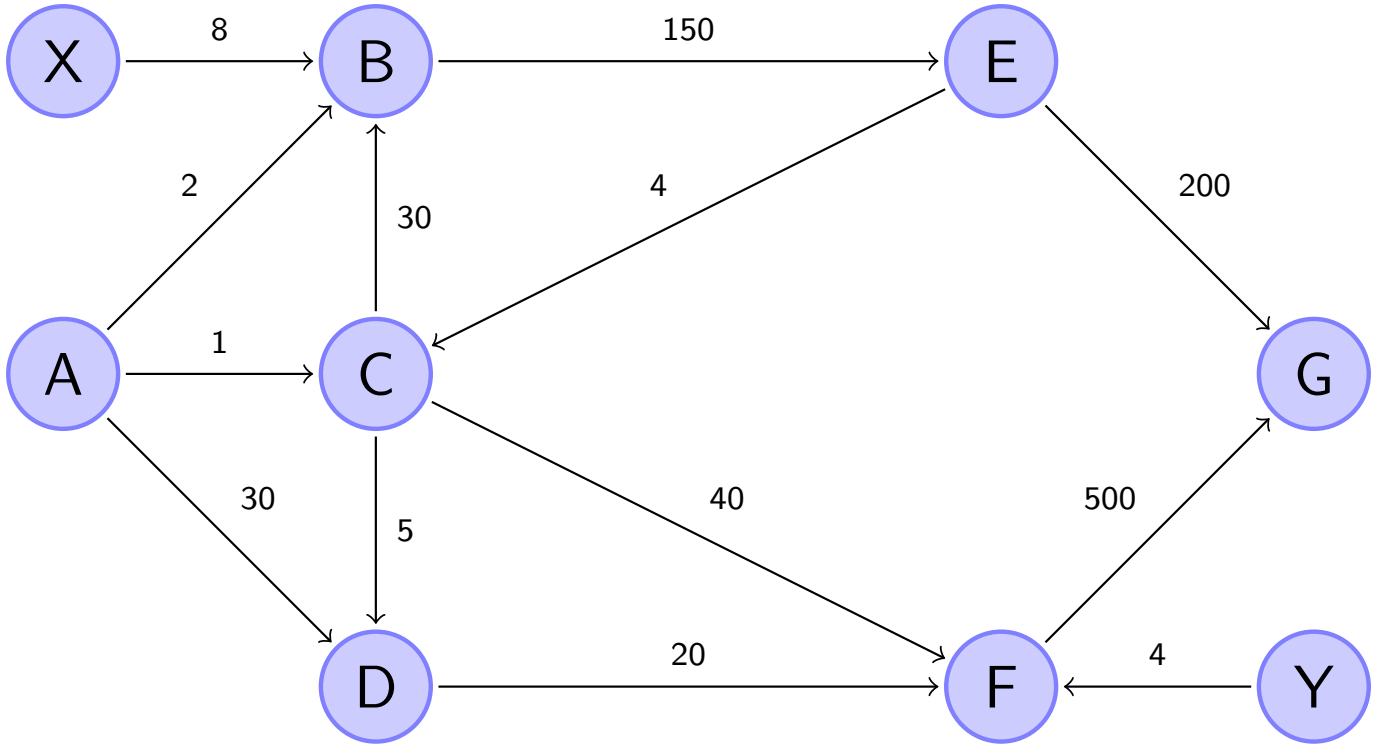
Intelligence Artificielle
Construire un plus court chemin

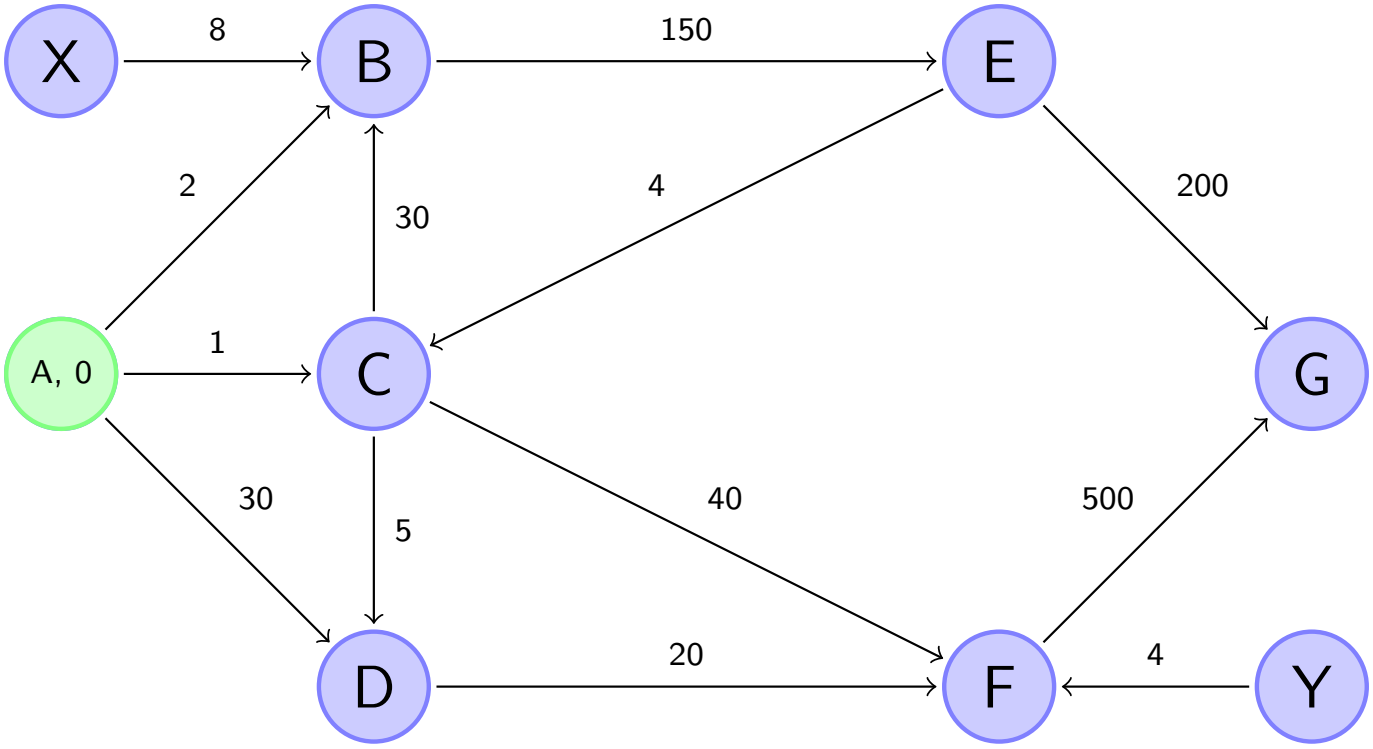
José Vander Meulen

19 octobre 2017

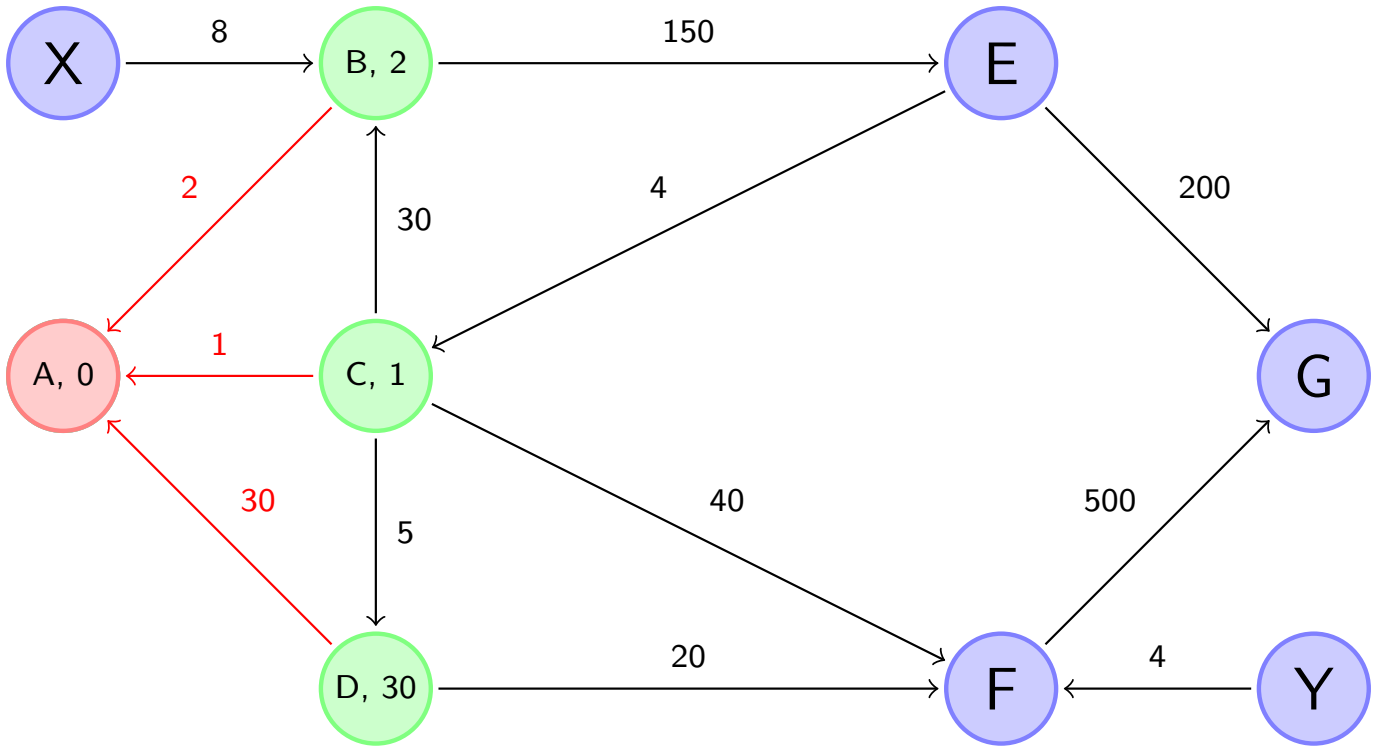
Algorithme de Dijkstra : example



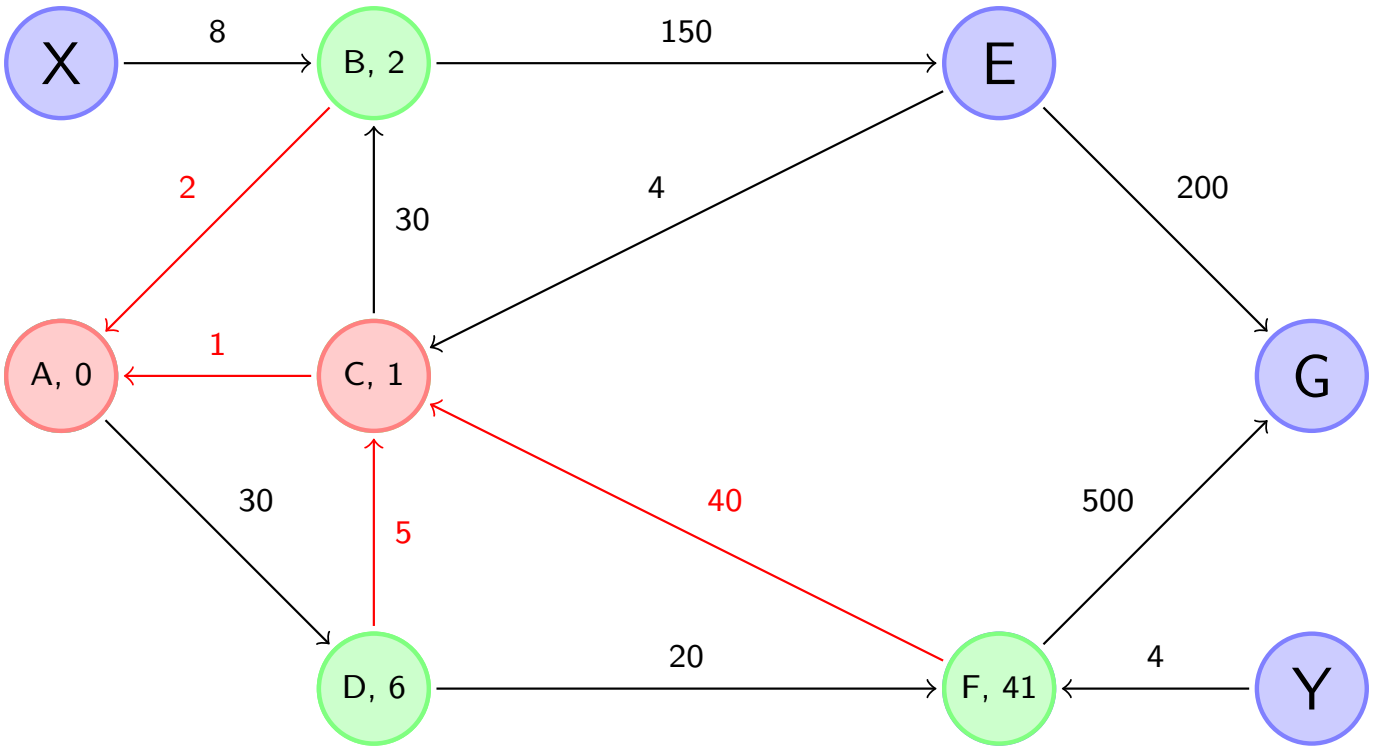
Algorithme de Dijkstra : example



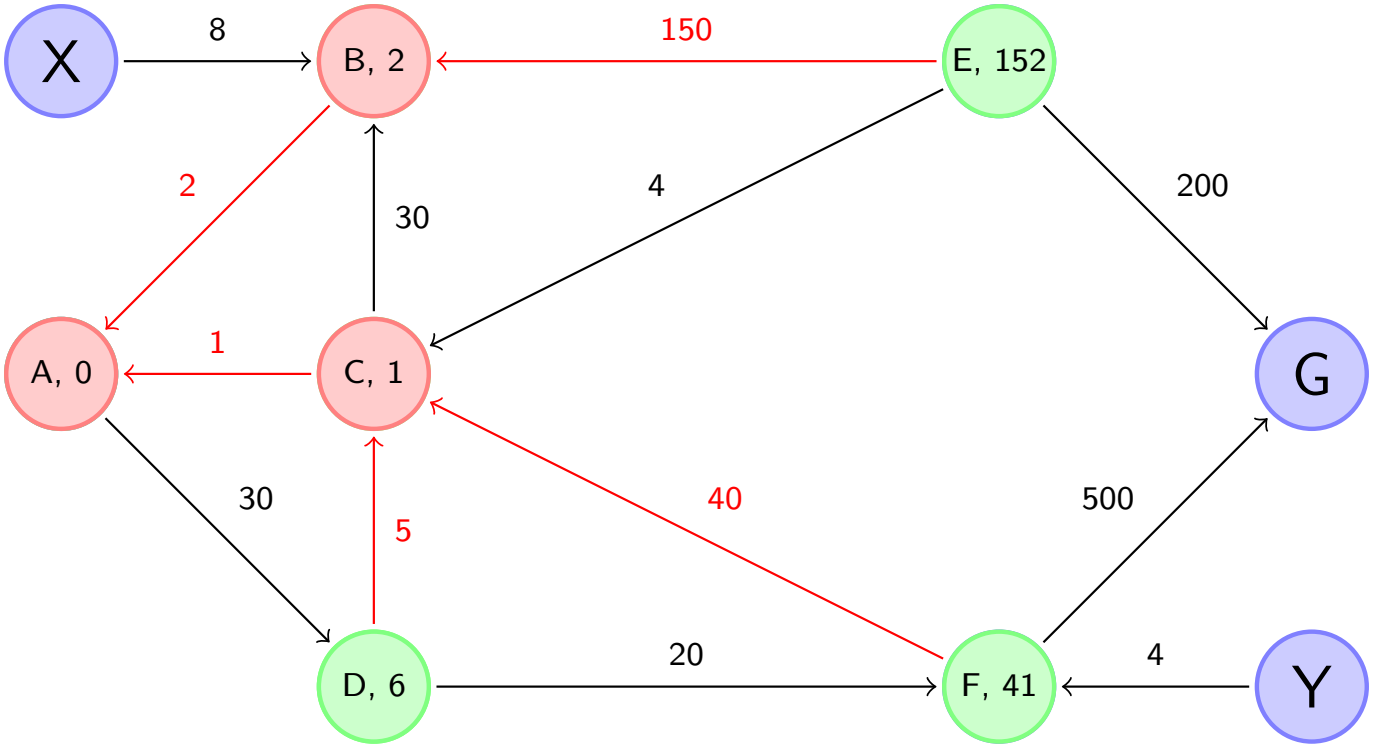
Algorithme de Dijkstra : example



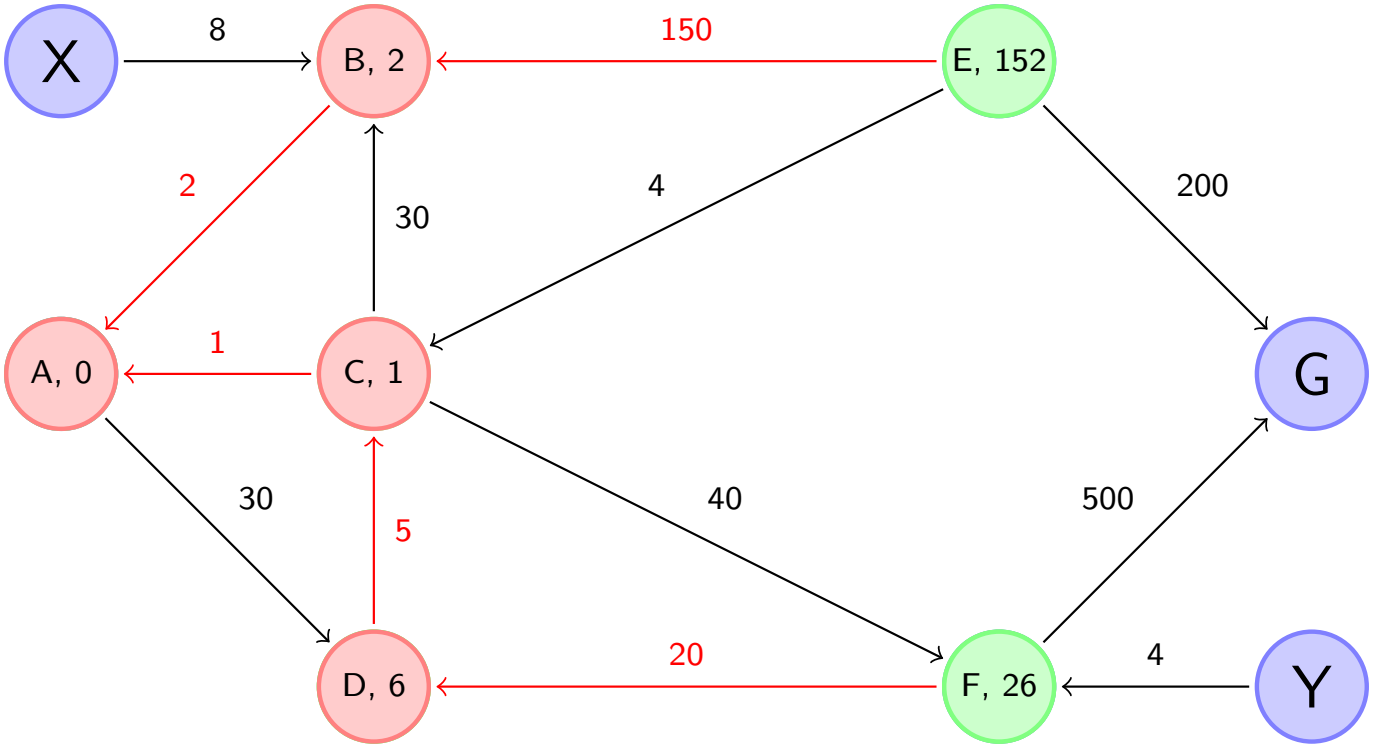
Algorithme de Dijkstra : example



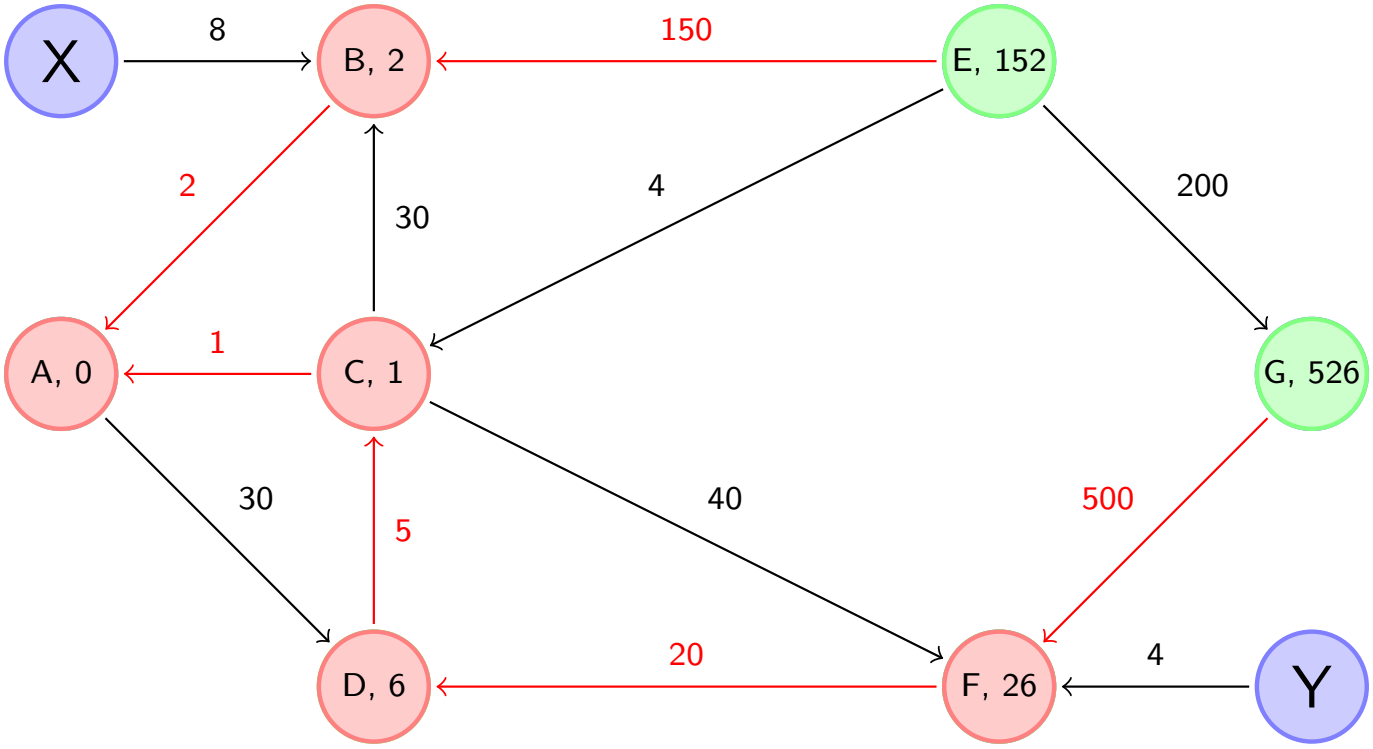
Algorithme de Dijkstra : example



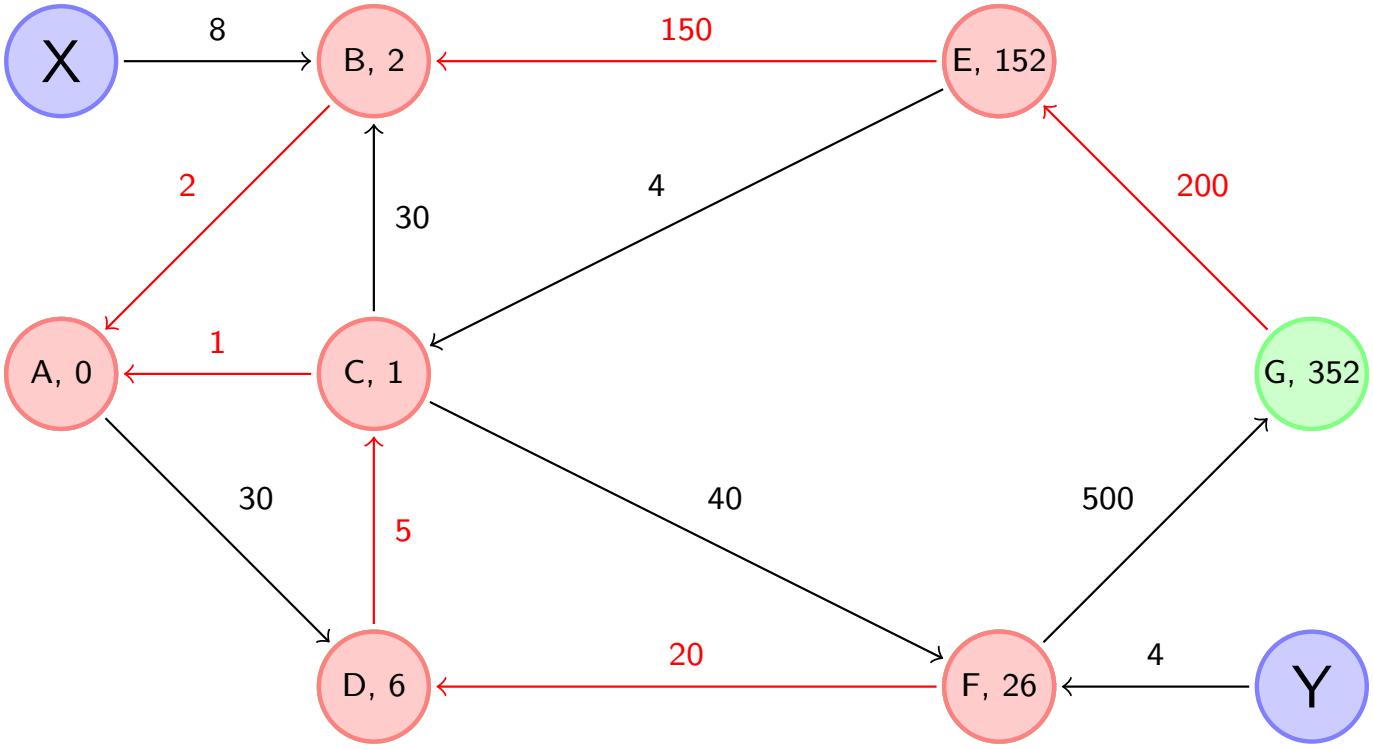
Algorithme de Dijkstra : example



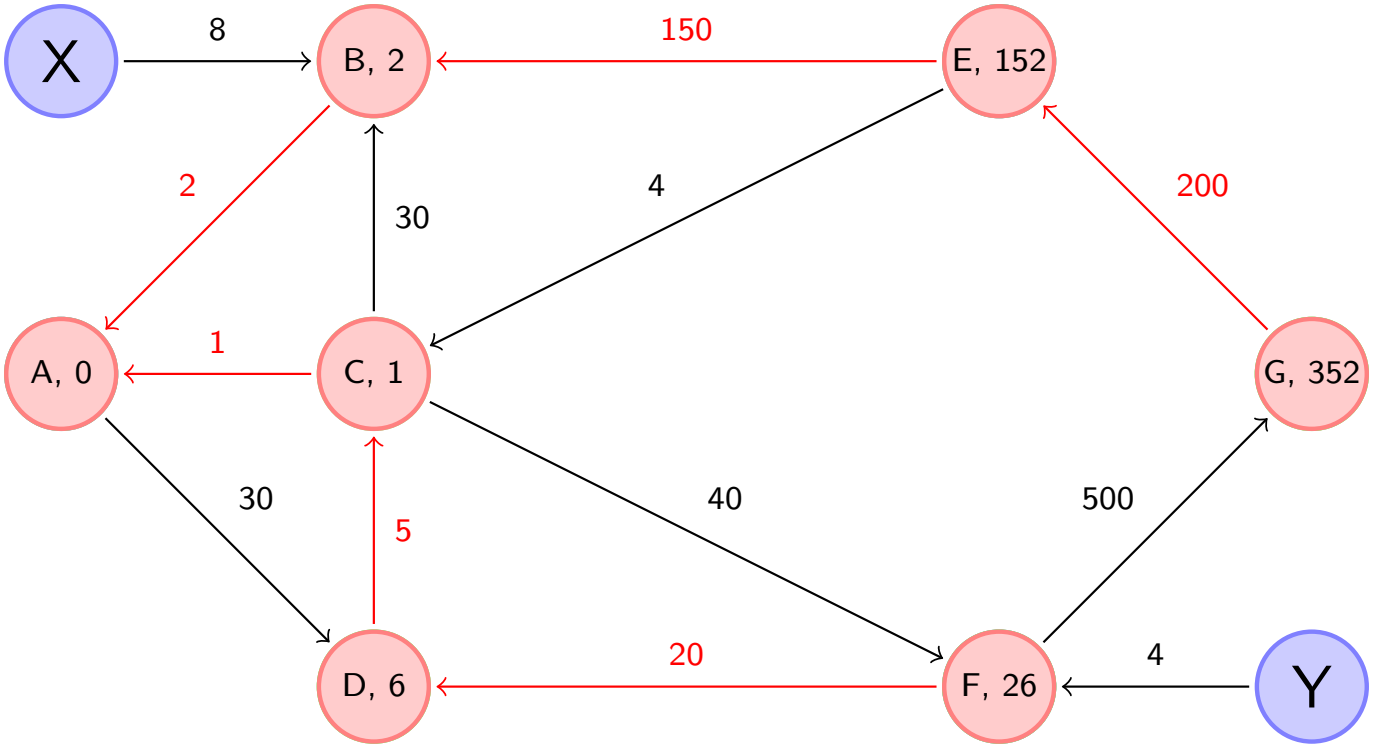
Algorithme de Dijkstra : example



Algorithme de Dijkstra : example



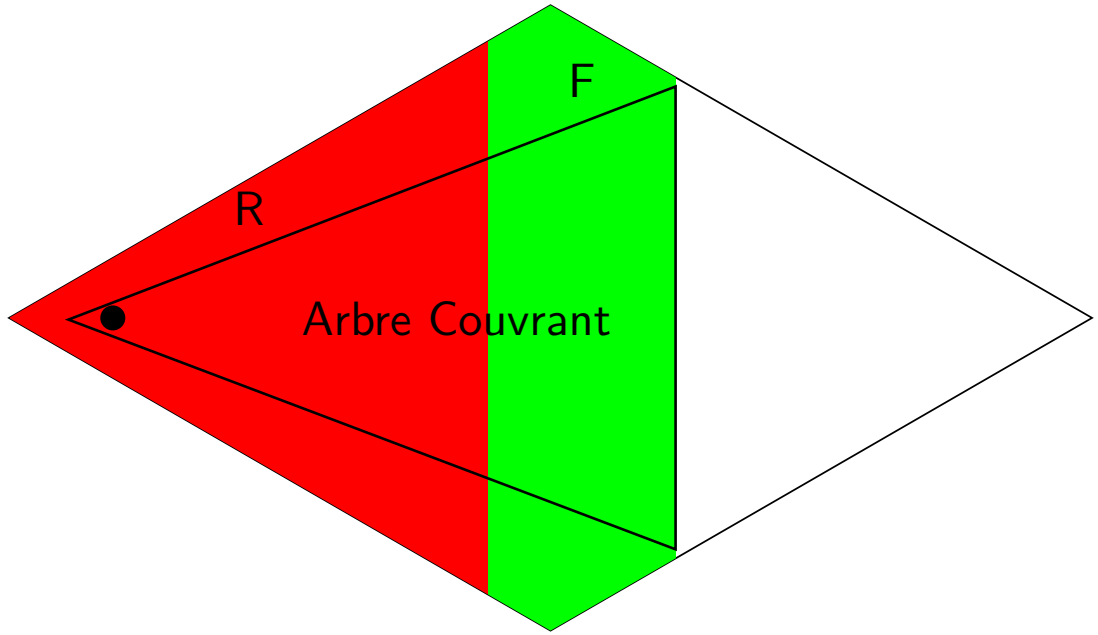
Algorithme de Dijkstra : example



Algorithme de Dijkstra : sous forme de table

	A	B	C	D	E	F	G	X	Y
0	0 (-)								
1	0 (-)	2 (A)	1 (A)	30 (A)					
2	0 (-)	2 (A)	1 (A)	6 (C)		41 (C)			
3	0 (-)	2 (A)	1 (A)	6 (C)	152 (B)	41 (C)			
4	0 (-)	2 (A)	1 (A)	6 (C)	152 (B)	26 (D)			
5	0 (-)	2 (A)	1 (A)	6 (C)	152 (B)	26 (D)	526 (E)		
6	0 (-)	2 (A)	1 (A)	6 (C)	152 (B)	26 (D)	352 (F)		
7	0 (-)	2 (A)	1 (A)	6 (C)	152 (B)	26 (D)	352 (F)		

Algorithme de Dijkstra : invariant



- tel que pour chaque état de R, on connaît un des chemins les plus courts de cet état vers l'état initial et on connaît le coût de cet état ; et

- tel que pour chaque état de F, on connaît un des chemins les plus courts – utilisant uniquement des états de $R \cup F$ – de cet état vers l'état initial et on connaît le coût de cet état

FIND-PATH

```
procedure FIND-PATH()
    (found, g) := CREATE-SPANNING-TREE()
    if found then
        g := REVERSE-PATH(g)
    end if
    return (found, g)
end procedure
```

CREATE-SPANNING-TREE

```
procedure CREATE-SPANNING-TREE()
    R := ∅
    F := {initial state}
    found := false

    while F ≠ ∅ ∧ ¬found do
        c := min(F) //O(log n)
        T := succ(c) \ R
        found := goal(c)
        F := F \ {c} //O(log n)
        F := F ∪ T
        MAJ(T, c)
    end while
    return (found, c)
end procedure
```

MAJ

procedure MAJ(H, p)
    while H $\neq \emptyset$ **do**
        s := a state of H
        H := H $\setminus \{s\}$
        if s n'a pas de parent **then**
            création de la transition s $\rightarrow$ p
            coût de s := coût de p + coût (p $\rightarrow$ s)
        else if coût de p + coût (s $\rightarrow$ p) $<$ coût de s **then**
            on efface le parent de s
            création de la transition s $\rightarrow$ p
            coût de s := coût de p + coût (p $\rightarrow$ s)
        end if
    end while
end procedure