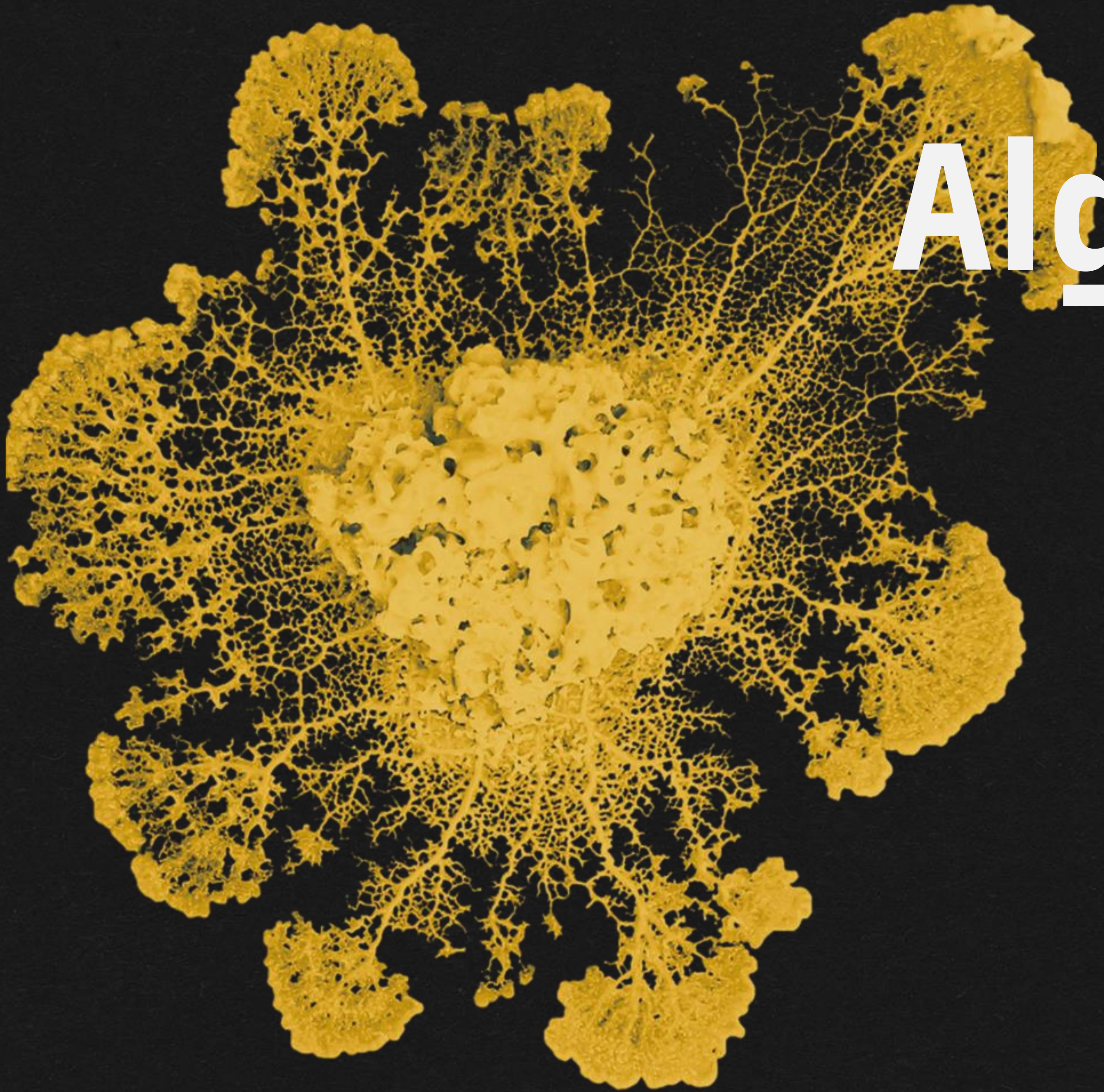
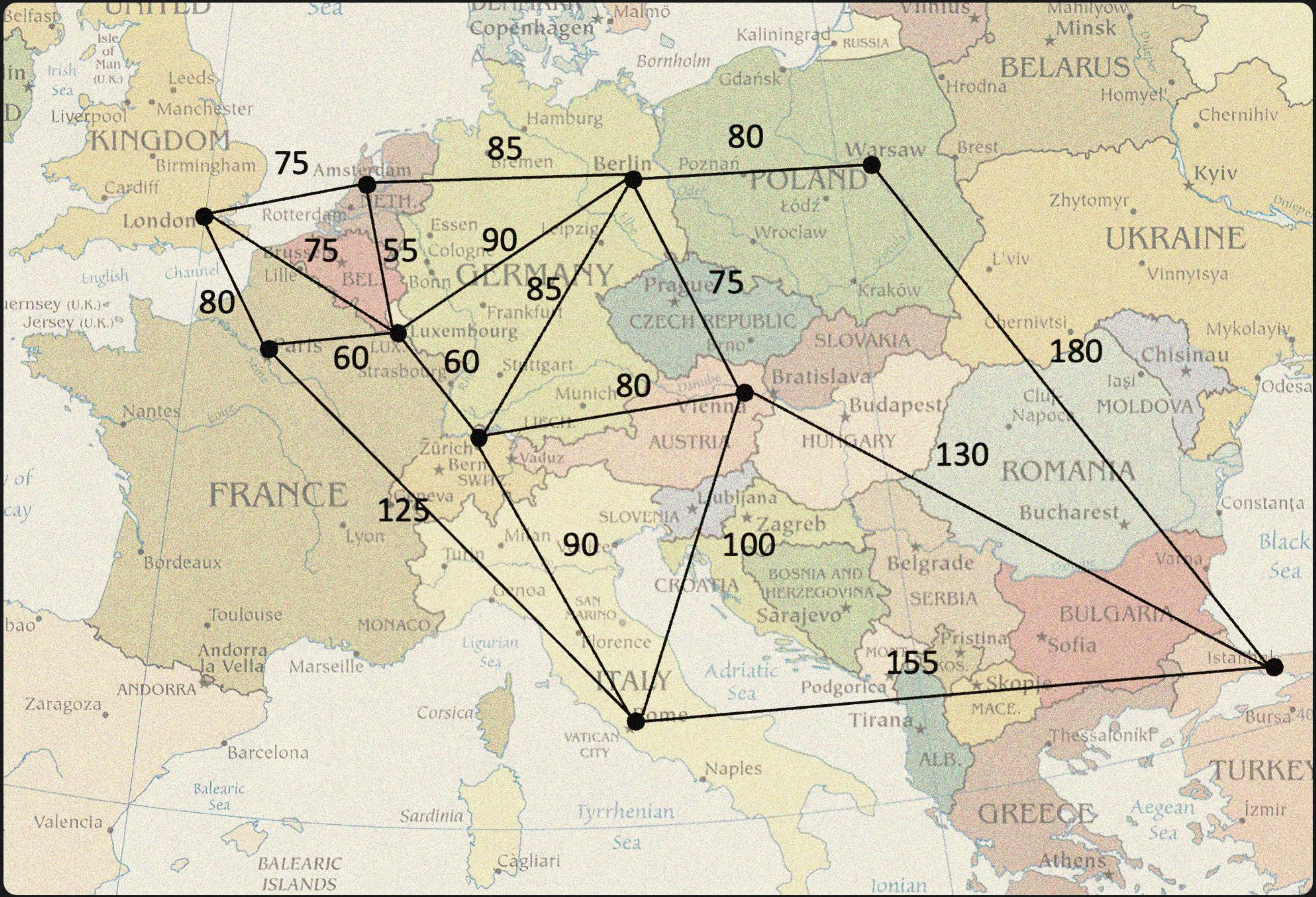


Algorithme de **Dijkstra**

L'algorithme du plus court chemin
écrit en 10 minutes à la terrasse d'un café.

Arthur Rouquan – Lycée Émile Combes – Pons, France – Février 2024


75
85
80
75
55
90
85
75
80
60
60
80
180
130
125
90
100
155
London
Amsterdam
Rotterdam
Berlin
Warsaw
Paris
Luxembourg
Zürich
Vienna
Rome
Istanbul
Hamburg
Bremen
Poznan
Gdańsk
Kaliningrad
Copenhagen
Malmö
Bornholm
Minsk
BELARUS
Hrodna
Homyel'
Brest
Chernihiv
Kyiv
Zhytomyr
UKRAINE
Vinnytsya
L'viv
POLAND
Łódź
Wroclaw
Kraków
Leipzig
Essen
Cologne
Bonn
GERMANY
Frankfurt
Stuttgart
Munich
Strasbourg
Prague
CZECH REPUBLIC
Brno
SLOVAKIA
Bratislava
Budapest
HUNGARY
AUSTRIA
Vaduz
Bern
SWITZ.
Geneva
Lyon
FRANCE
Nantes
Bordeaux
Toulouse
Andorra la Vella
ANDORRA
Marseille
MONACO
Turin
Milan
Genoa
SAN MARINO
Florence
ITALY
VATICAN CITY
Naples
Corsica
Sardinia
Cagliari
Tyrrhenian Sea
Ligurian Sea
Adriatic Sea
SLOVENIA
Ljubljana
Zagreb
CROATIA
BOSNIA AND HERZEGOVINA
Sarajevo
Belgrade
SERBIA
Podgorica
Pristina
Skopje
MACE.
Tirana
ALB.
ROMANIA
Cluj-Napoca
Bucharest
Iași
Chisinau
MOLDOVA
Odesa
Mykolayiv
Chernivtsi
Constanța
Black Sea
Varna
BULGARIA
Sofia
Thessaloniki
GREECE
Athens
Aegean Sea
İzmir
Bursa
TURKEY
Lille
BEL.
LUX.
NETH.
KINGDOM
Birmingham
Cardiff
Liverpool
Manchester
Leeds
Isle of Man (U.K.)
Irish Sea
English Channel
Guernsey (U.K.)
Jersey (U.K.)
Zaragoza
Barcelona
Valencia
Balearic Sea
BALEARIC ISLANDS
Ionian

Quelques définitions

Soit un graphe **non-orienté**.

Une **chaîne** est une suite d'arêtes consécutives.

Une chaîne est **simple** si elle ne repasse pas par une même arête.

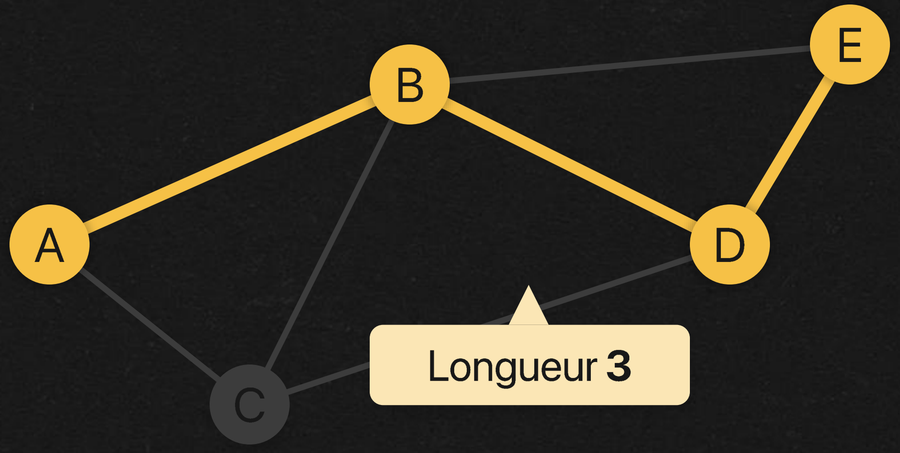
La **longueur** d'une chaîne est pour un :

- graphe **non-pondéré**, le **nombre d'arêtes** de la chaîne
- graphe **pondéré**, la **somme des poids** de ses arêtes

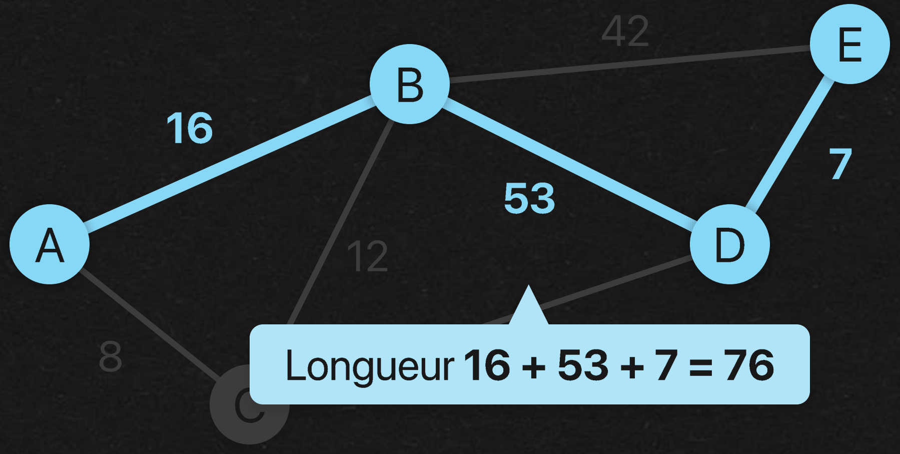
Pour un graphe **orienté** :

- Remplacer le terme « chaîne » par « **chemin** »
- Remplacer le terme « arête » par « **arc** »

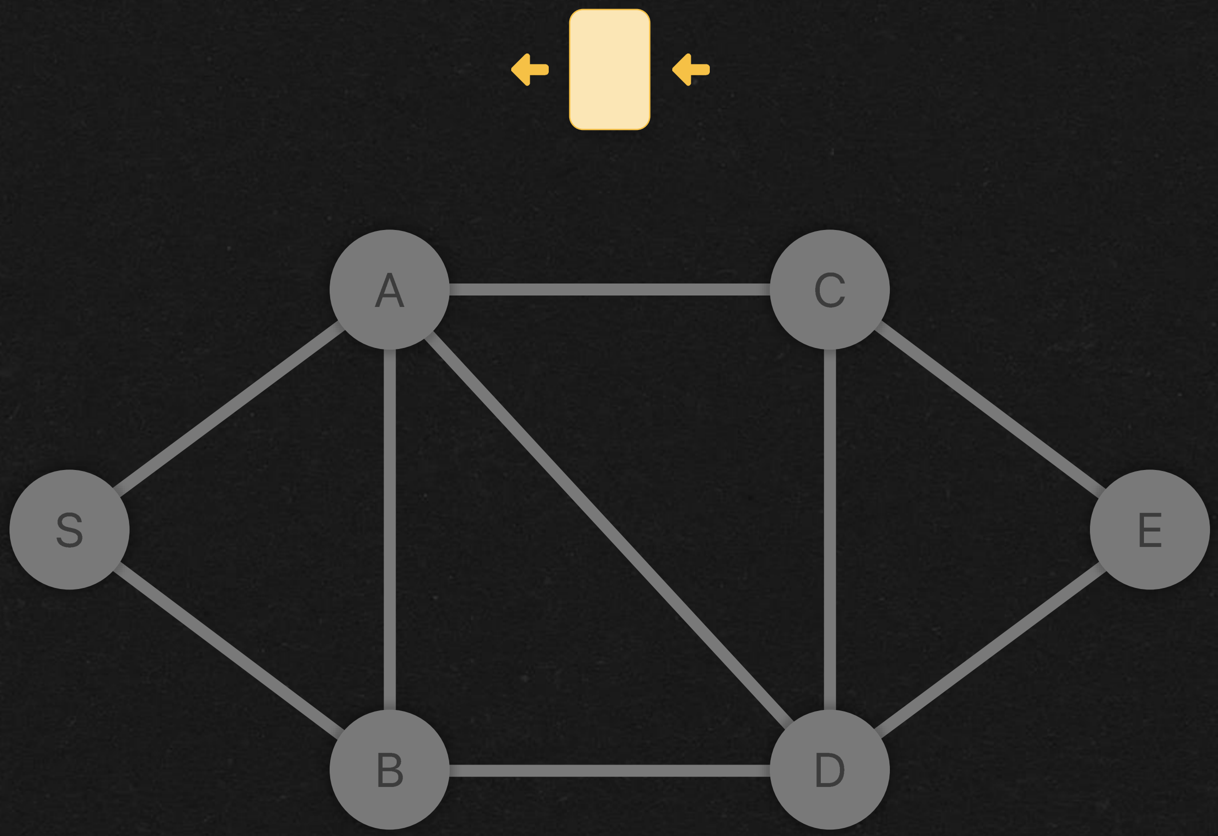
Le problème du **plus court chemin** consiste à déterminer le chemin / la chaîne de plus petite longueur qui relie deux sommets.



- **A – B – D – E** est une **chaîne simple**
- **A – B – D – B – E** est une **chaîne non-simple**



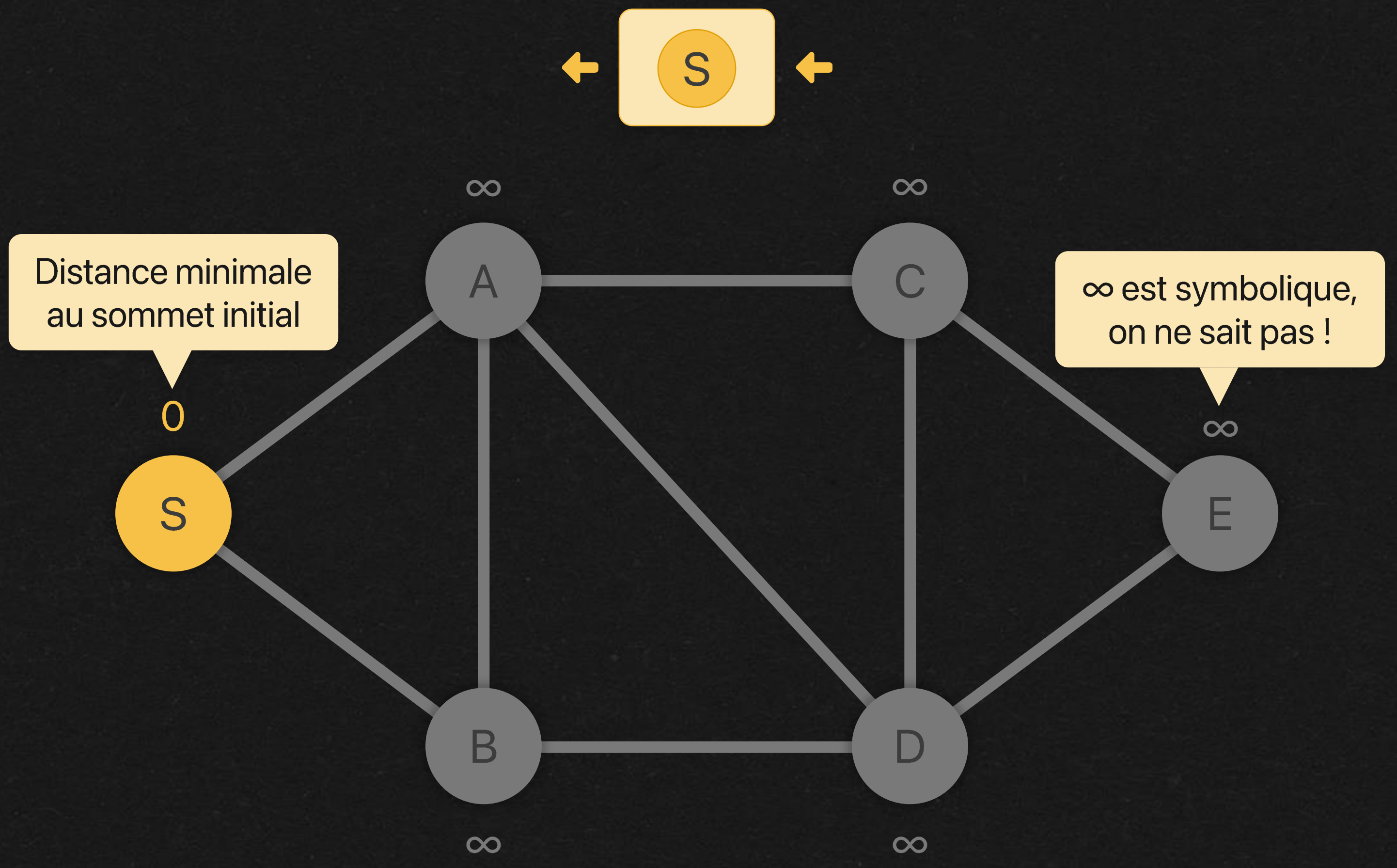
File des sommets à visiter



Sommet non-rencontré
Sommet à visiter
Sommet en cours de visite
Sommet visité

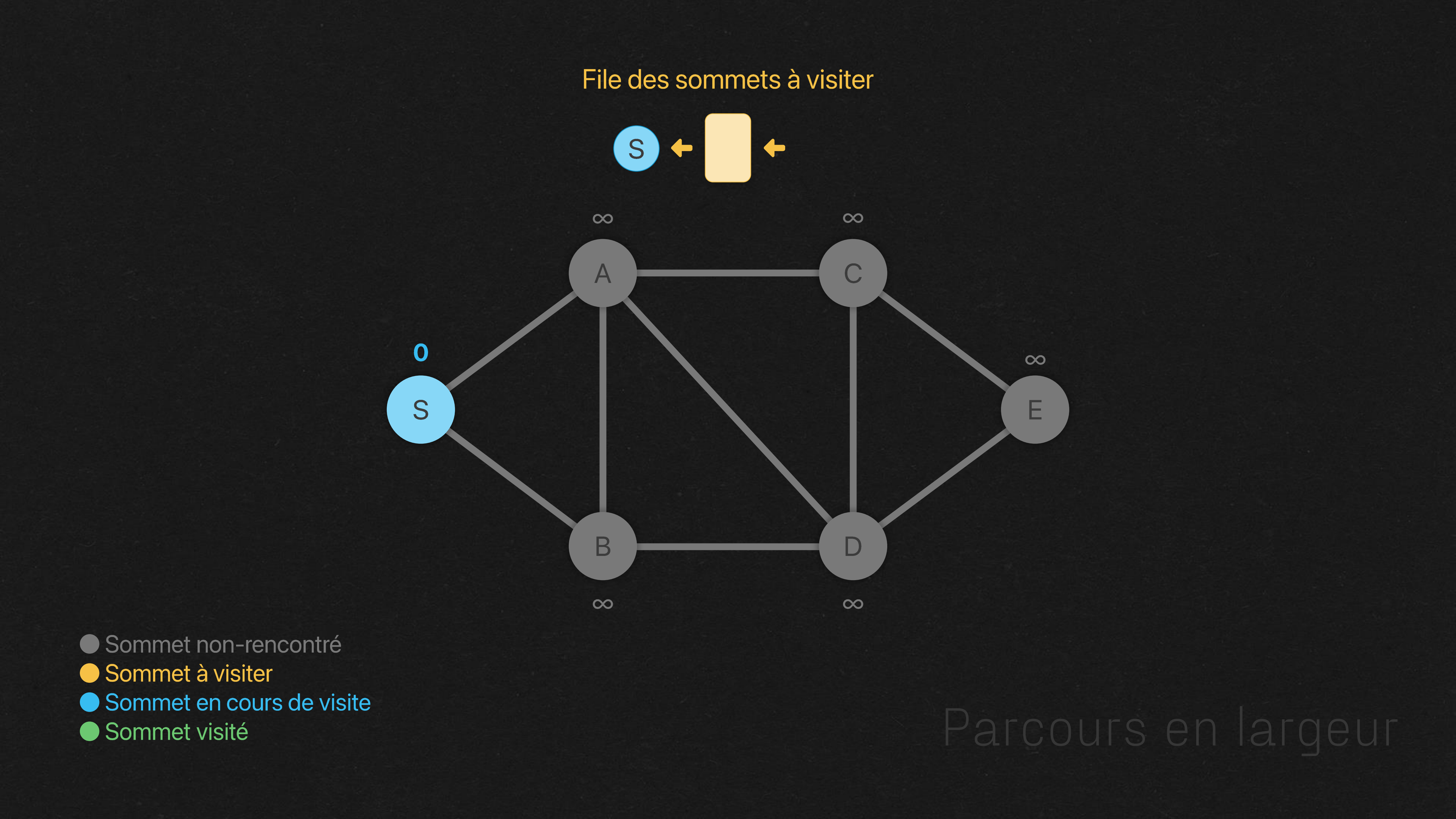
Parcours en largeur

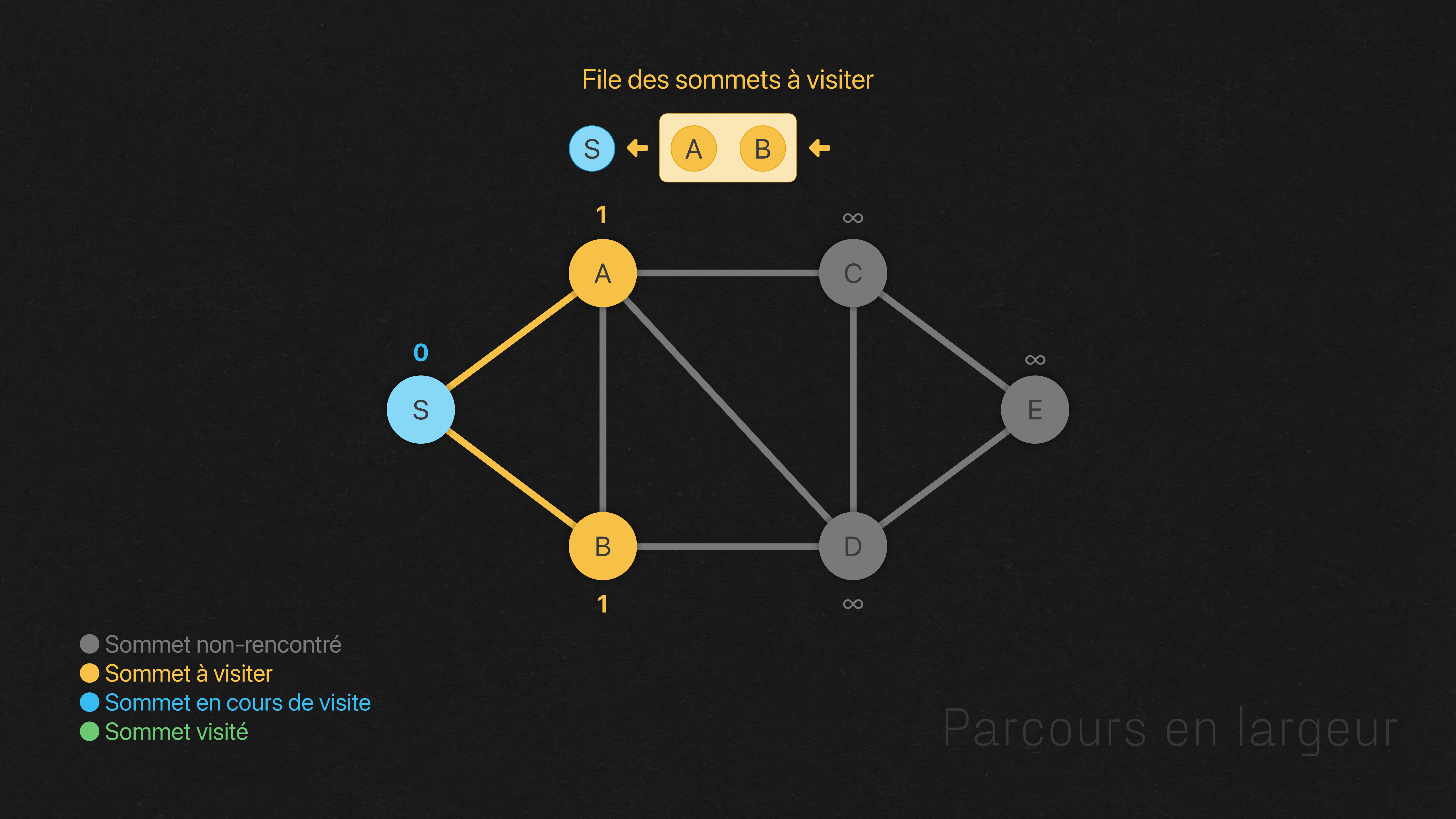
File des sommets à visiter



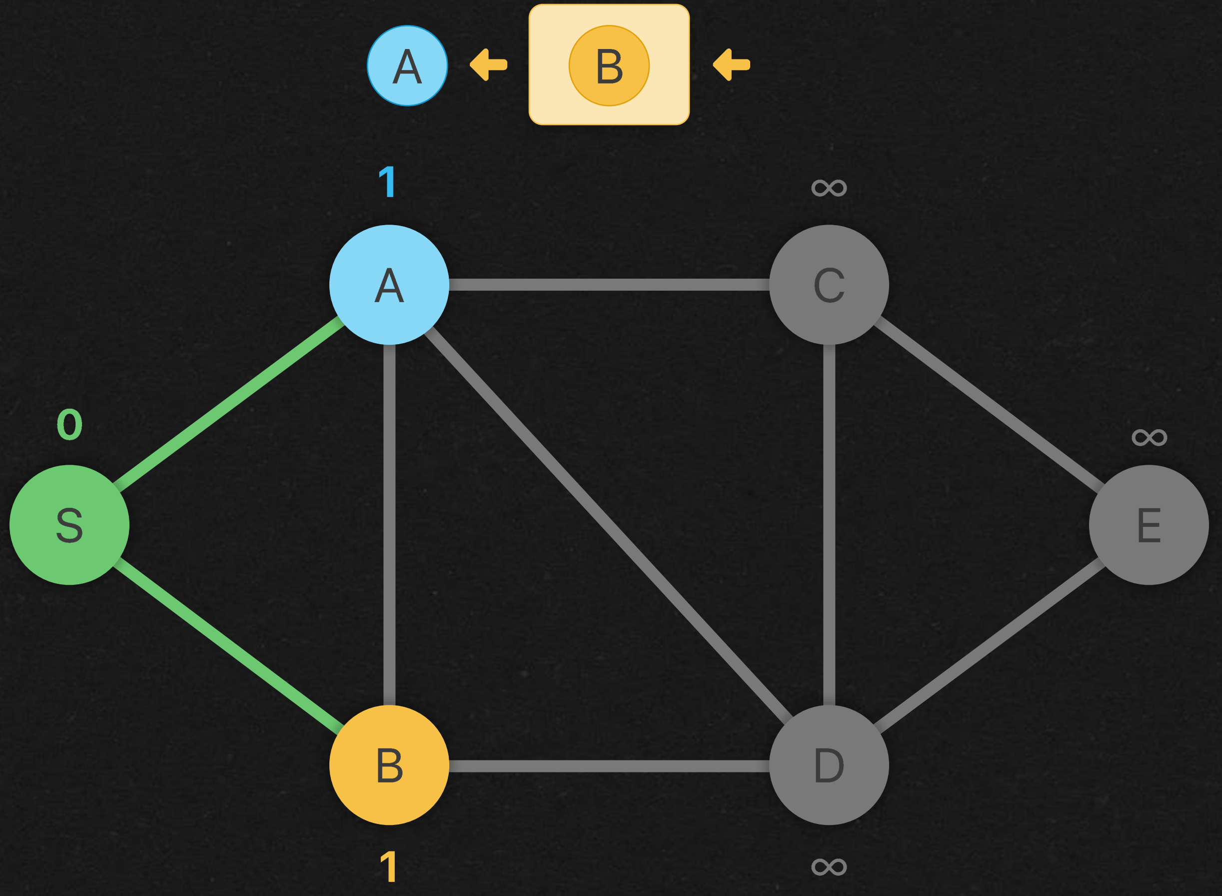
- Sommet non-rencontré
- Sommet à visiter
- Sommet en cours de visite
- Sommet visité

Parcours en largeur


File des sommets à visiter
S
∞
A
∞
C
0
S
∞
E
B
D
∞
∞
Sommet non-rencontré
Sommet à visiter
Sommet en cours de visite
Sommet visité
Parcours en largeur

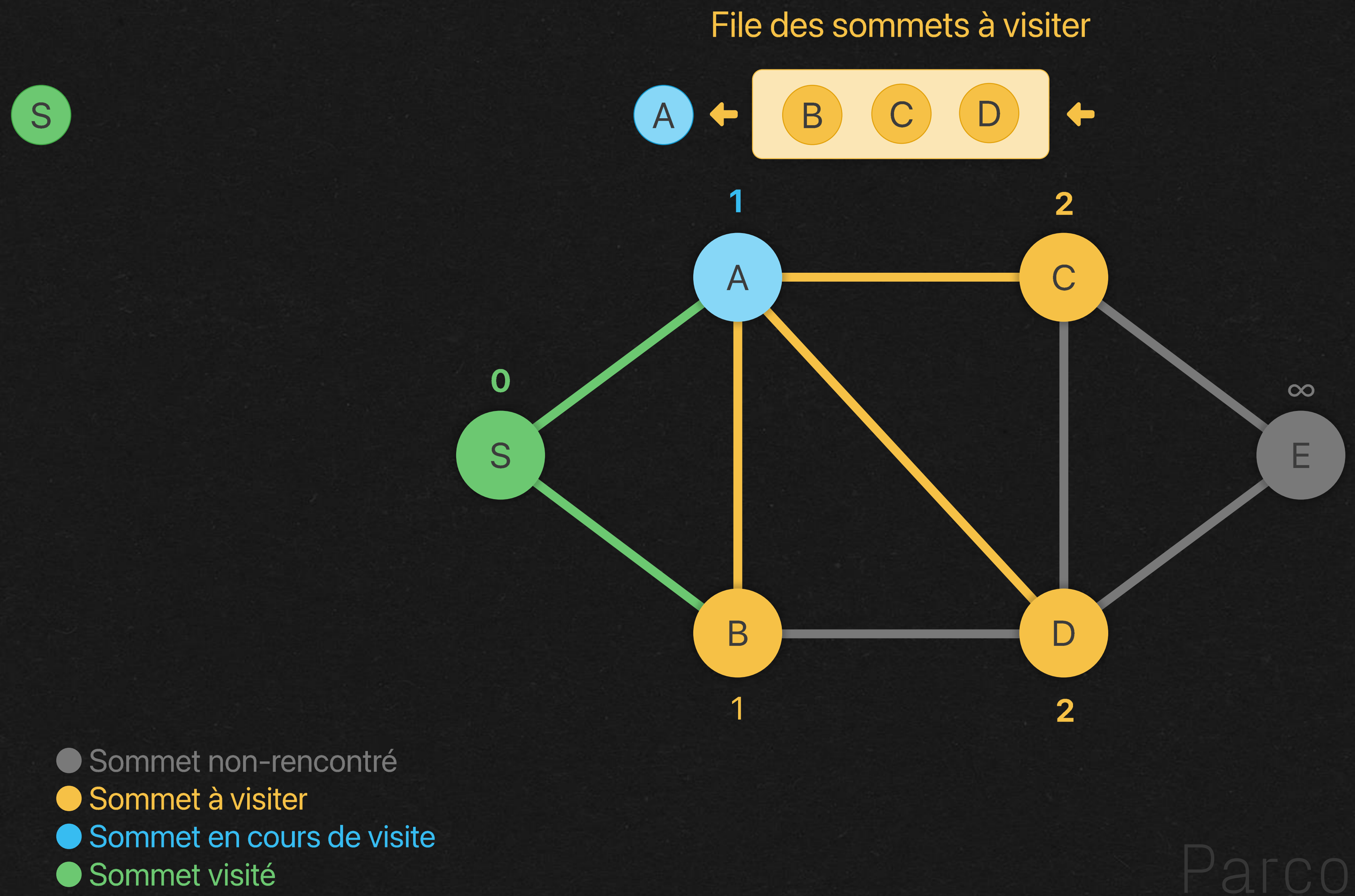

File des sommets à visiter
S
A
B
1
∞
0
∞
A
C
S
E
B
D
1
∞
Sommet non-rencontré
Sommet à visiter
Sommet en cours de visite
Sommet visité
Parcours en largeur

File des sommets à visiter



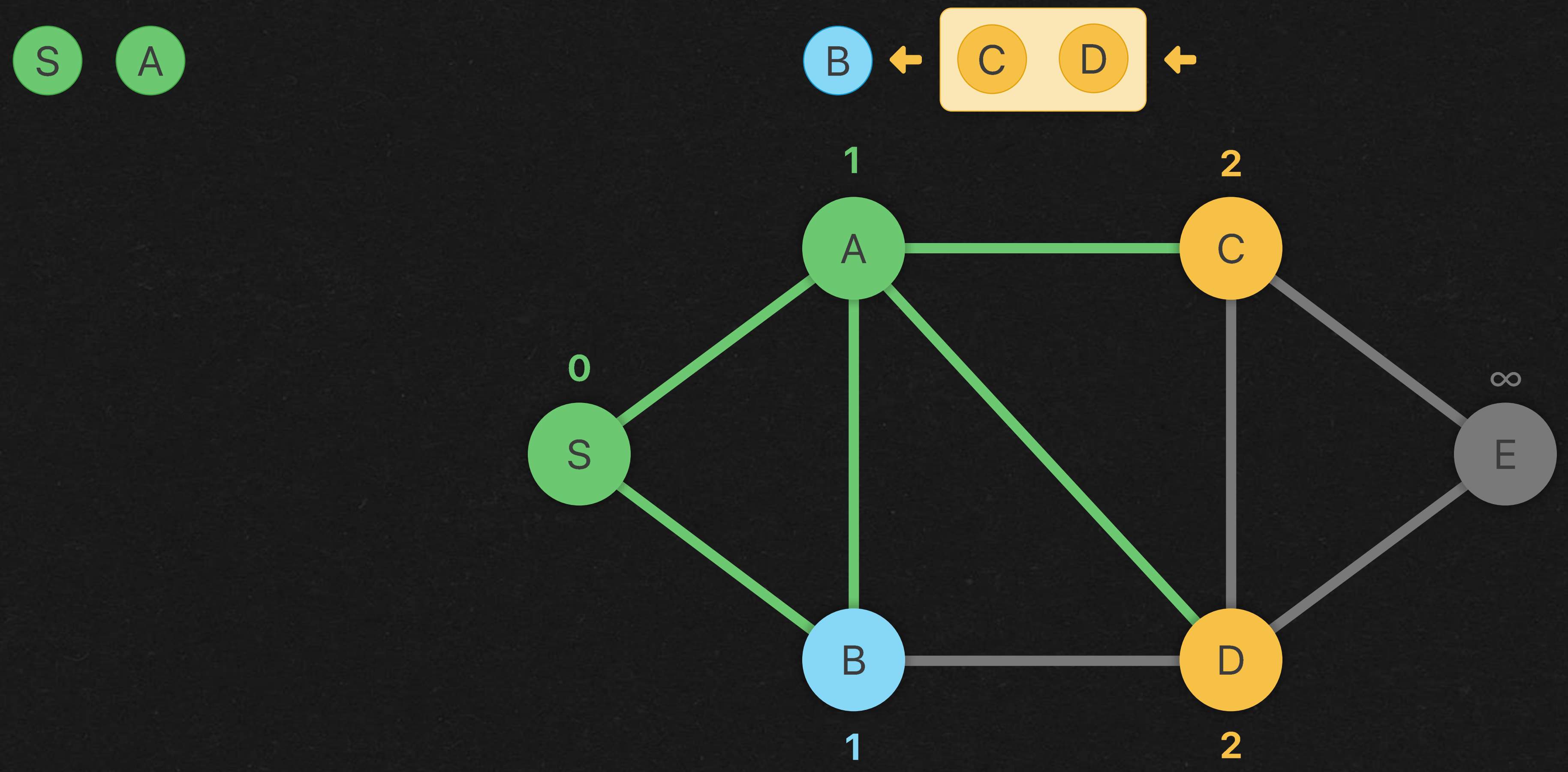
- Sommet non-rencontré
- Sommet à visiter
- Sommet en cours de visite
- Sommet visité

Parcours en largeur



Parcours en largeur

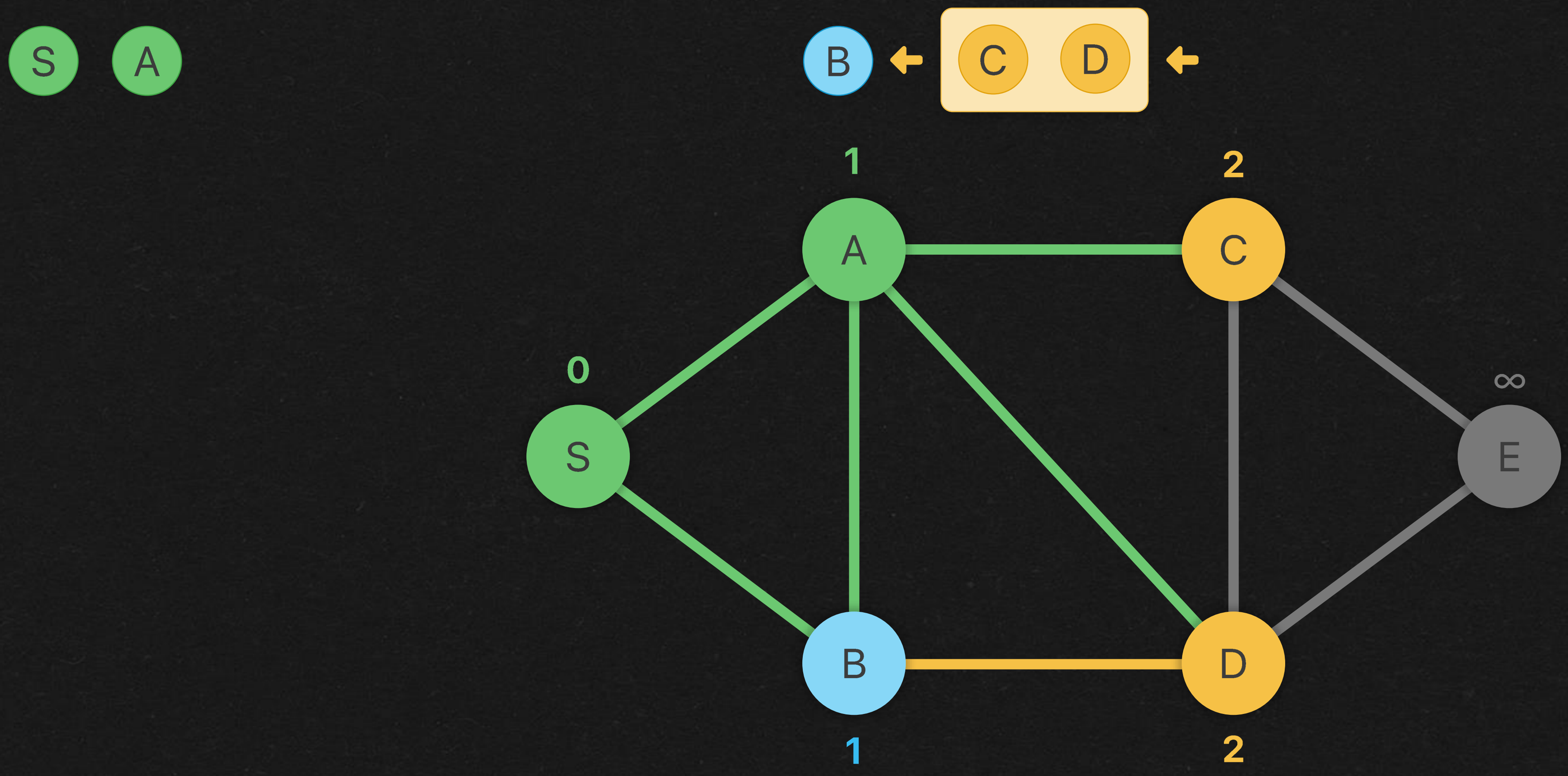
File des sommets à visiter



- Sommet non-rencontré
- Sommet à visiter
- Sommet en cours de visite
- Sommet visité

Parcours en largeur

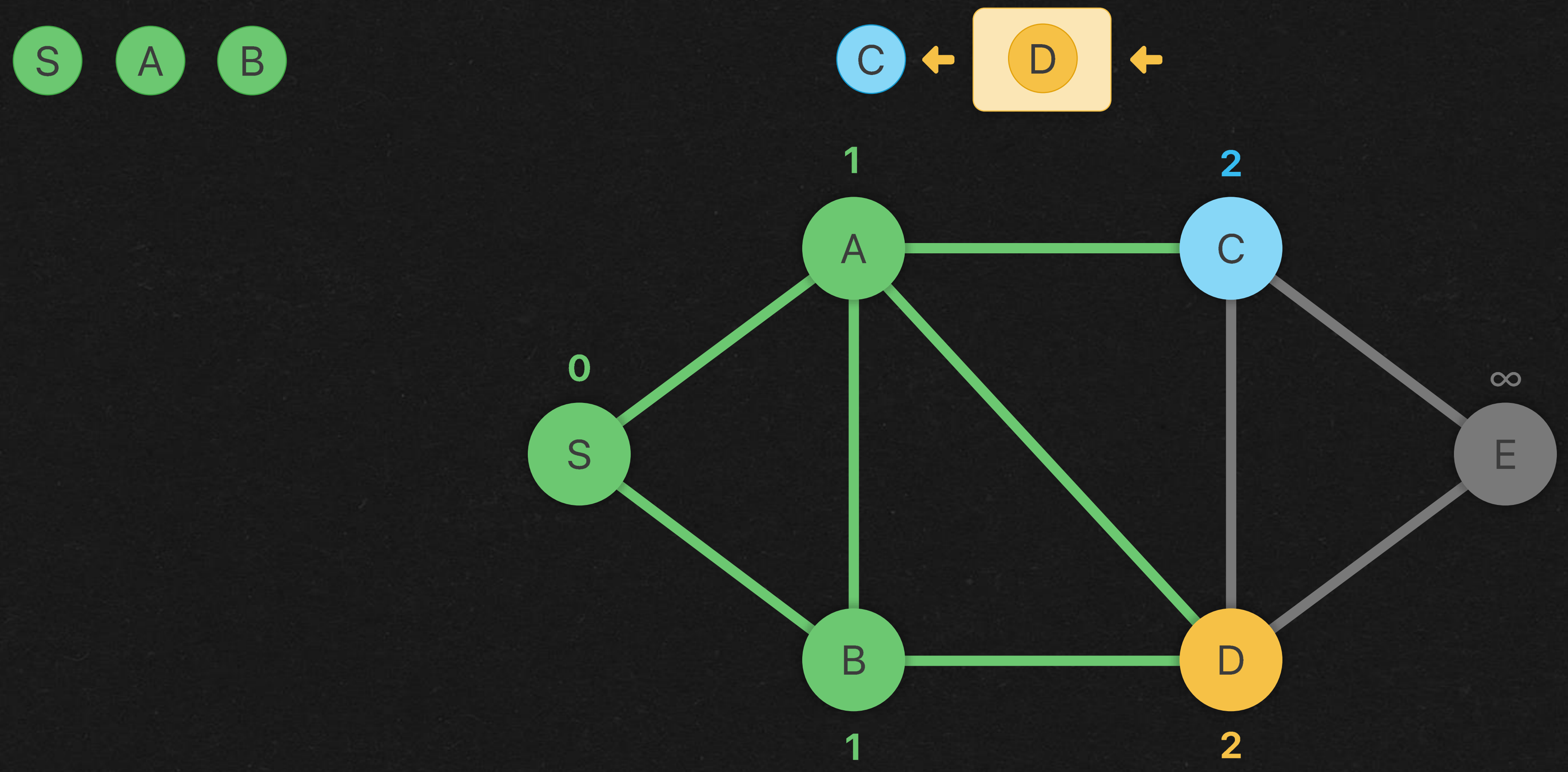
File des sommets à visiter



- Sommet non-rencontré
- Sommet à visiter
- Sommet en cours de visite
- Sommet visité

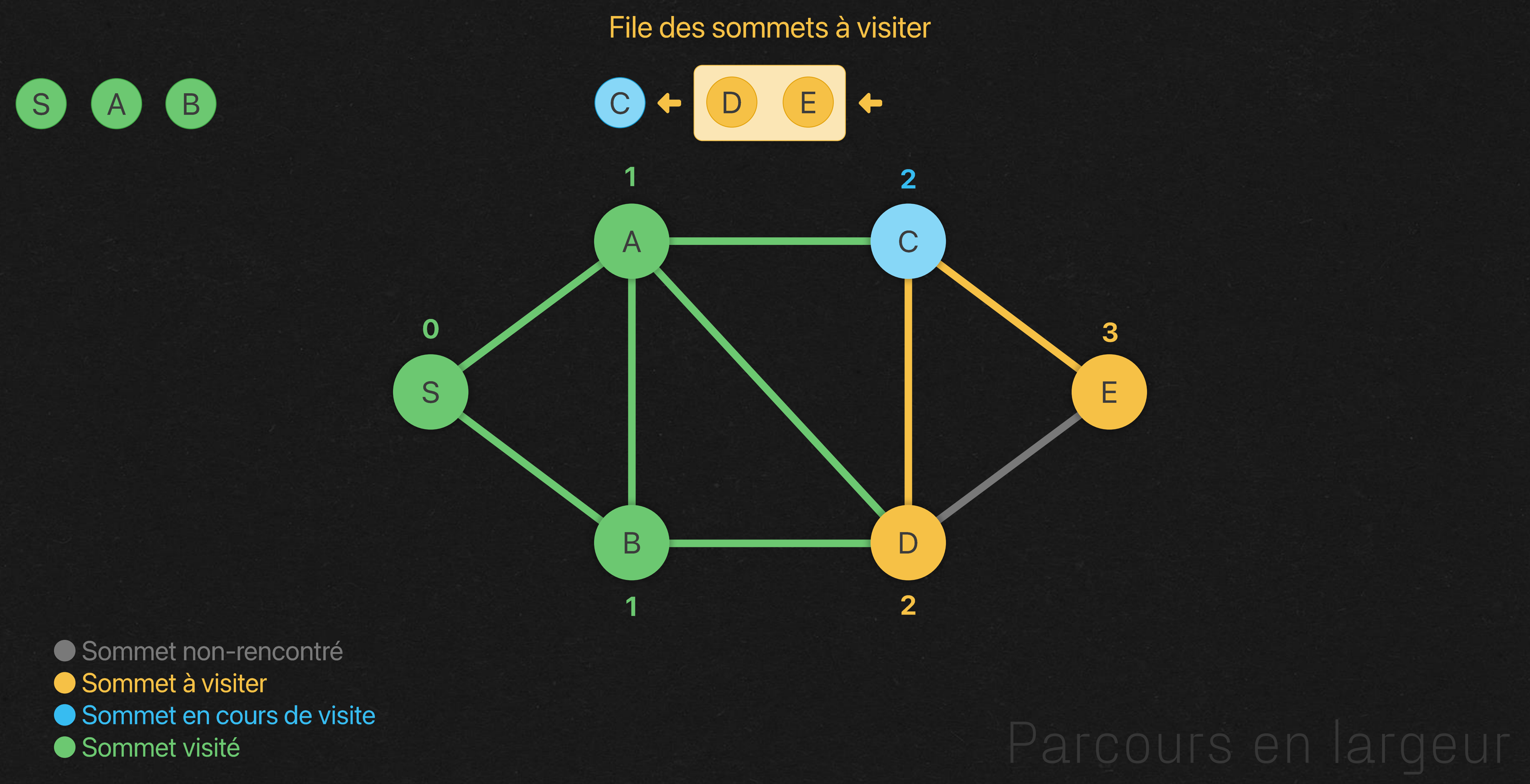
Parcours en largeur

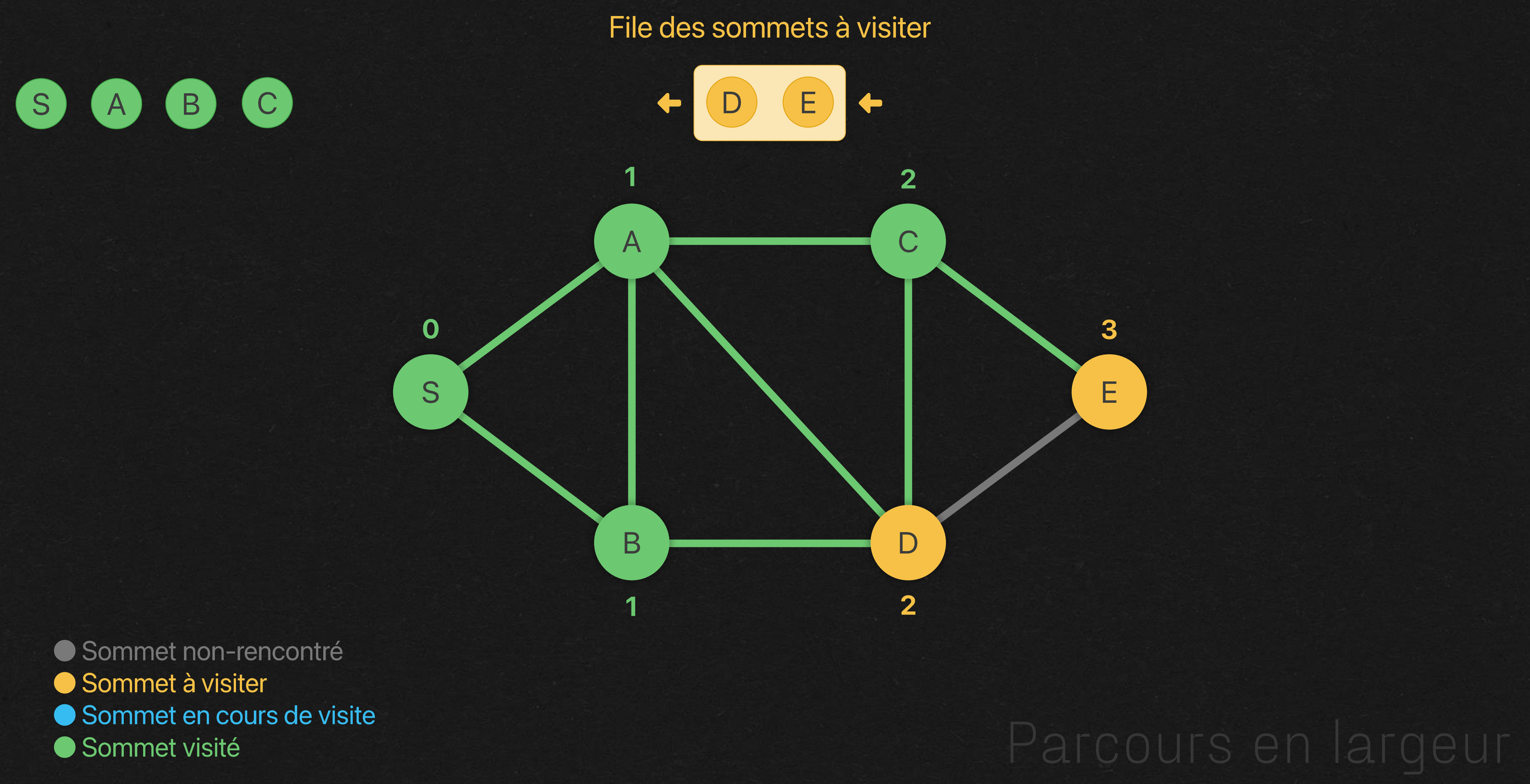
File des sommets à visiter

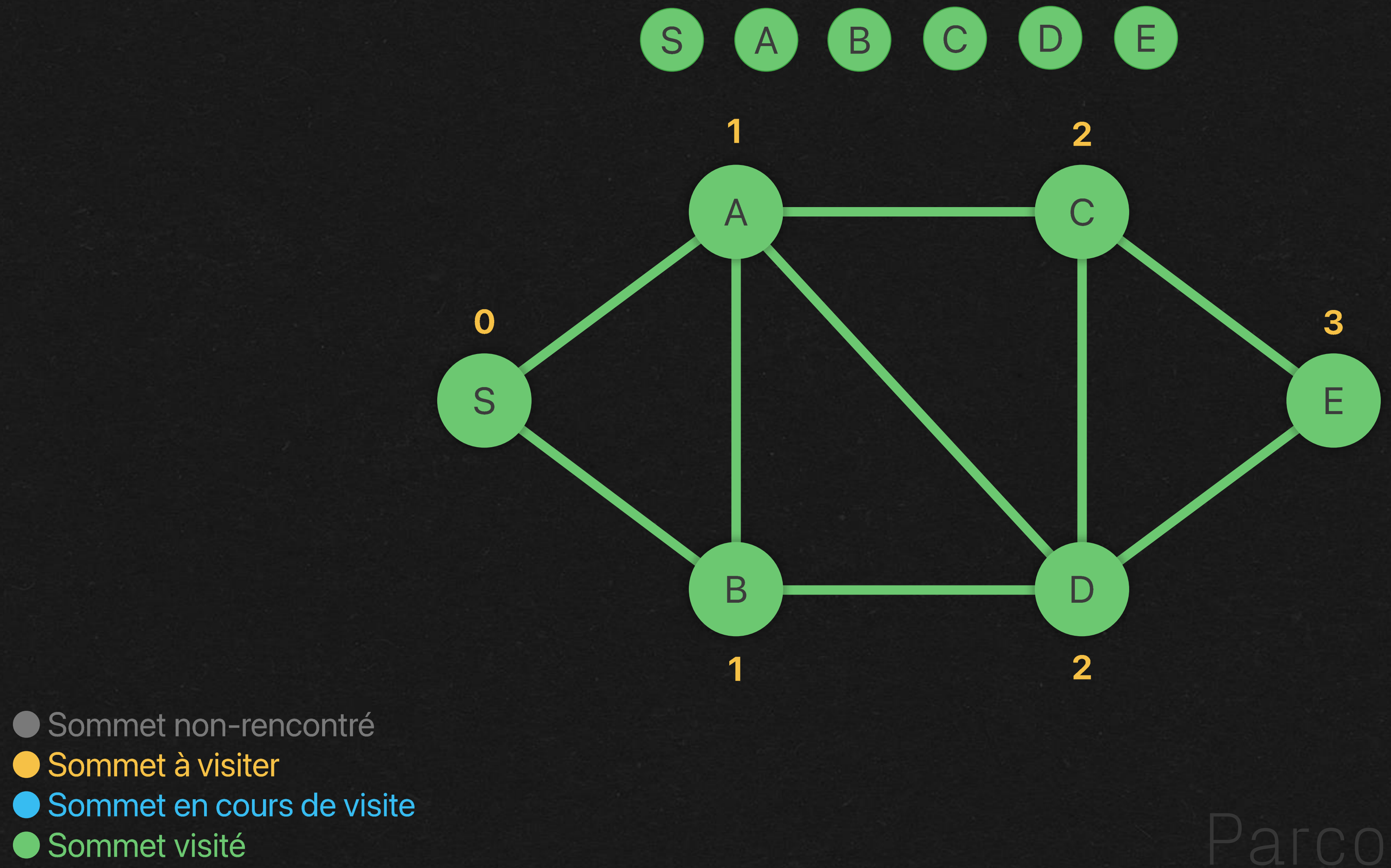


- Sommet non-rencontré
- Sommet à visiter
- Sommet en cours de visite
- Sommet visité

Parcours en largeur


File des sommets à visiter
S
A
B
C
D
E
1
2
0
3
A
C
S
E
B
D
1
2
Sommet non-rencontré
Sommet à visiter
Sommet en cours de visite
Sommet visité
Parcours en largeur


S A B C
File des sommets à visiter
D E
1 2
A C
0 3
S E
B D
1 2
Sommet non-rencontré
Sommet à visiter
Sommet en cours de visite
Sommet visité
Parcours en largeur



Parcours en largeur

Rappels sur le parcours en largeur

Pseudocode du parcours en largeur ci-contre.

Les sommets dans la file sont toujours ajoutés par ordre croissant de leur distance au sommet initial.

À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial.

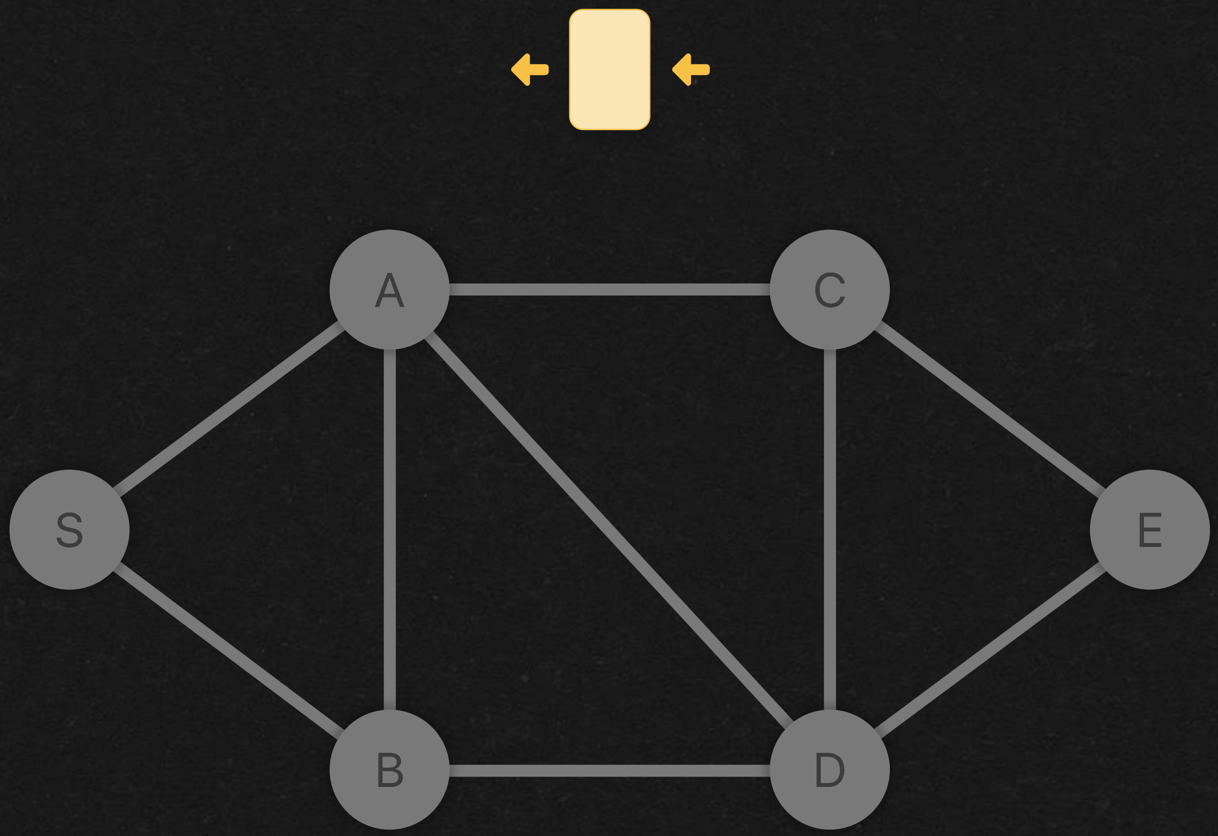
Le problème du **plus court chemin** consiste à déterminer le chemin / la chaîne de plus petite longueur qui relie deux sommets.

Presque résolu pour cas d'un graphe **non-pondéré** ! Avec le parcours en largeur, on a les distances minimales, mais pas les chemins… comment faire ?

Il suffit de se souvenir d'où l'on vient ! Pour chaque sommet, on garde en mémoire son **prédécesseur,** i.e. le sommet par lequel on est arrivé à celui-ci.

```
Entrées: Un graphe G et un sommet initial s
Sortie: La distance min entre s et les autres sommets

distance[u] = ∞ pour tout sommet u de G
distance[s] = 0

initialiser une file F vide
enfiler s dans F
marquer s

tant que F n'est pas vide
    défiler F dans u
    pour chaque voisin v non-marqué de u
        enfiler v dans F
        marquer v
        distance[v] = distance[u] + 1

retourner distance
```

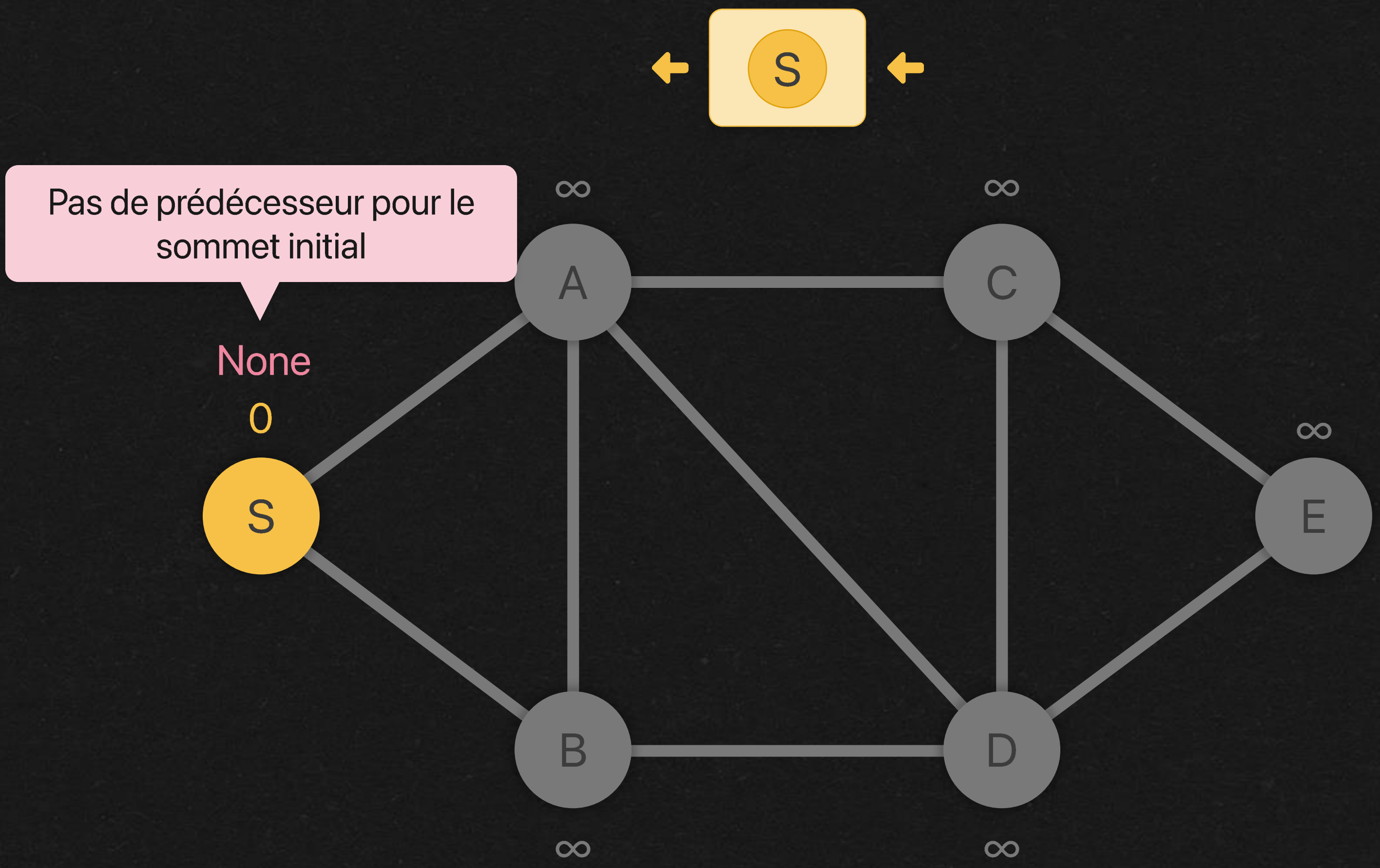
File des sommets à visiter



Sommet non-rencontré
Sommet à visiter
Sommet en cours de visite
Sommet visité

Parcours en largeur

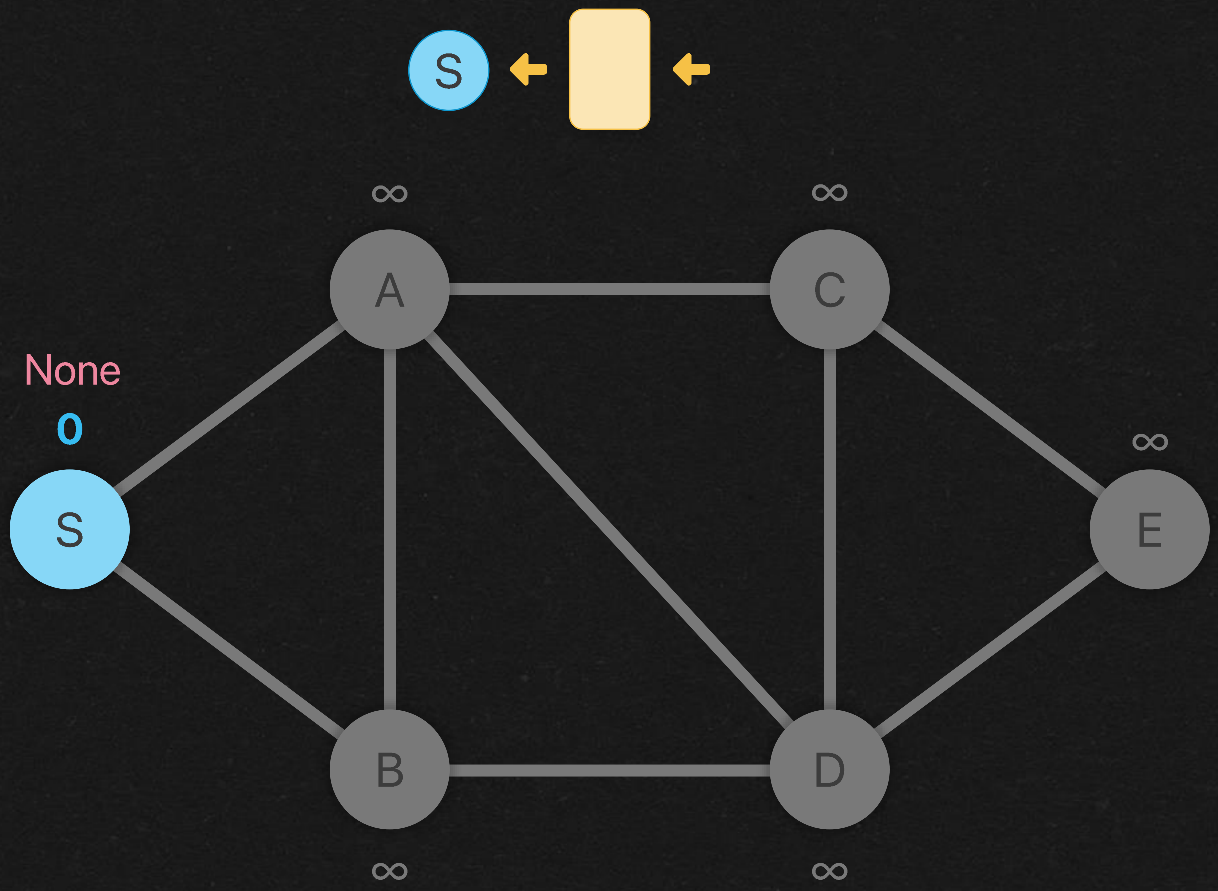
File des sommets à visiter



Sommet non-rencontré
Sommet à visiter
Sommet en cours de visite
Sommet visité

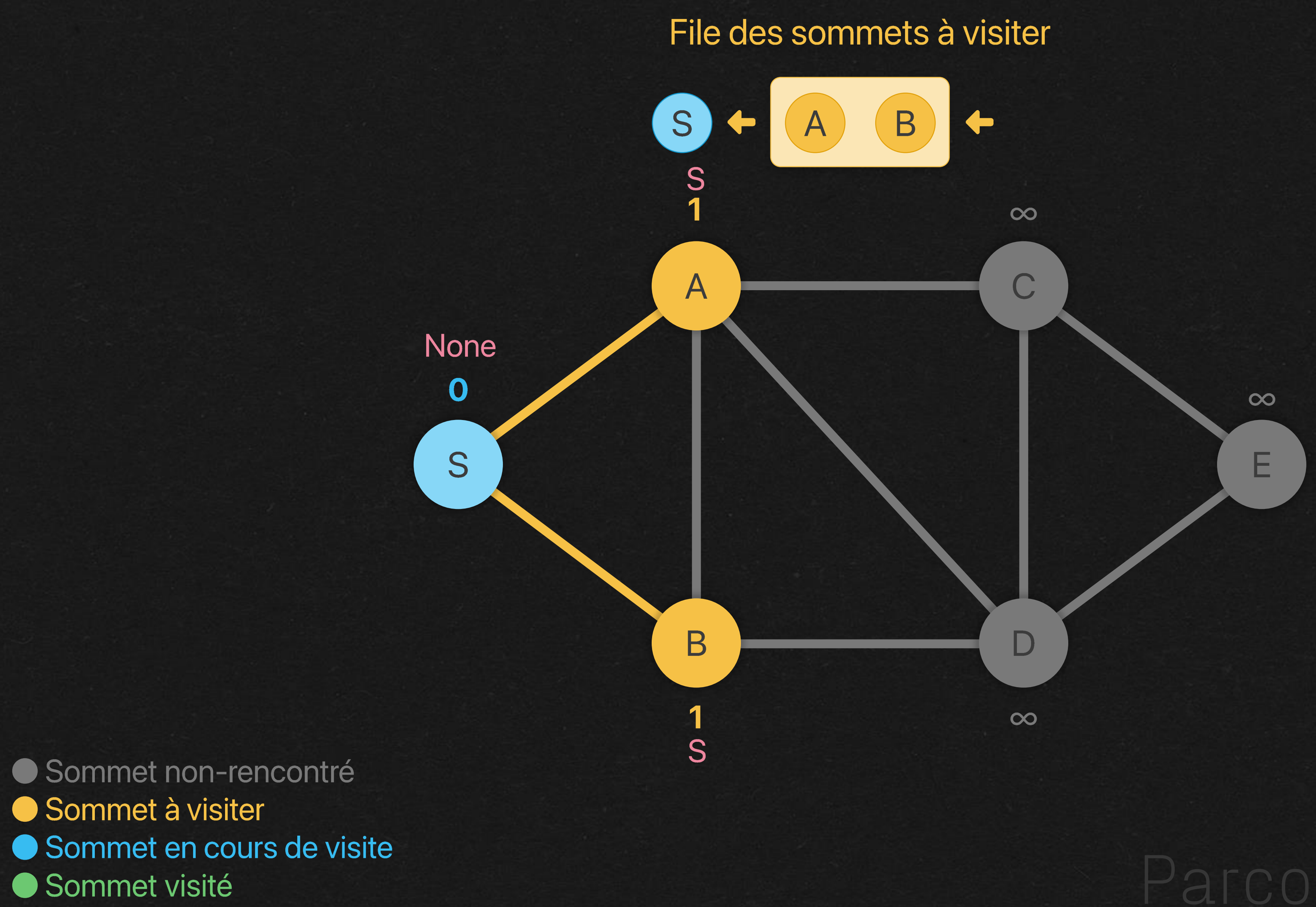
Parcours en largeur

File des sommets à visiter



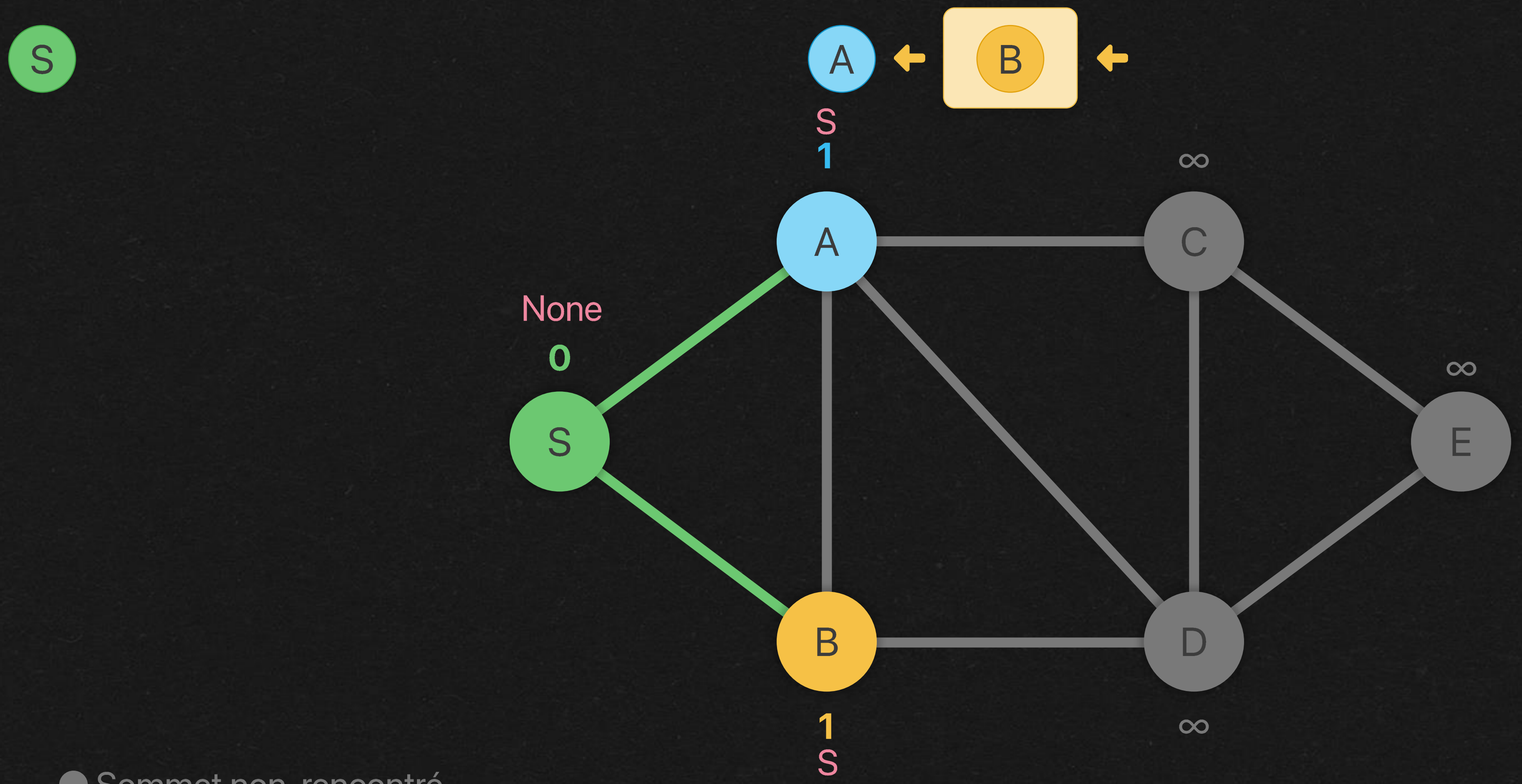
- Sommet non-rencontré
- Sommet à visiter
- Sommet en cours de visite
- Sommet visité

Parcours en largeur



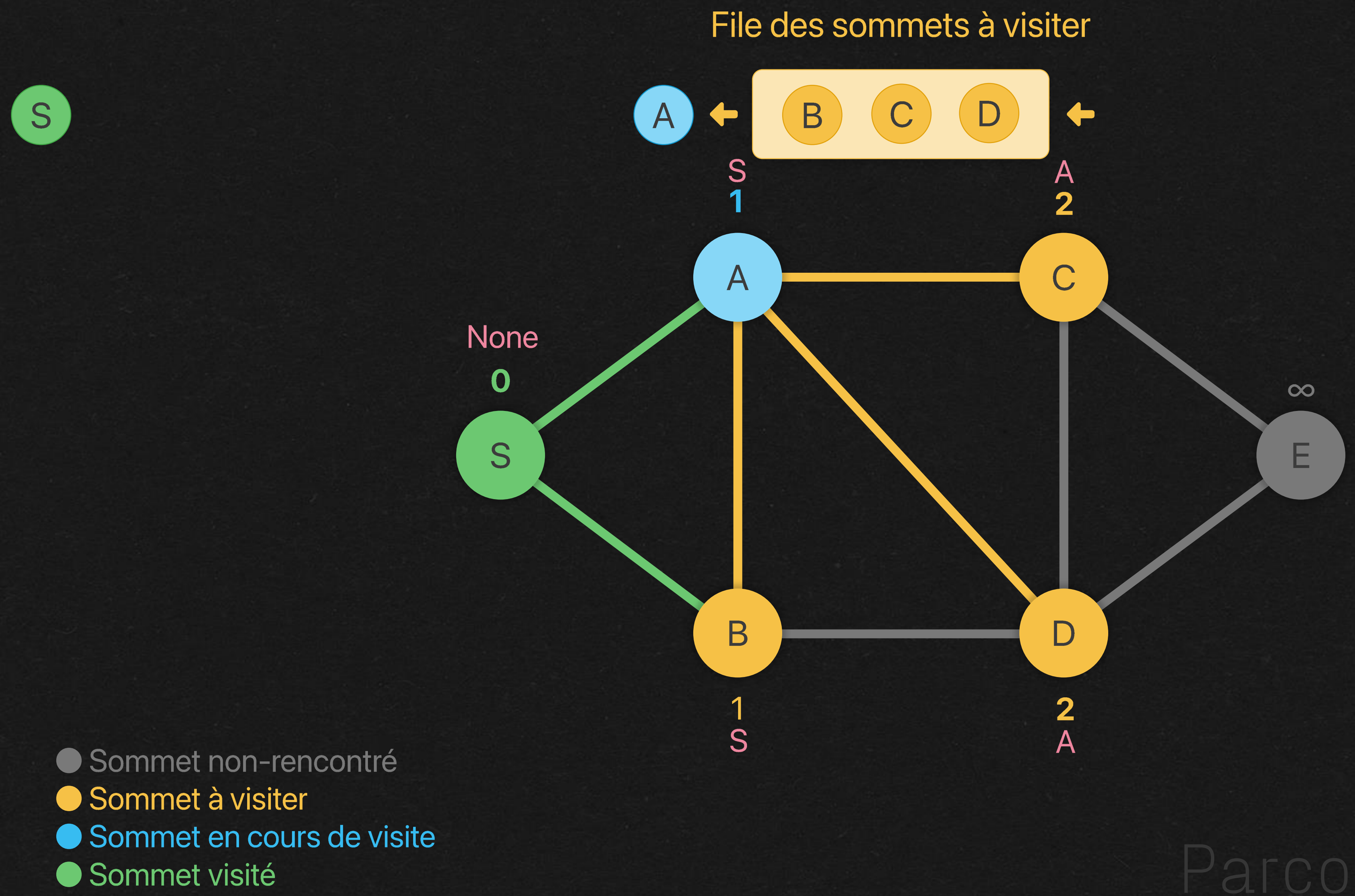
Parcours en largeur

File des sommets à visiter

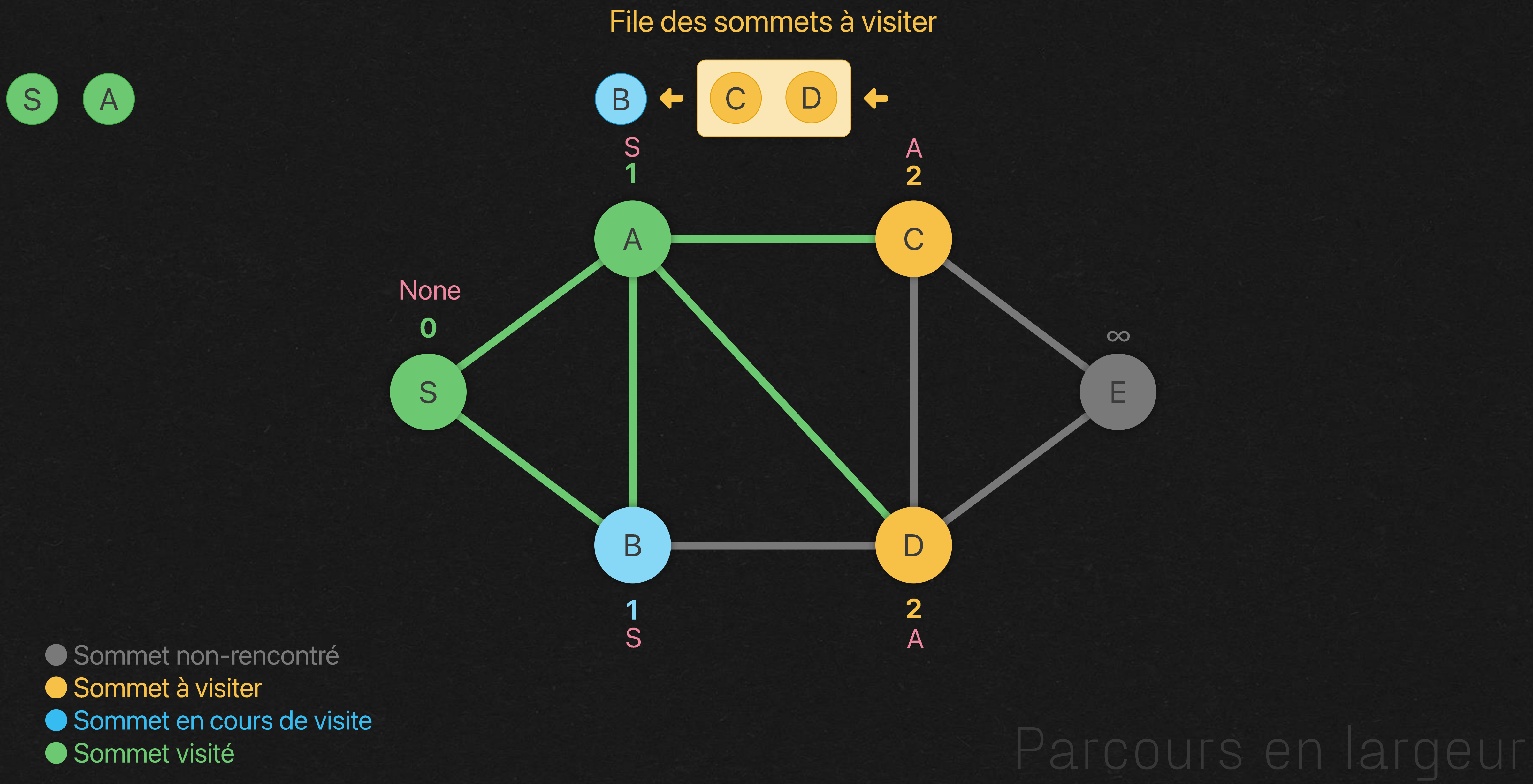


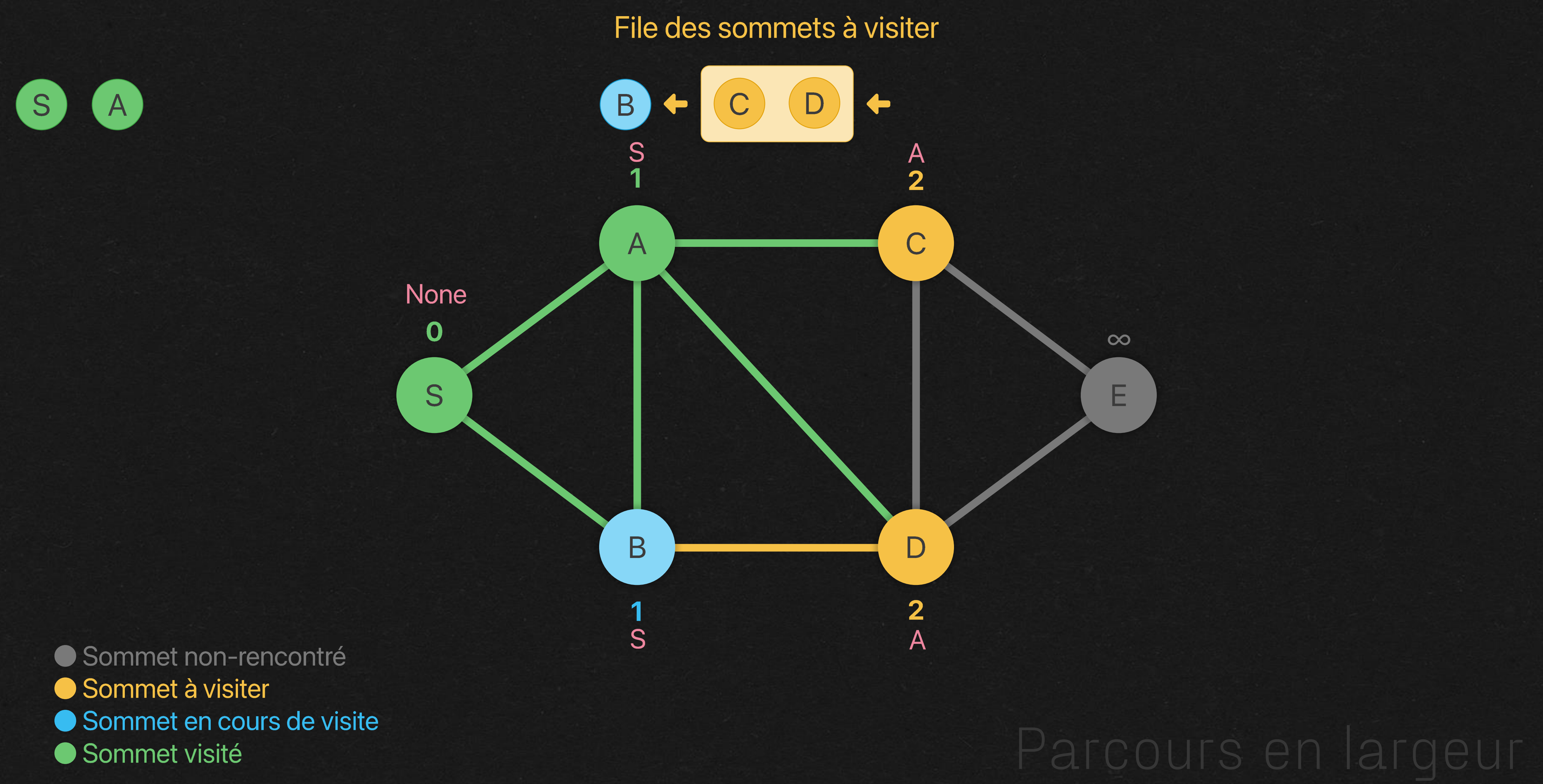
- Sommet non-rencontré
- Sommet à visiter
- Sommet en cours de visite
- Sommet visité

Parcours en largeur

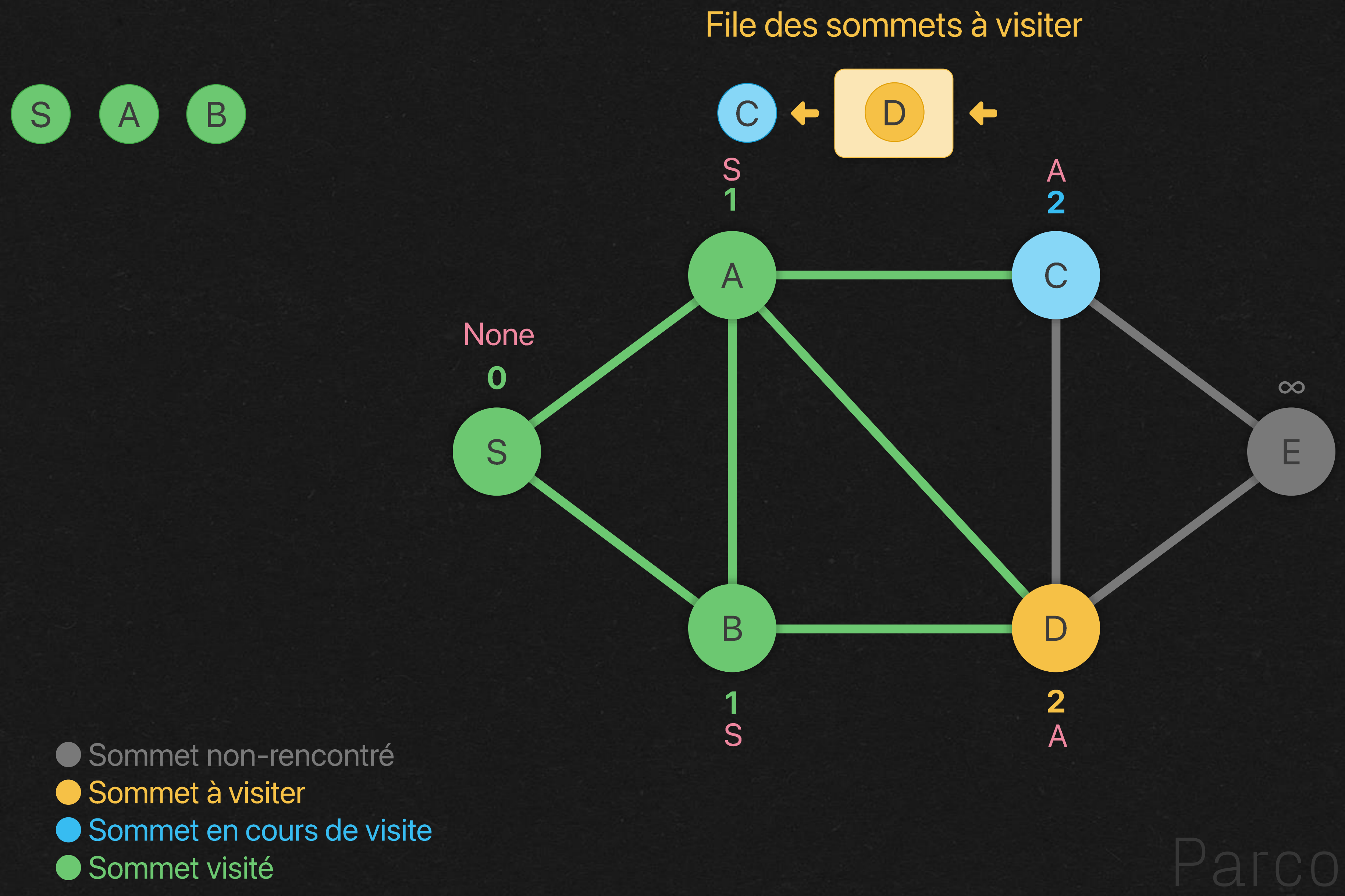


Parcours en largeur

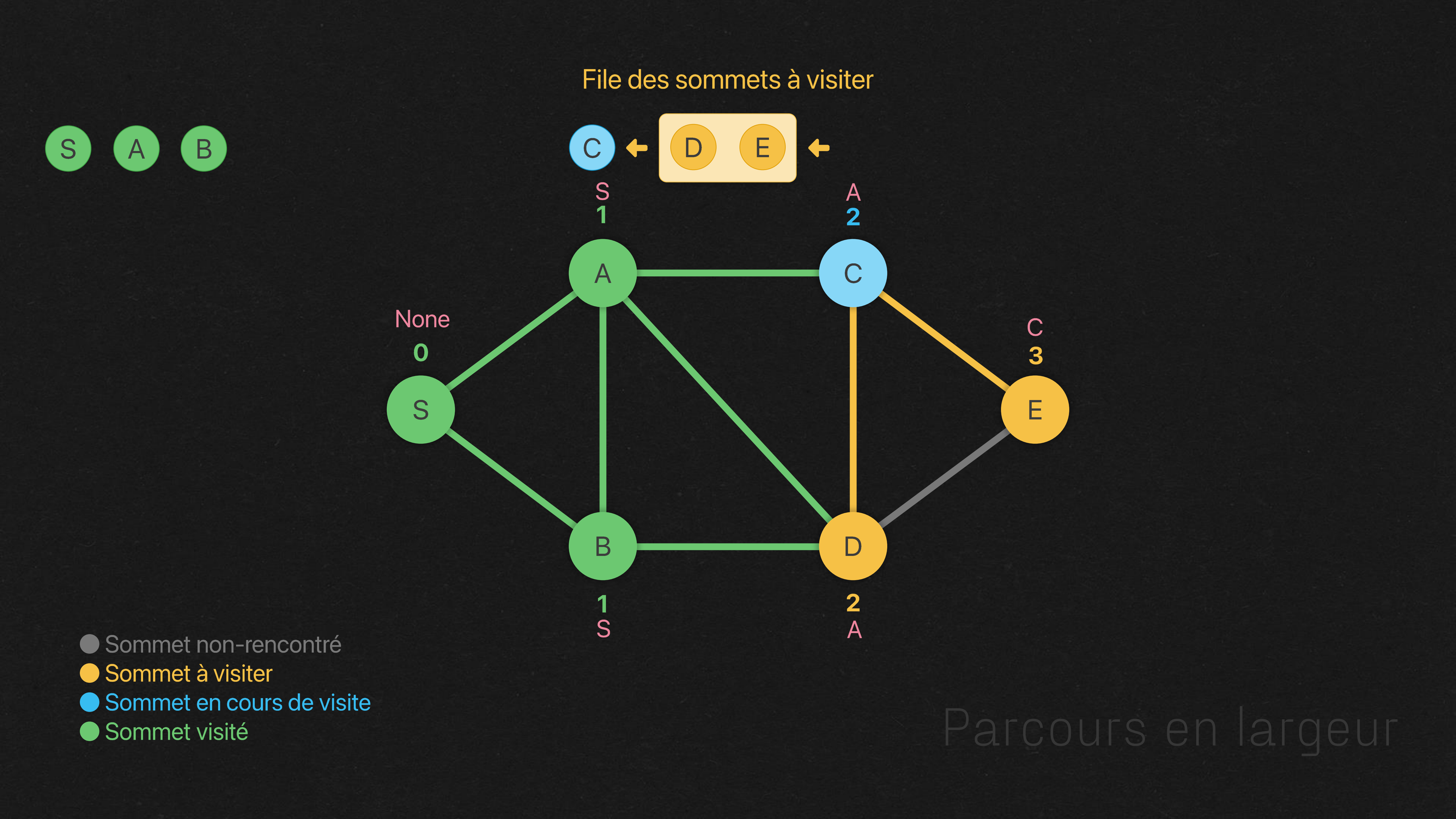

File des sommets à visiter
S
A
B
C
D
S
1
A
2
None
0
∞
S
A
C
E
B
D
1
S
2
A
Sommet non-rencontré
Sommet à visiter
Sommet en cours de visite
Sommet visité
Parcours en largeur

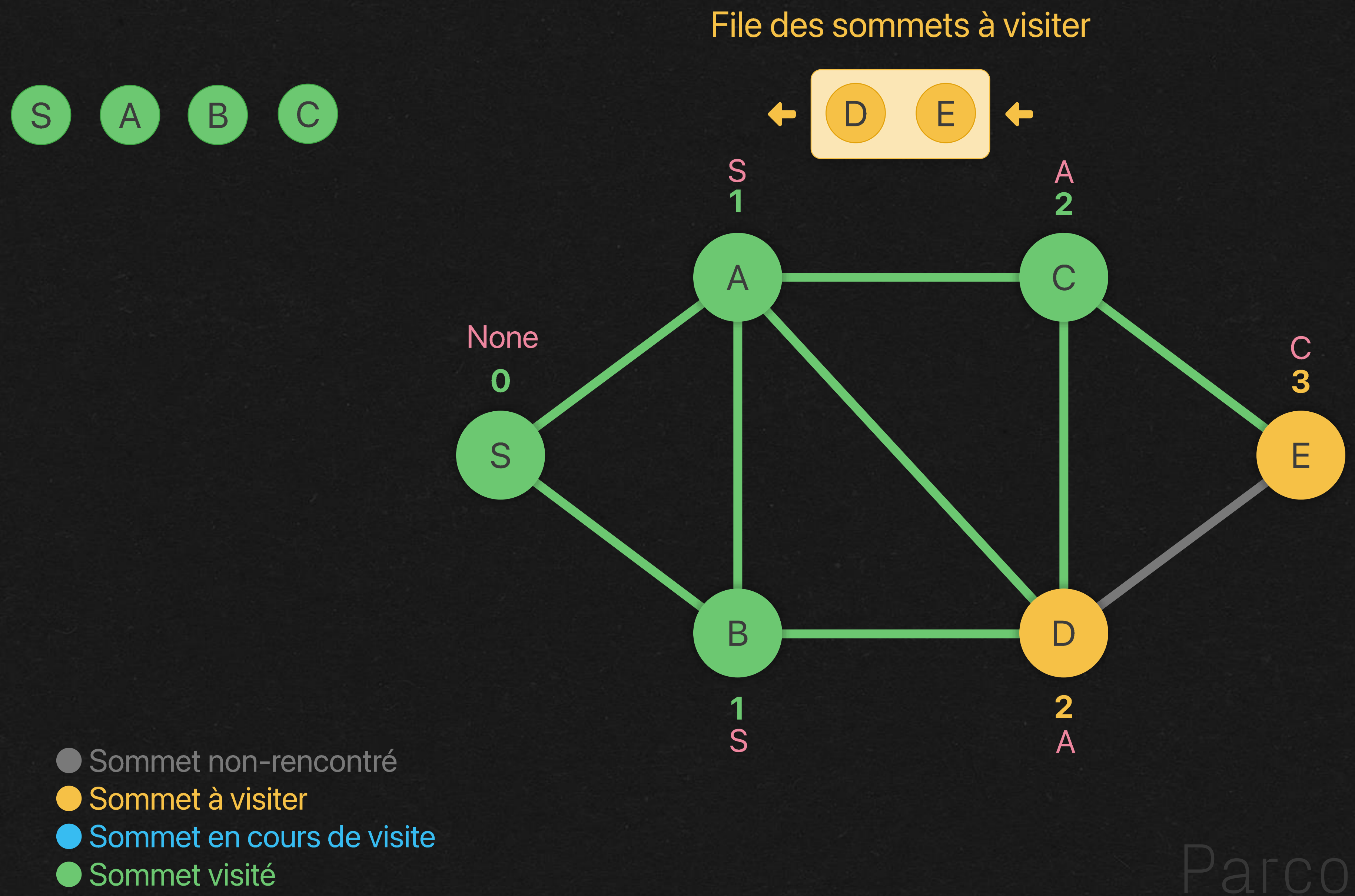


File des sommets à visiter
S
A
B
C
D
S
1
A
2
None
0
∞
A
C
S
E
B
D
1
S
2
A
Sommet non-rencontré
Sommet à visiter
Sommet en cours de visite
Sommet visité
Parcours en largeur

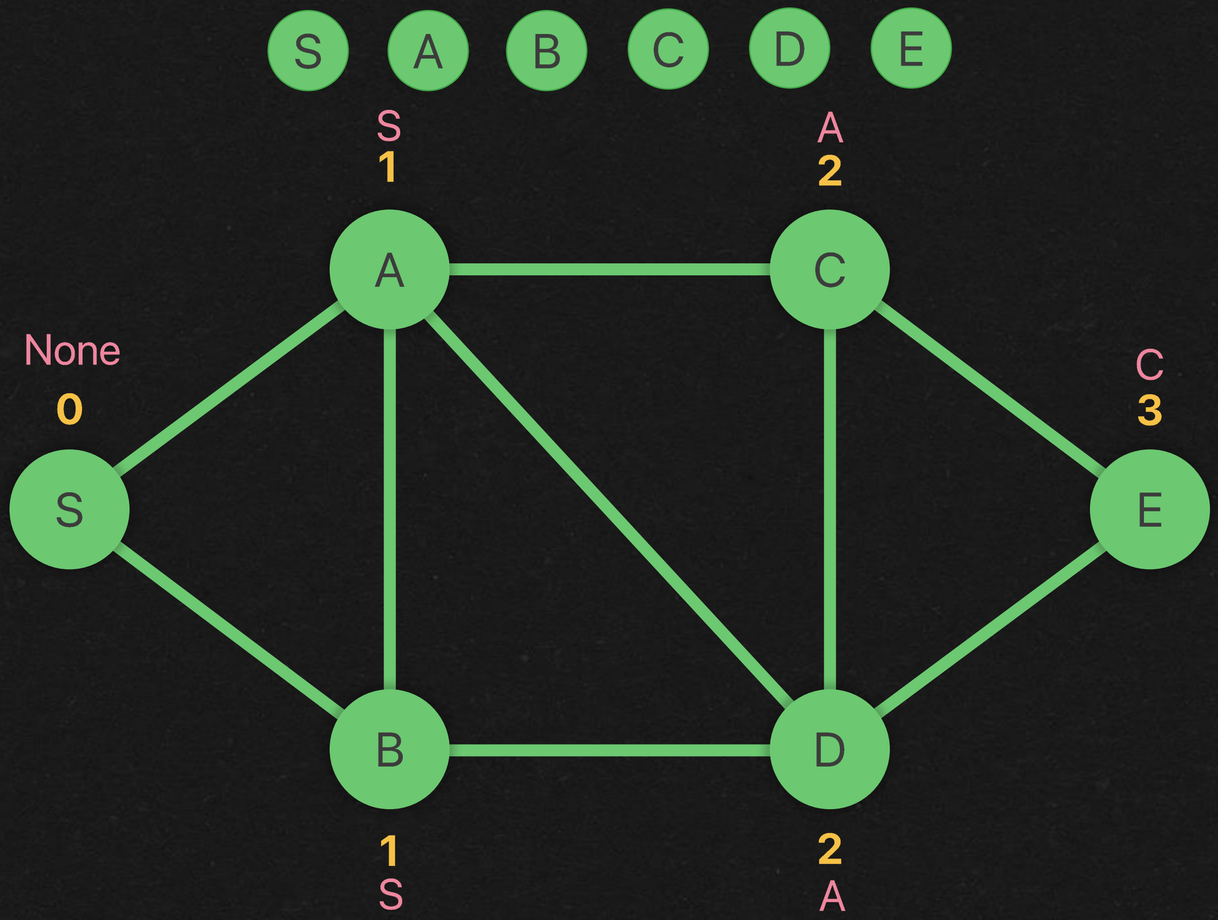


Parcours en largeur


File des sommets à visiter
S
A
B
C
D
E
S
1
A
2
None
0
C
3
1
S
2
A
Sommet non-rencontré
Sommet à visiter
Sommet en cours de visite
Sommet visité
Parcours en largeur



Parcours en largeur



Sommet non-rencontré
Sommet à visiter
Sommet en cours de visite
Sommet visité

Parcours en largeur

Algorithme final

```
Entrées: Un graphe G et un sommet initial s
Sortie: Chemins et distances des plus courts chemin de s
        vers tous les sommets

distance[u] = ∞ pour tout sommet u de G
distance[s] = 0
prédécesseur[u] = None pour tout sommet u de G

initialiser une file F vide
enfiler s dans F
marquer s

tant que F n’est pas vide
    défiler F dans u
    pour chaque voisin v non-marqué de u
        enfiler v dans F
        marquer v
        distance[v] = distance[u] + 1
        prédécesseur[v] = u

retourner distance, prédécesseur
```

Algorithme final

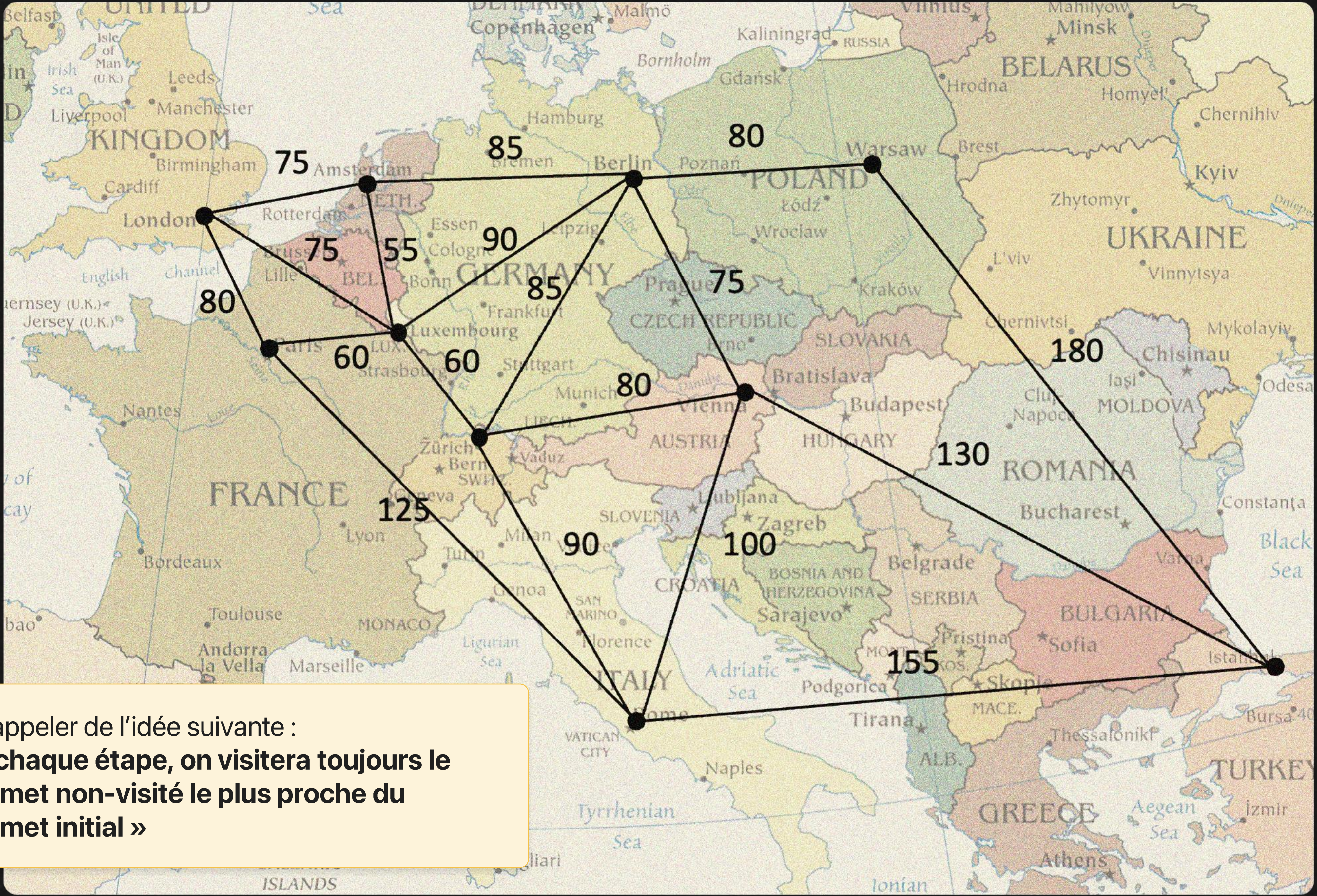
```
Entrées: Un graphe G et un sommet initial s
Sortie: Chemins et distances des plus courts chemin de s
        vers tous les sommets

distance[u] = ∞ pour tout sommet u de G
distance[s] = 0
prédécesseur[u] = None pour tout sommet u de G

initialiser une file F vide
enfiler s dans F
marquer s

tant que F n’est pas vide
    défiler F dans u
    pour chaque voisin v non-marqué de u
        enfiler v dans F
        marquer v
        distance[v] = distance[u] + 1
        prédécesseur[v] = u

retourner distance, prédécesseur
```

Piouf ! On a résolu le problème du plus court chemin pour un **graphe non-pondéré** grâce au parcours en largeur !

Fonctionne aussi pour les **graphes pondérés uniformément** (même poids sur les arêtes).

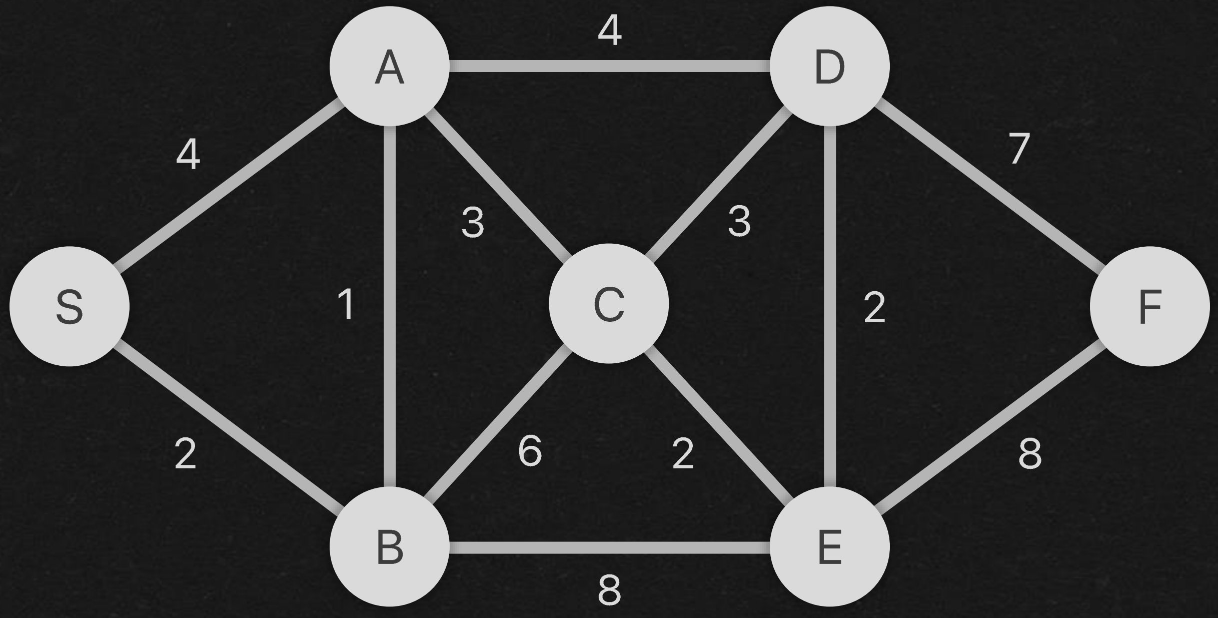
Mais comment trouver le plus court chemin pour un **graphe pondérés** cette fois-ci ?

Se rappeler de l’idée suivante :
« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »



Se rappeler de l'idée suivante :

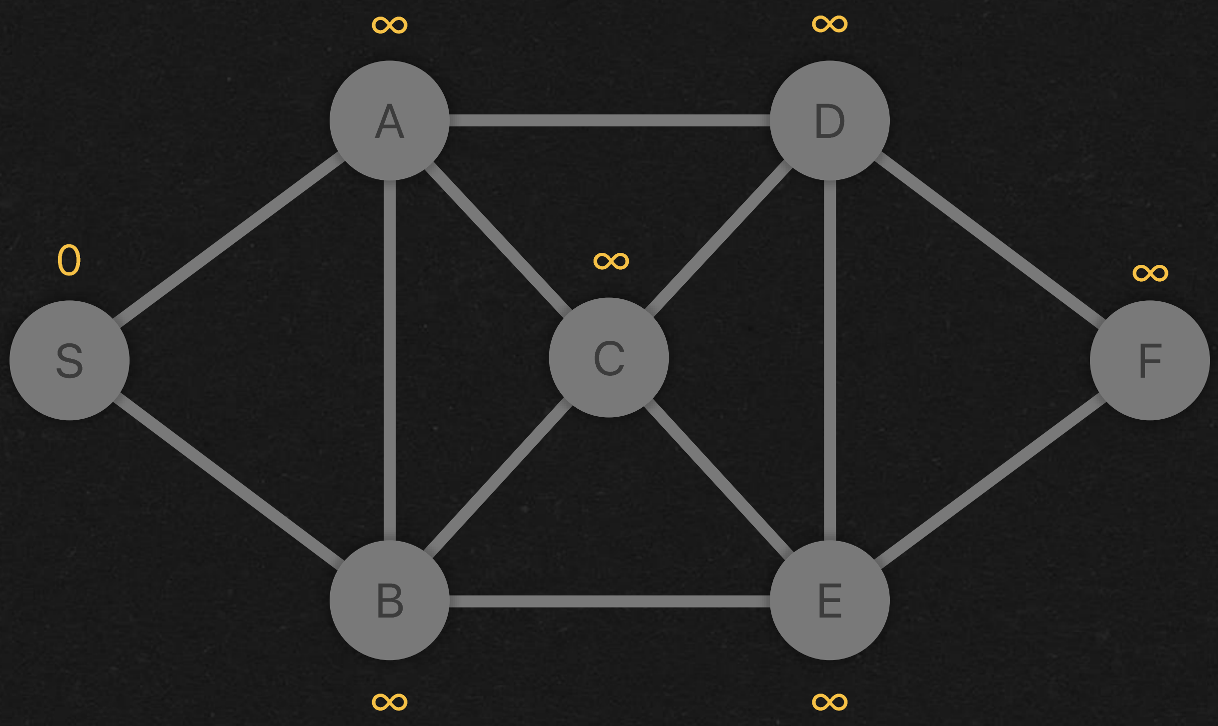
« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

> « À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

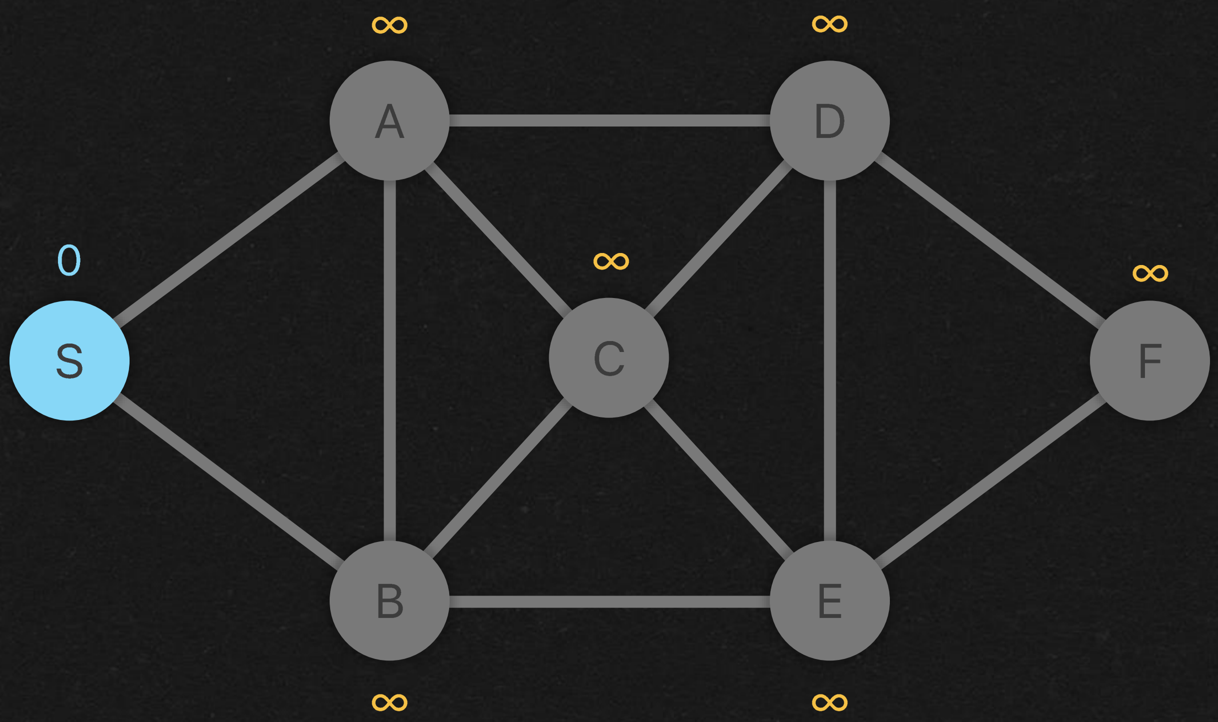
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

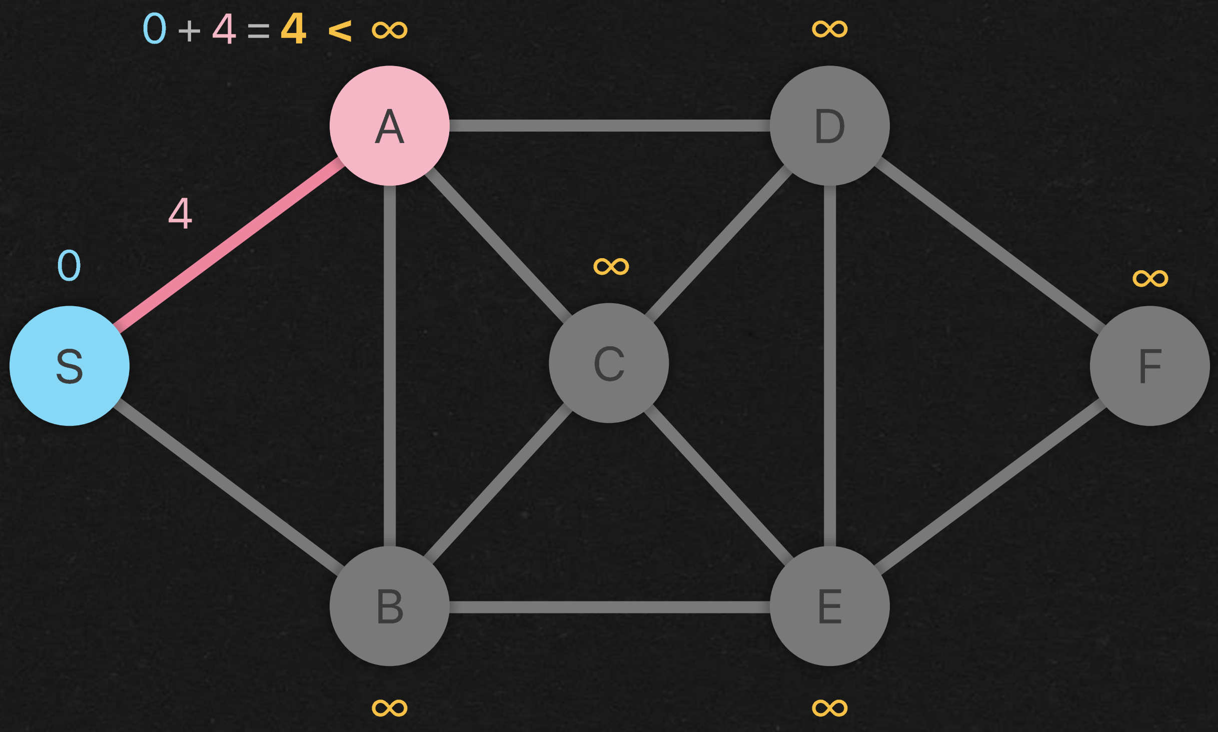
Algorithme de Dijkstra



- Sommet non-visité
- Sommet en cours de visite
- Voisin courant
- Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

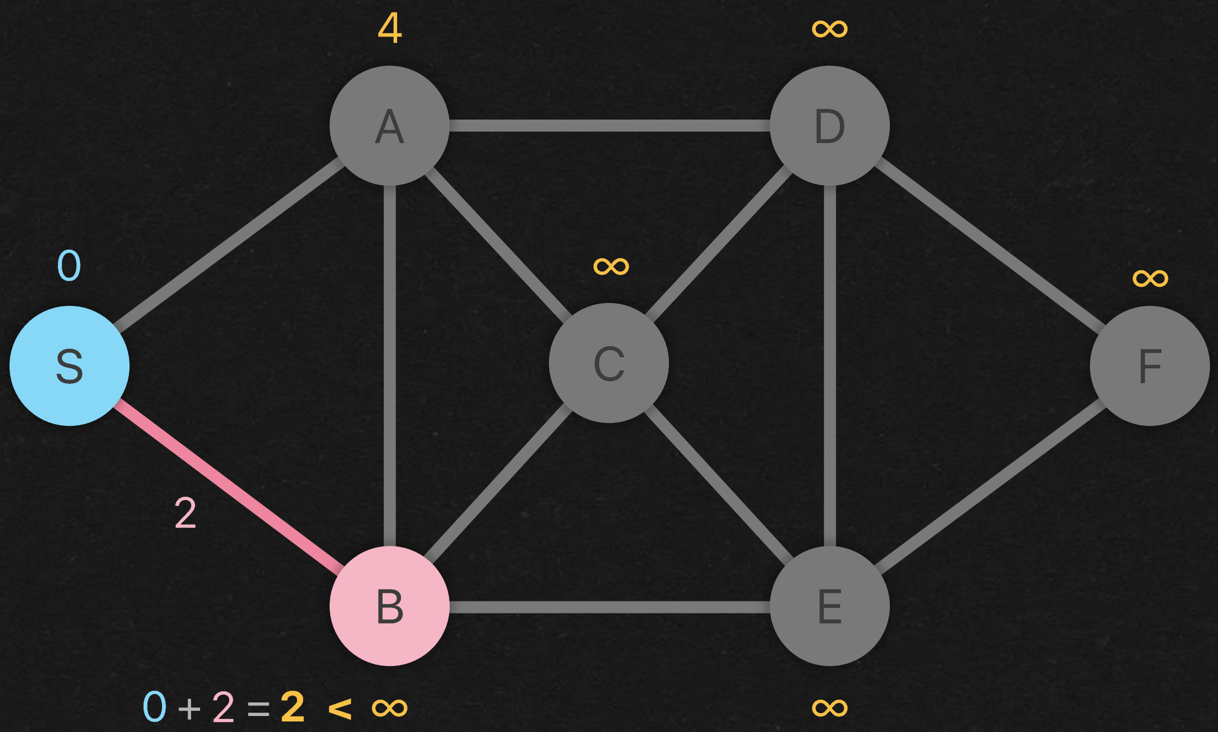
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

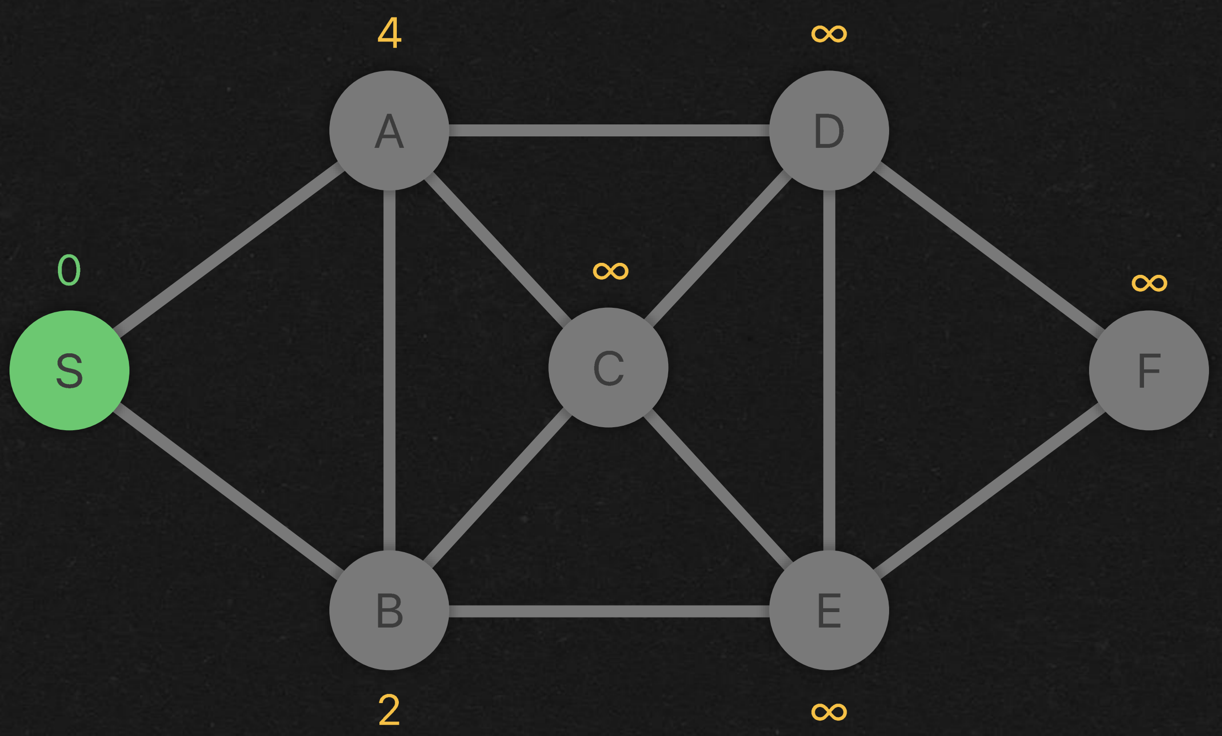
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

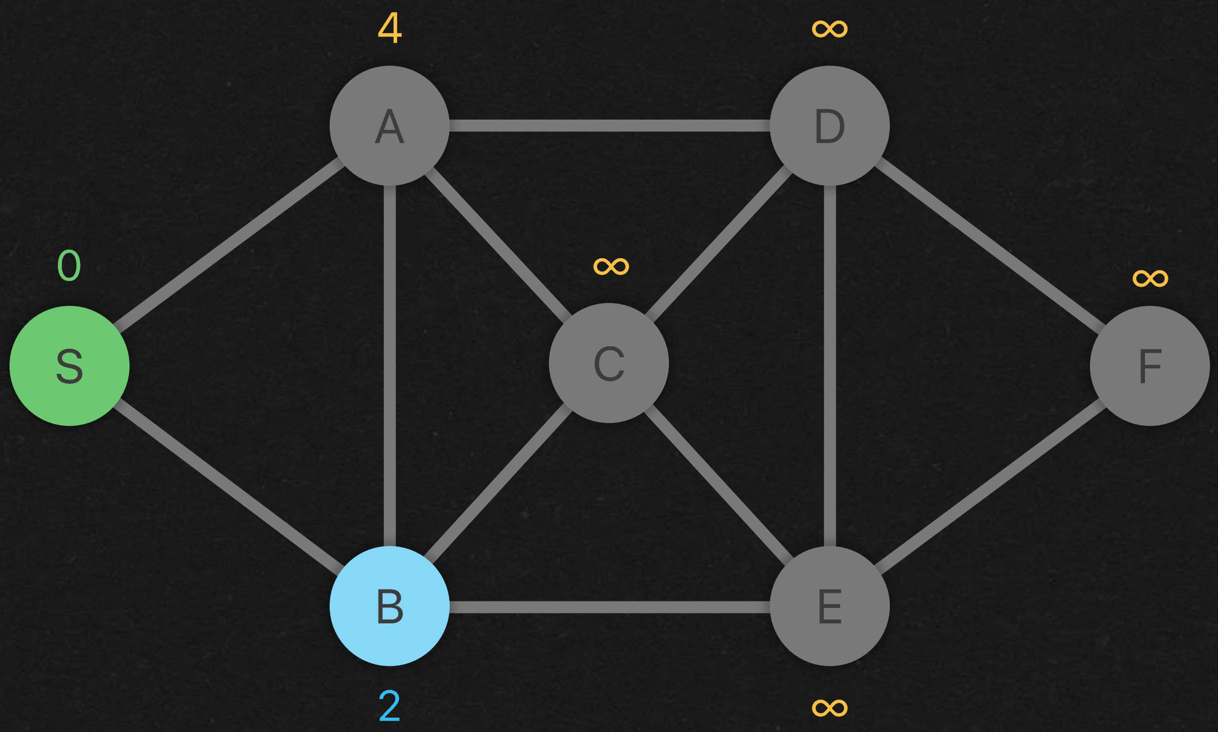
Algorithme de Dijkstra



- Sommet non-visité
- Sommet en cours de visite
- Voisin courant
- Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

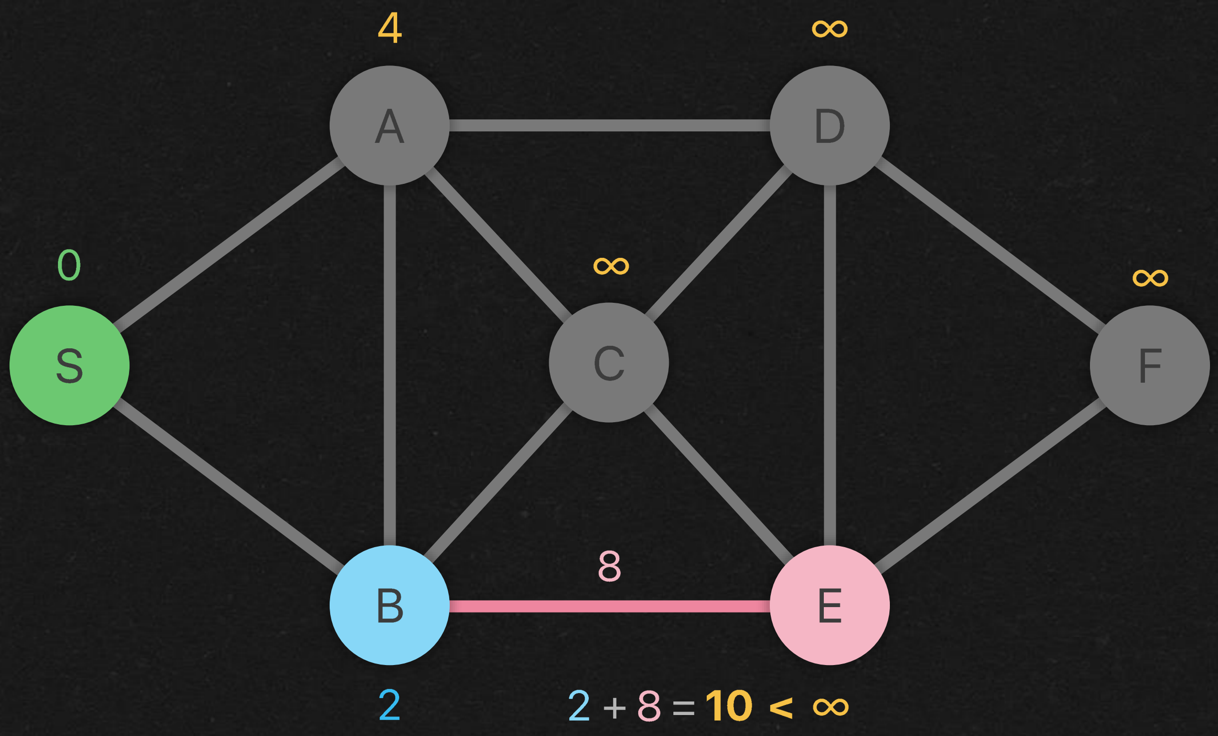
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

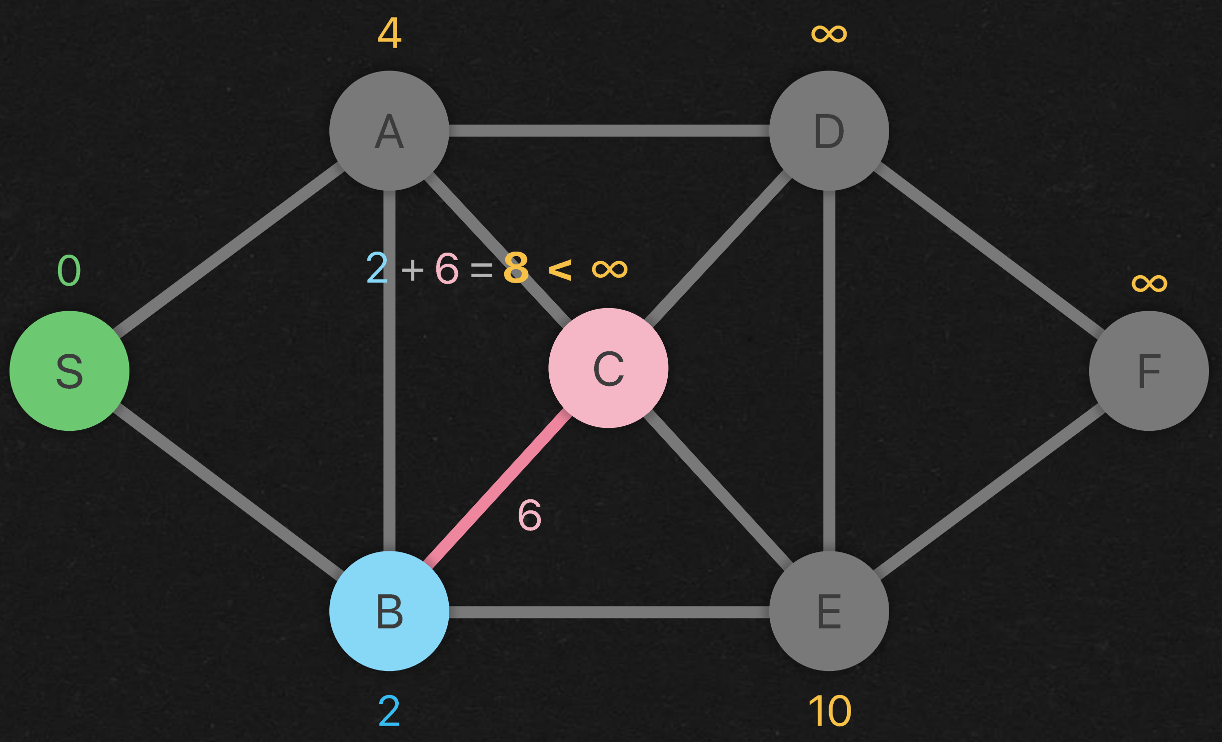
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

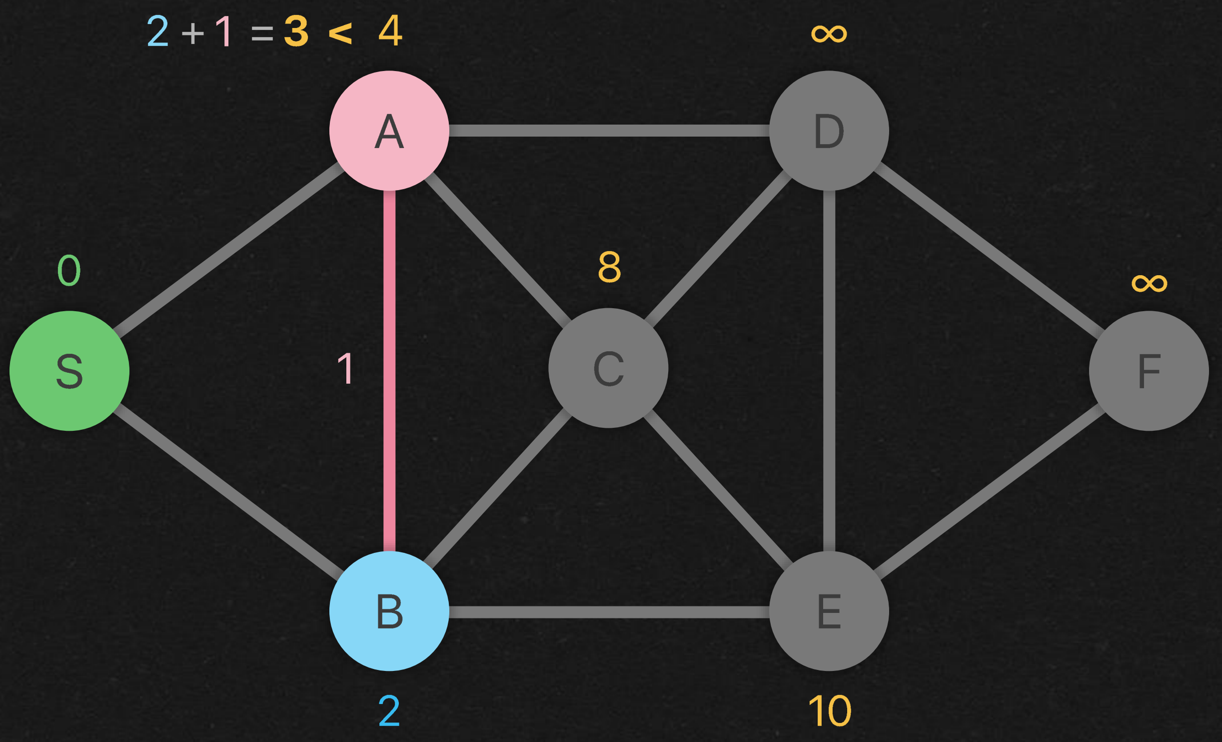
Algorithme de Dijkstra



- Sommet non-visité
- Sommet en cours de visite
- Voisin courant
- Sommet visité

> « À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

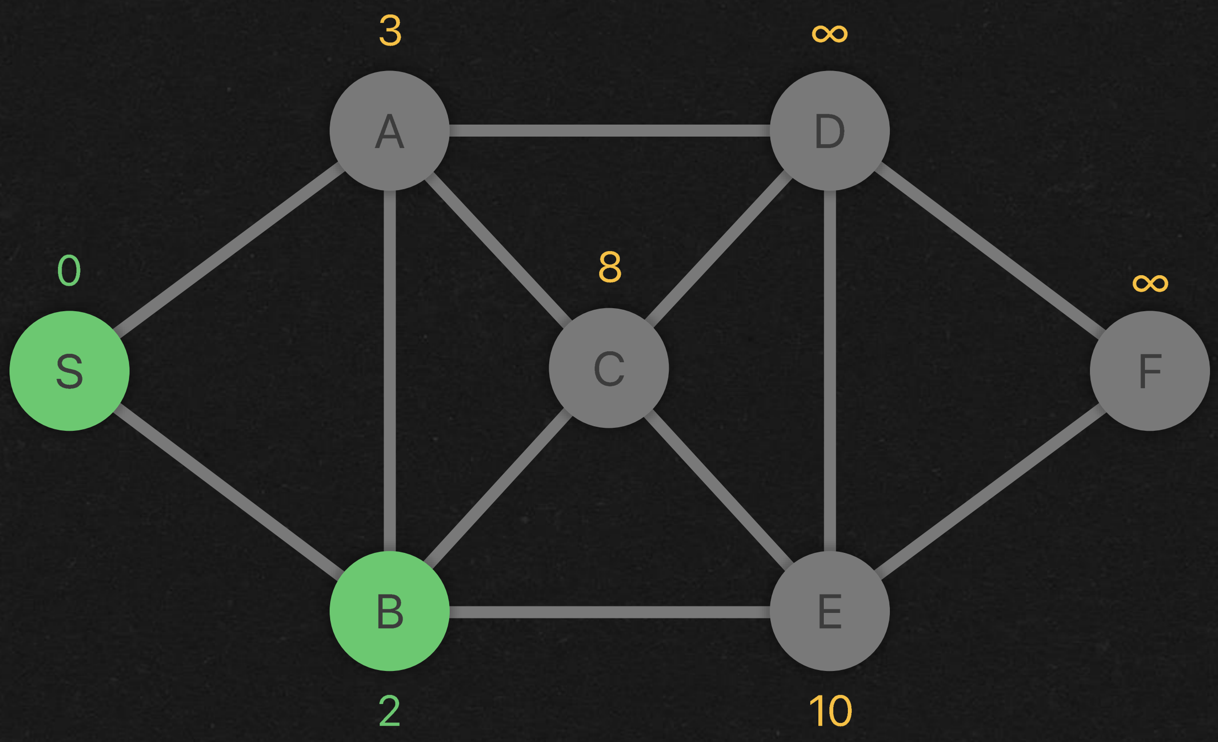
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

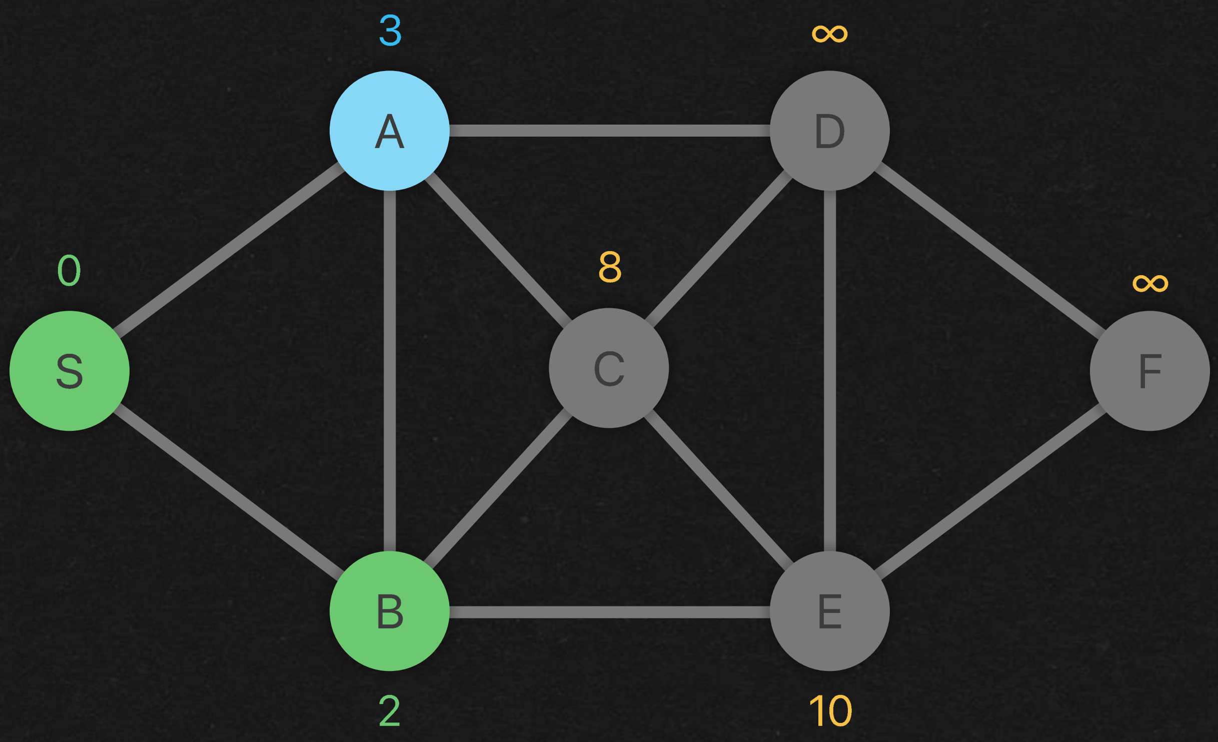
Algorithme de Dijkstra



- Sommet non-visité
- Sommet en cours de visite
- Voisin courant
- Sommet visité

> « À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

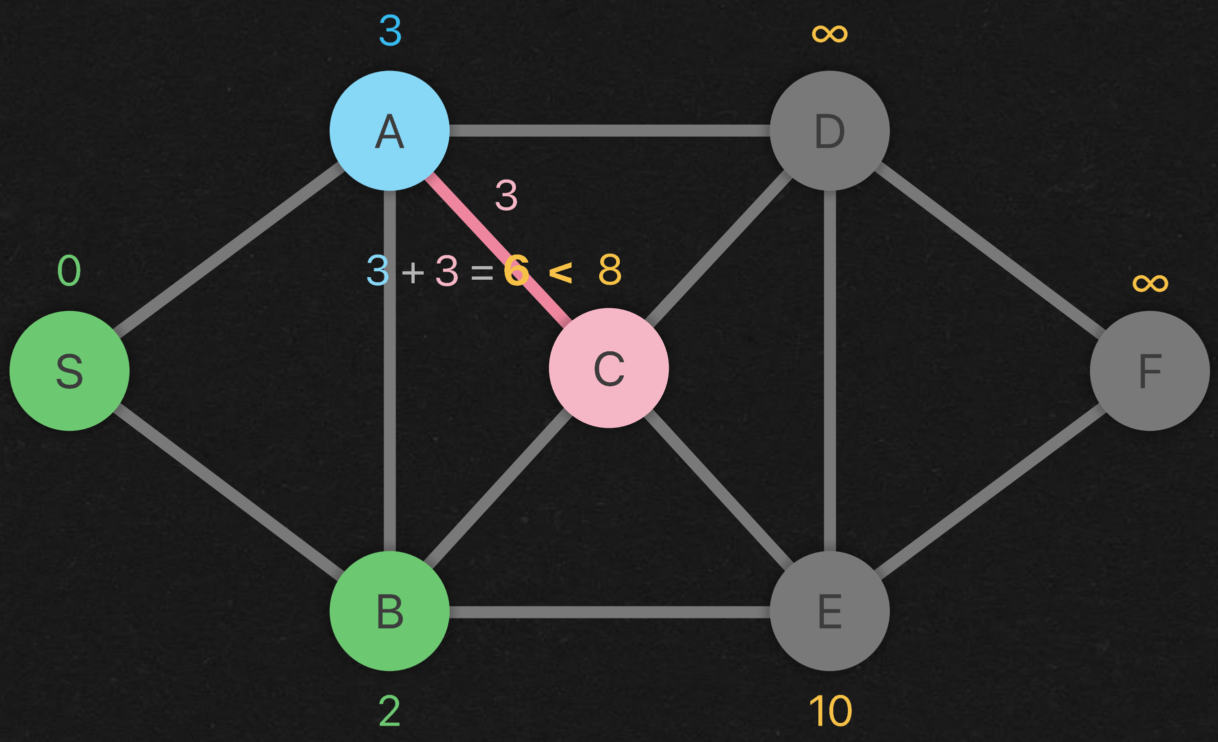
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

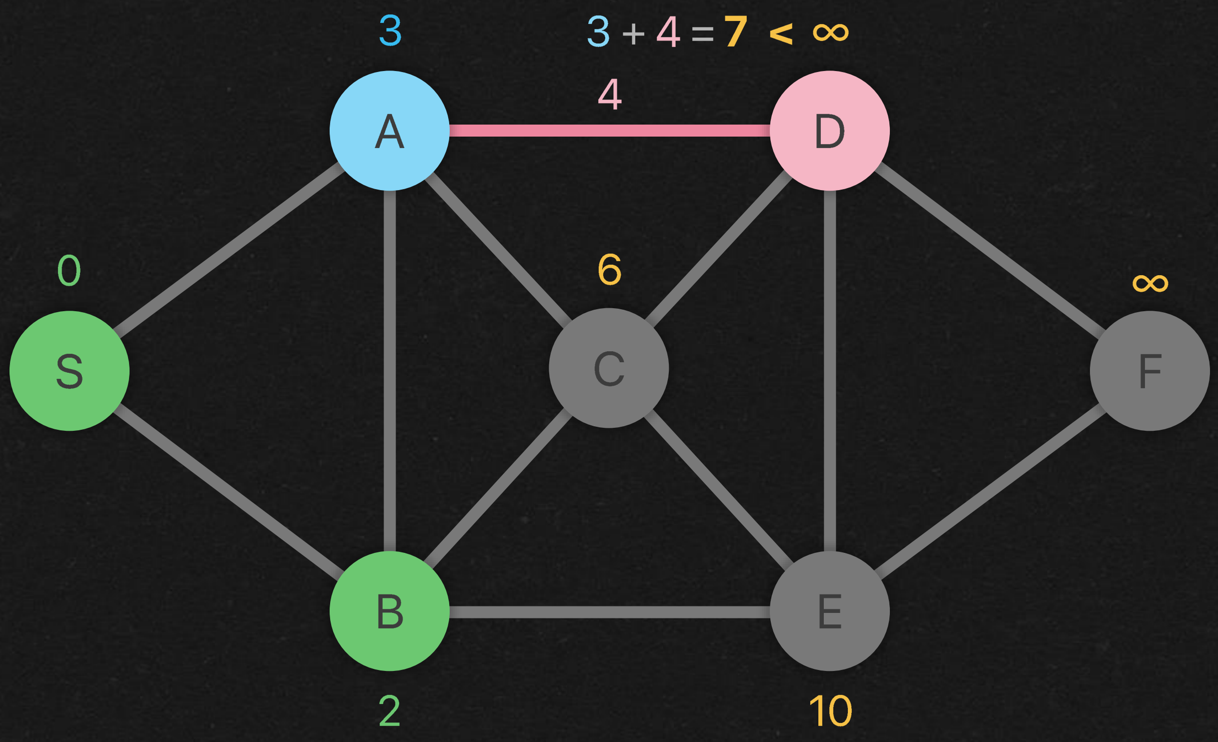
Algorithme de Dijkstra



- Sommet non-visité
- Sommet en cours de visite
- Voisin courant
- Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

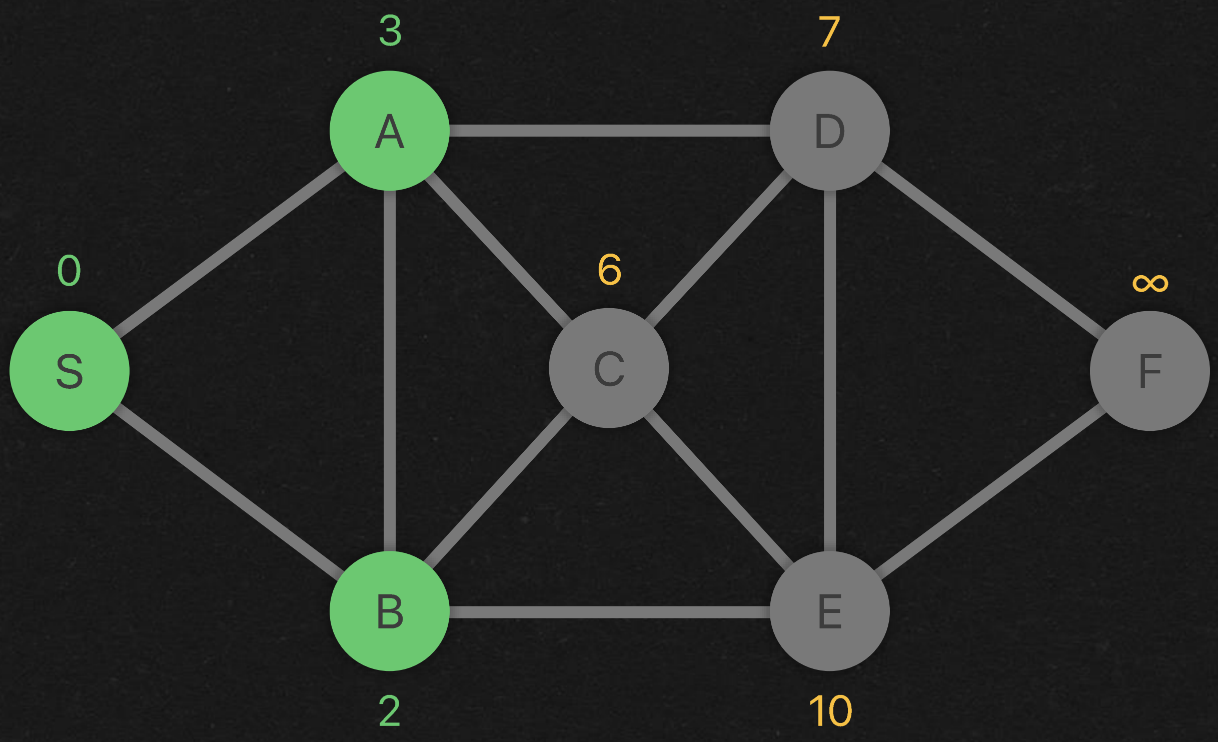
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

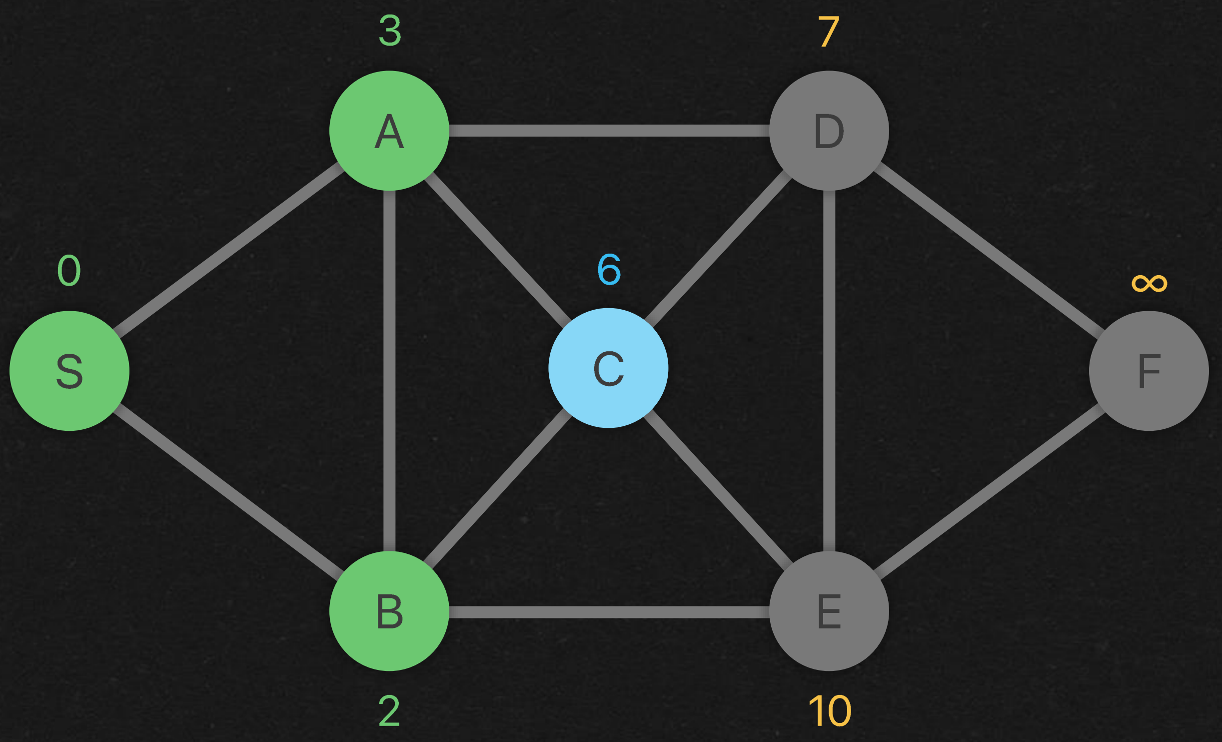
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

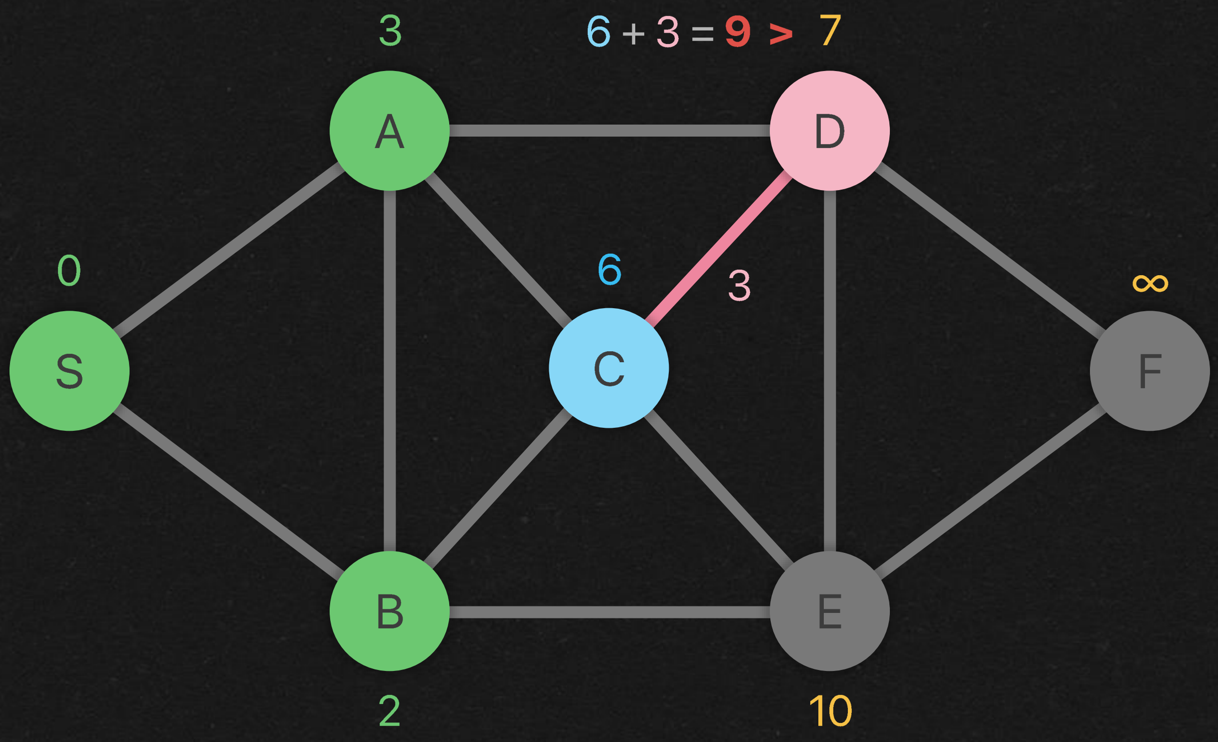
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

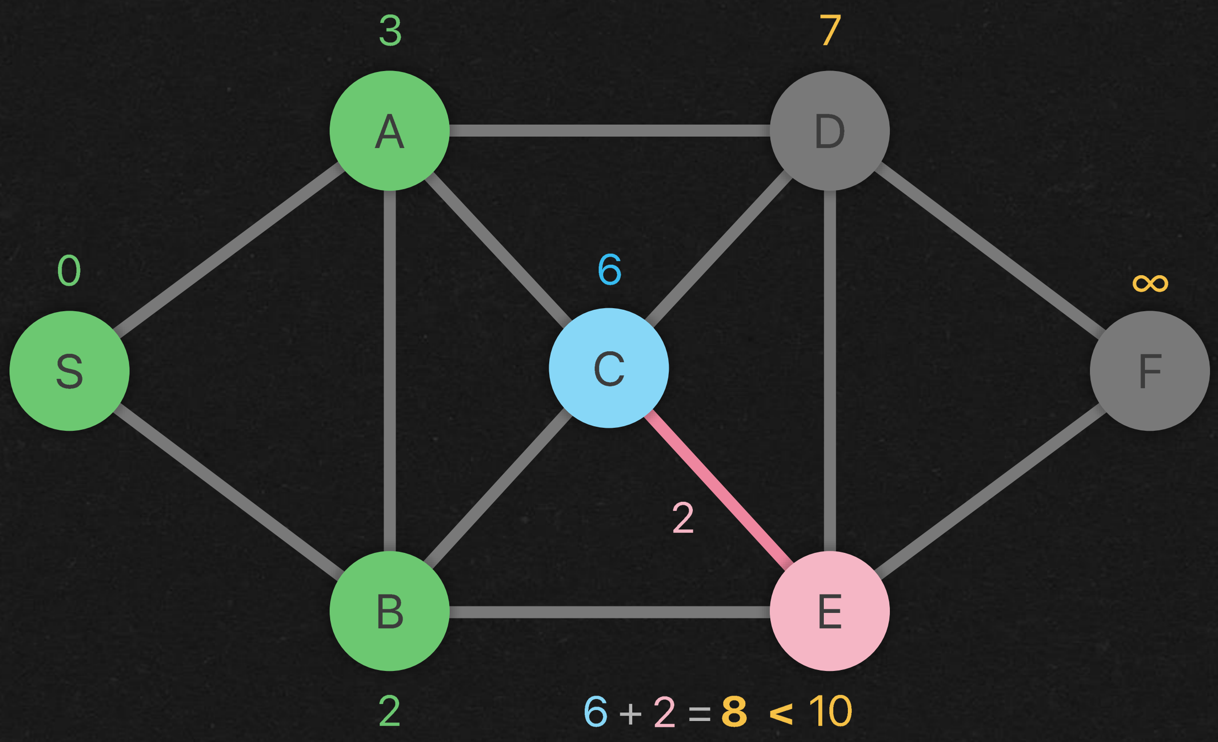
Algorithme de Dijkstra



- Sommet non-visité
- Sommet en cours de visite
- Voisin courant
- Sommet visité

> « À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

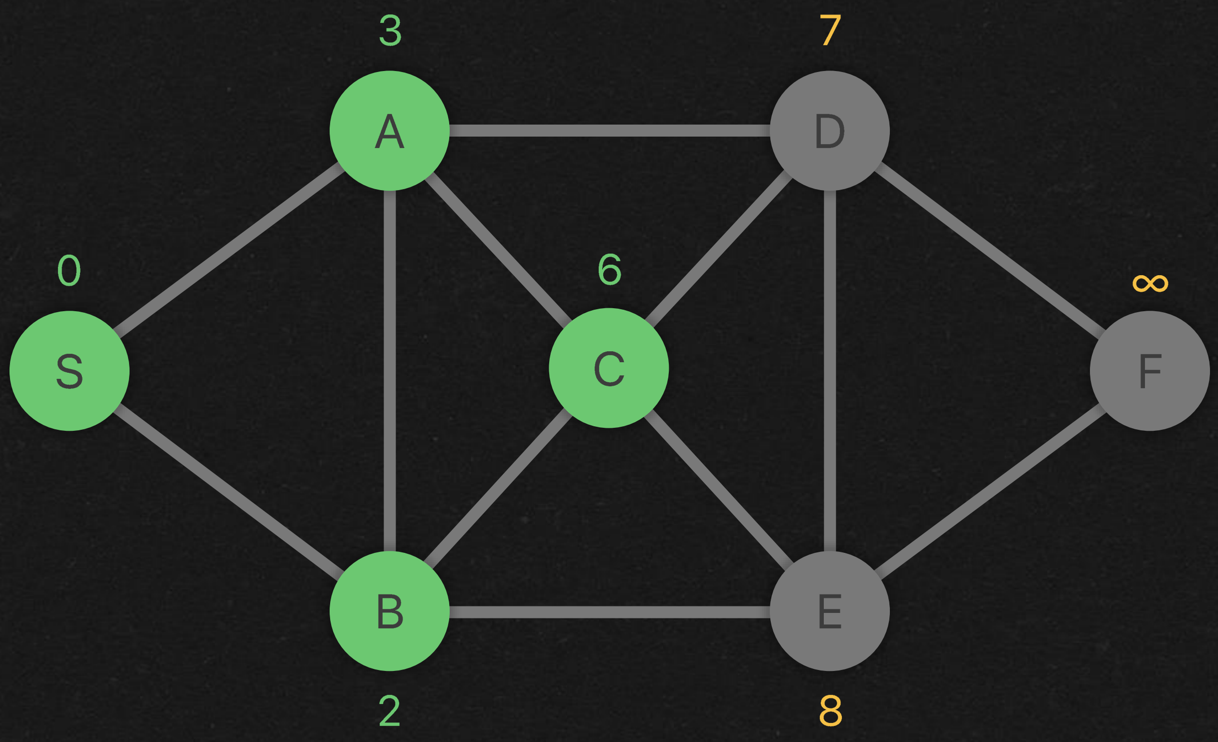
Algorithme de Dijkstra



- Sommet non-visité
- Sommet en cours de visite
- Voisin courant
- Sommet visité

> « À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

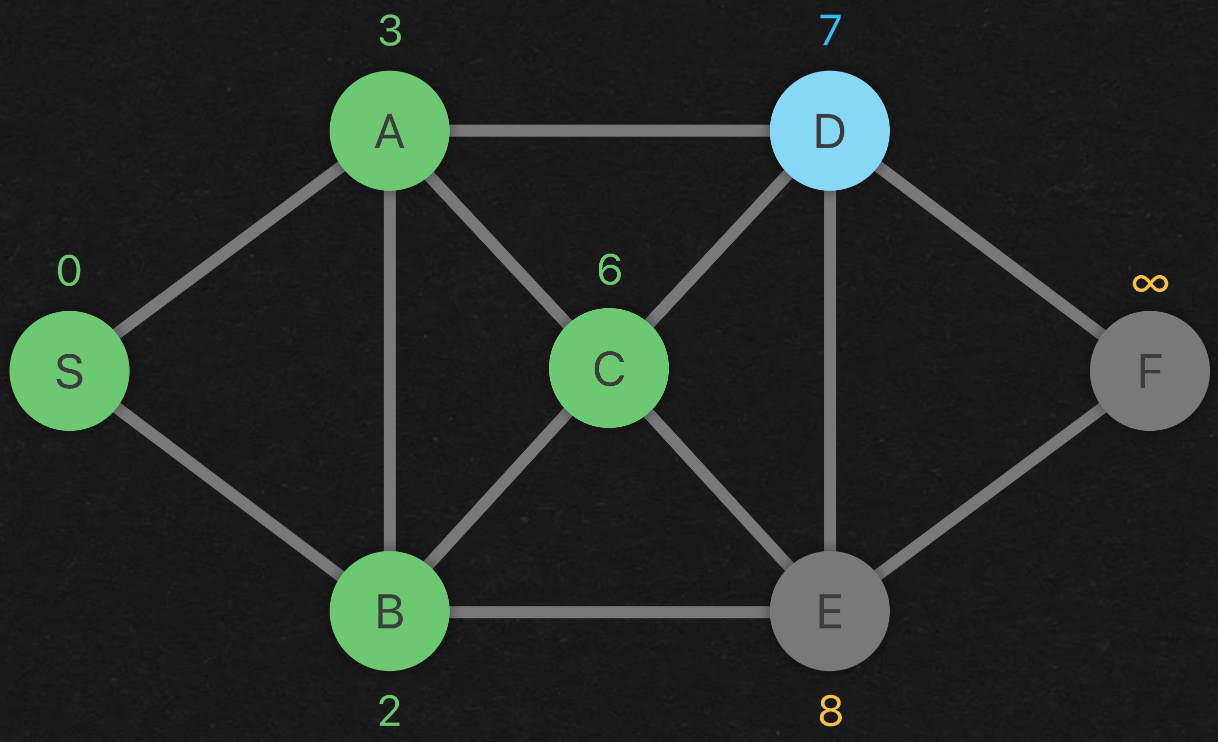
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

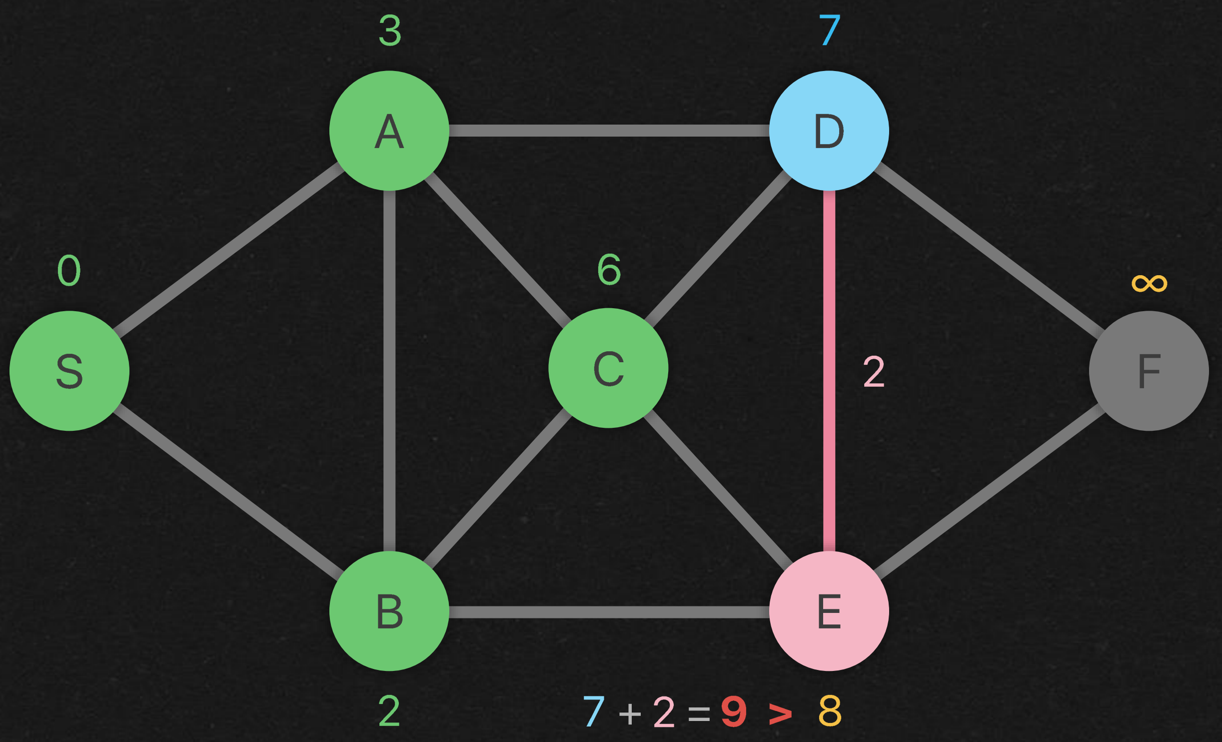
Algorithme de Dijkstra



- Sommet non-visité
- Sommet en cours de visite
- Voisin courant
- Sommet visité

> « À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

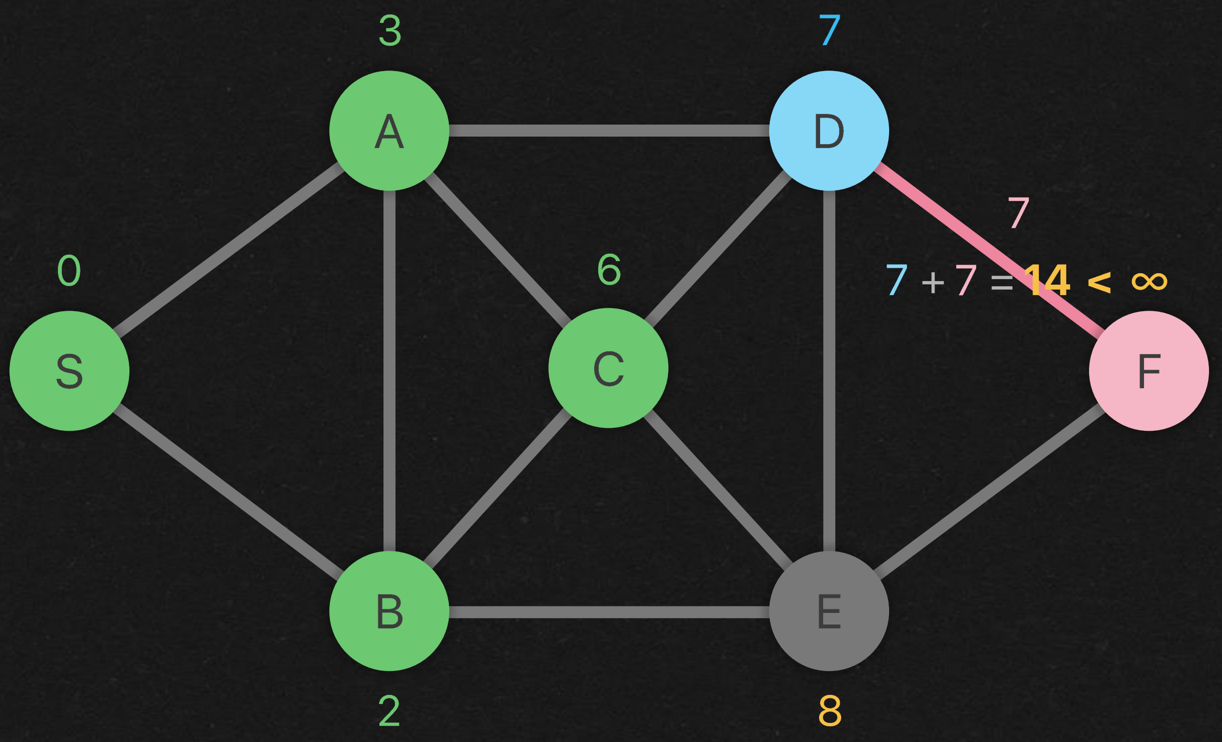
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

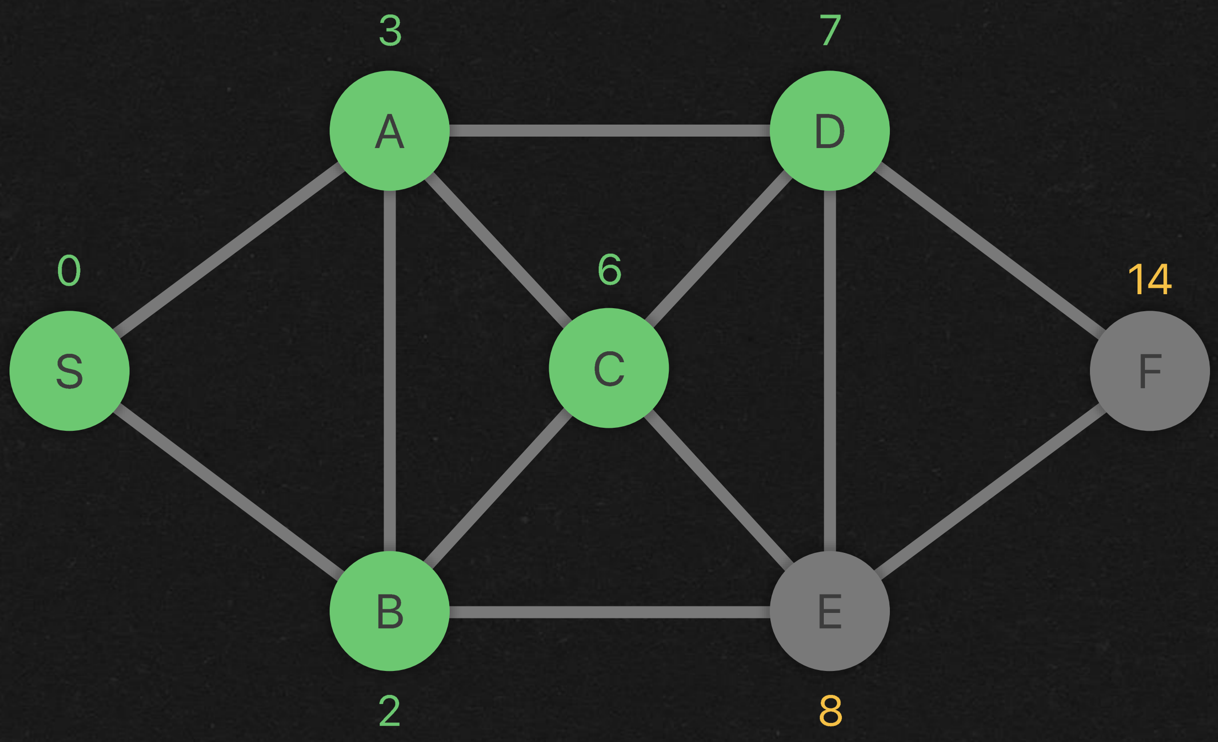
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

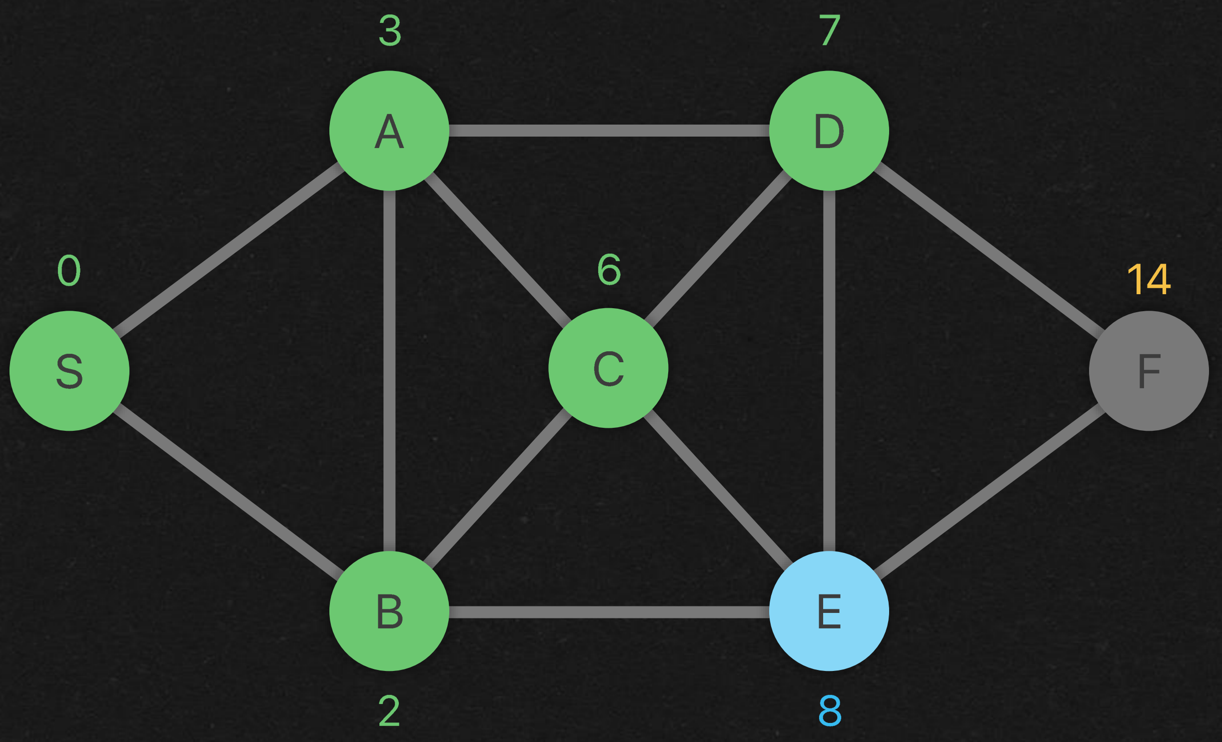
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

> « À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

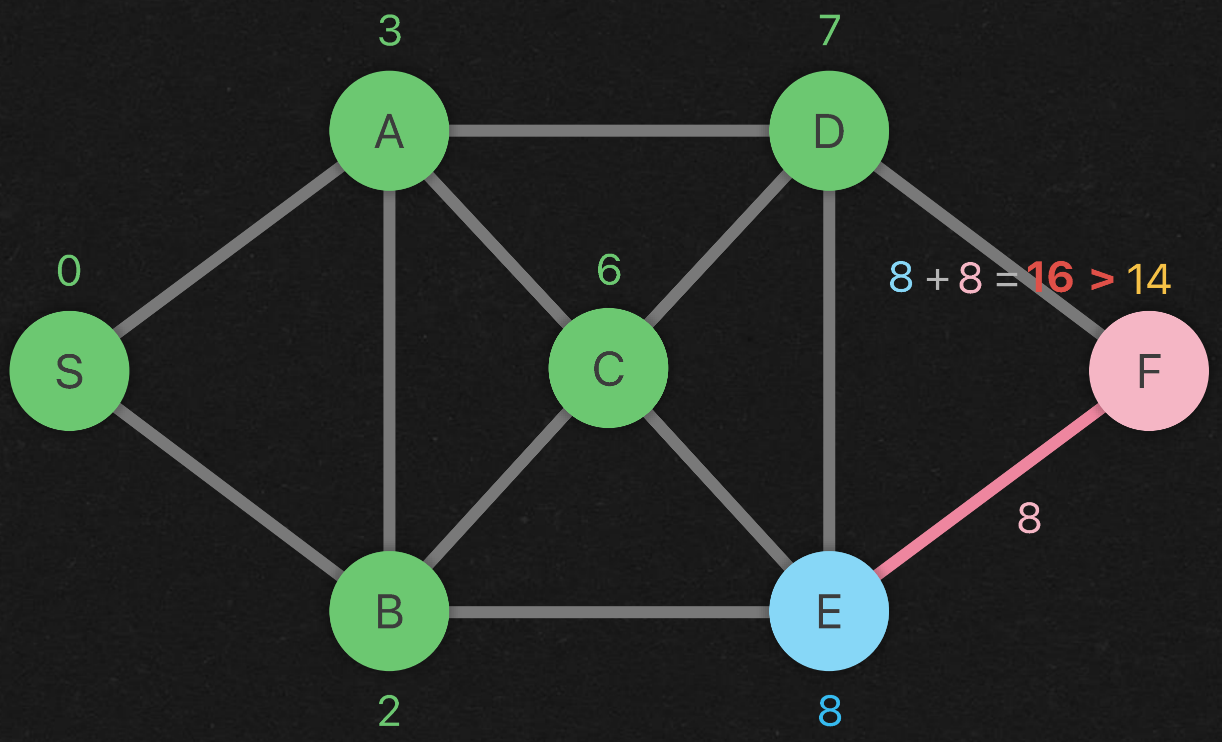
Algorithme de Dijkstra



- Sommet non-visité
- Sommet en cours de visite
- Voisin courant
- Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

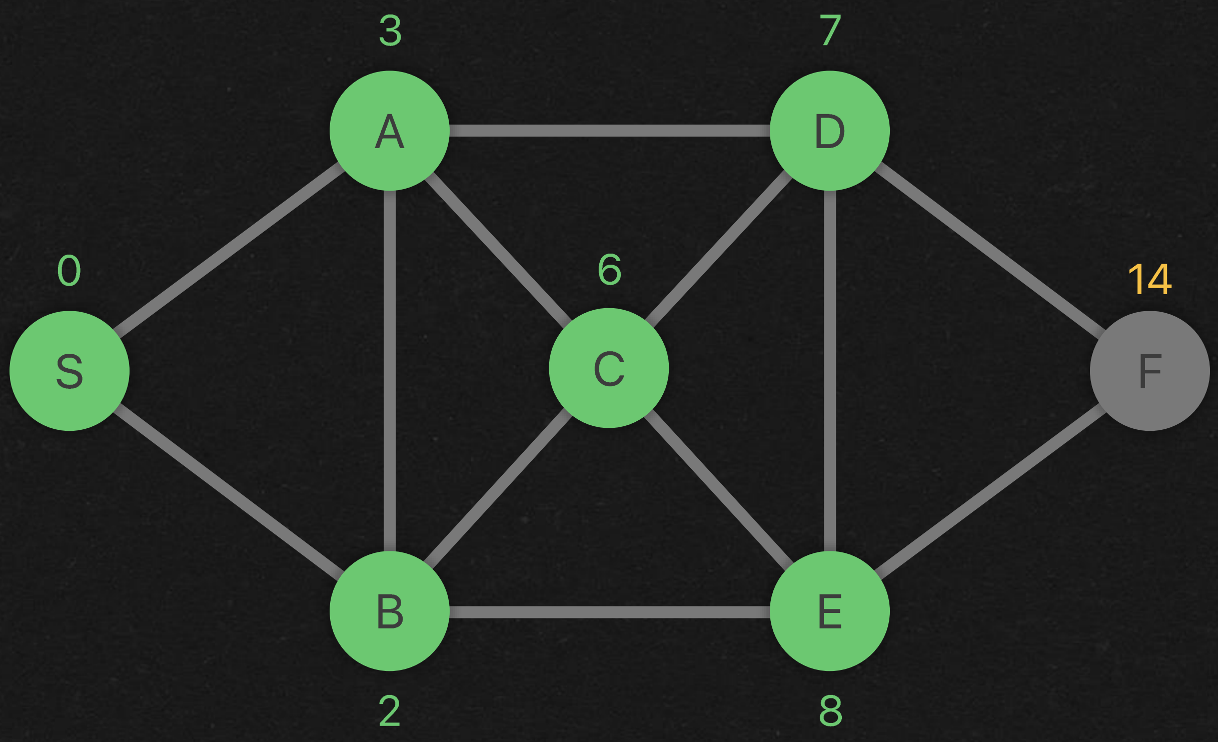
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

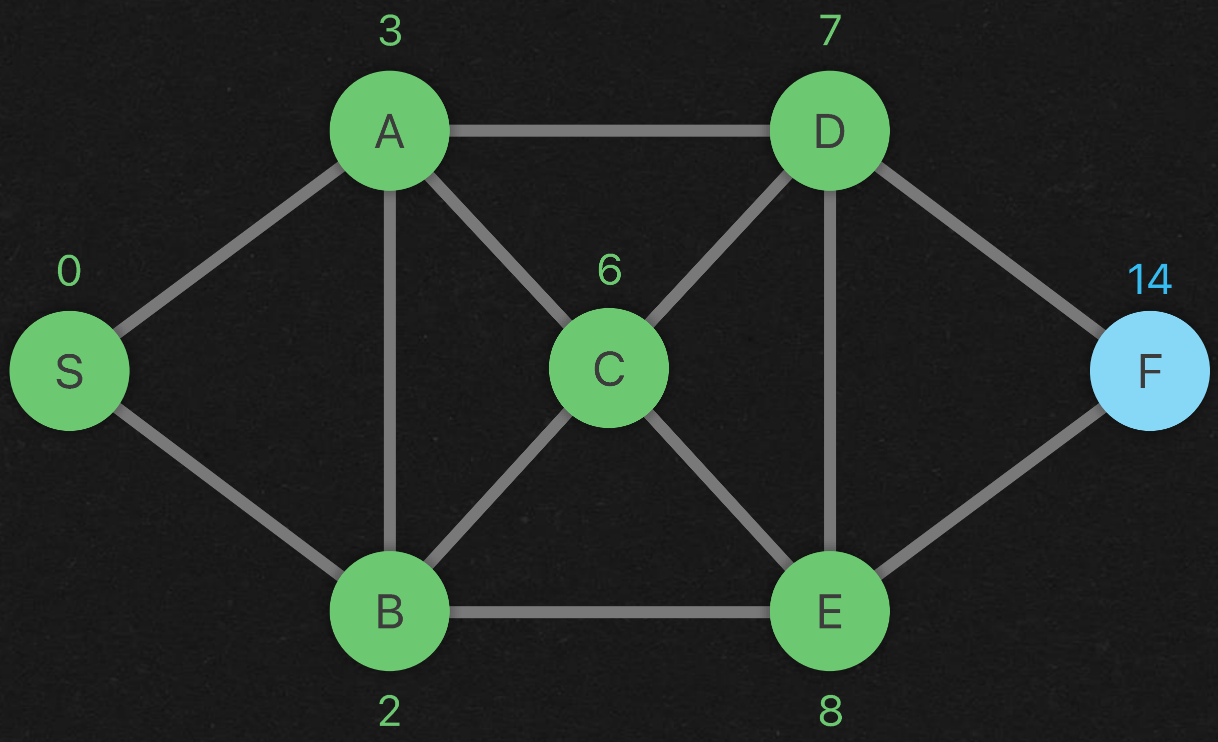
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

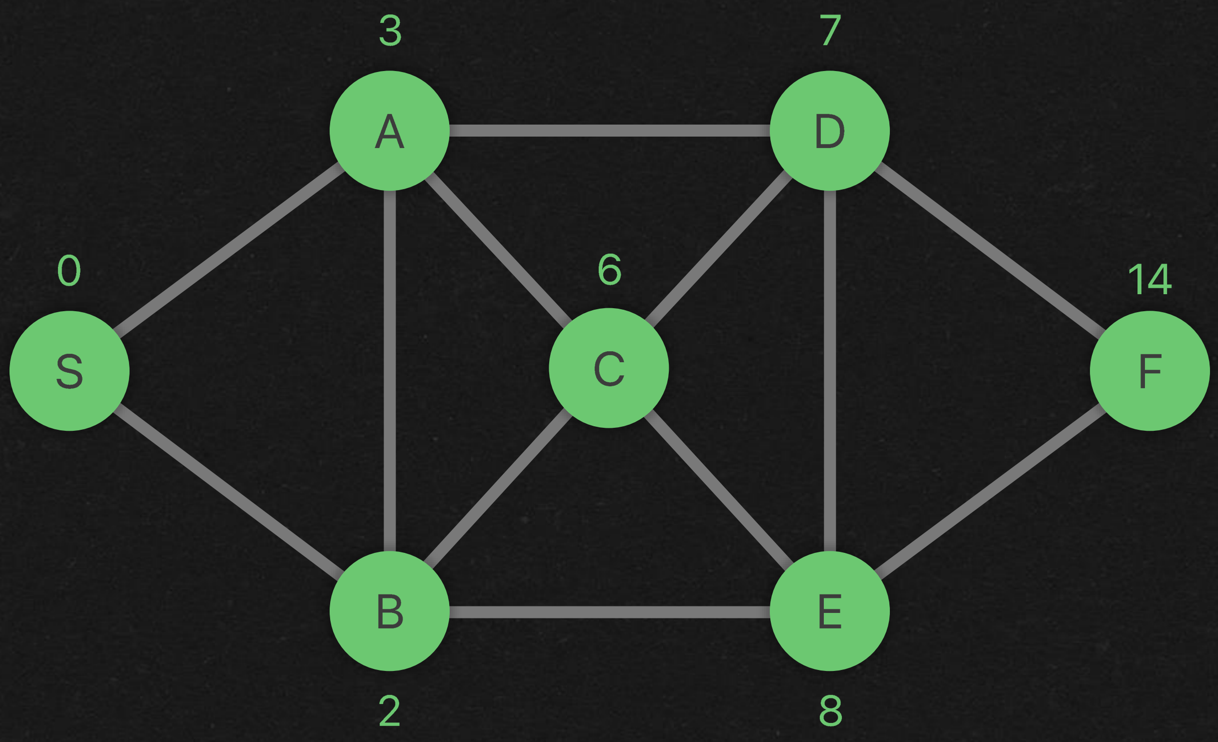
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

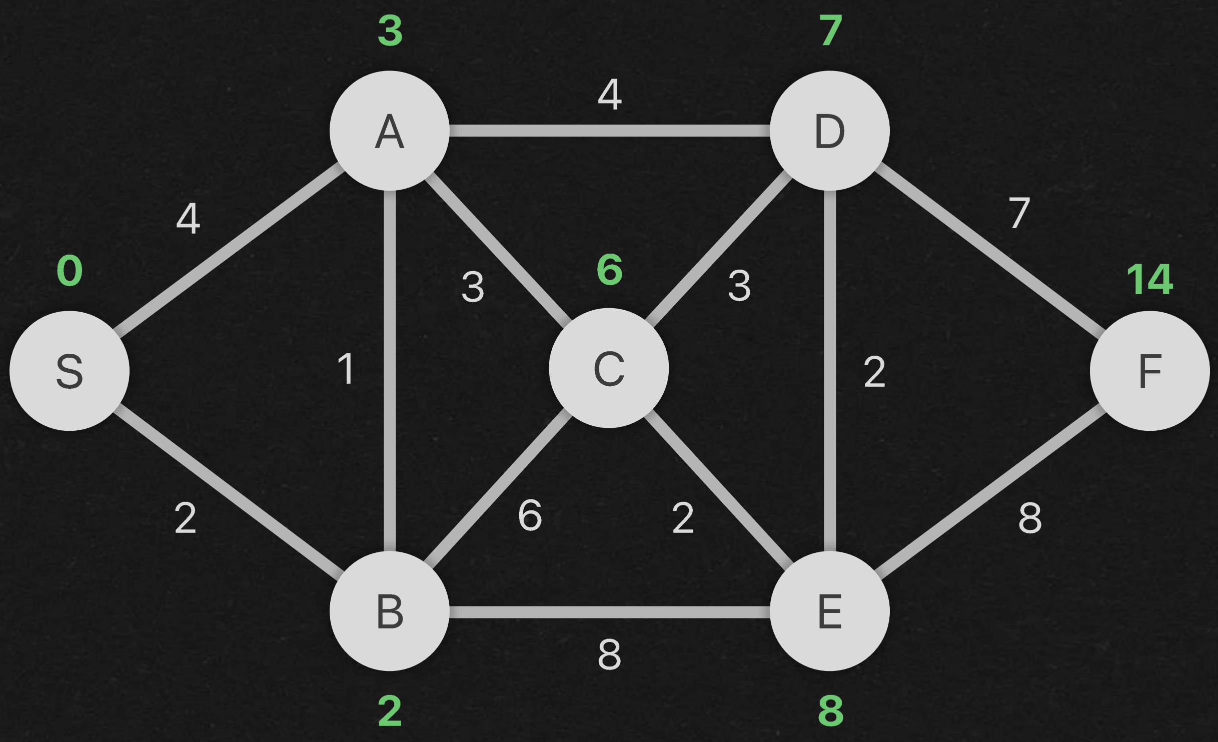
Algorithme de Dijkstra



Sommet non-visité
Sommet en cours de visite
Voisin courant
Sommet visité

« À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

Algorithme de Dijkstra



- Sommet non-visité
- Sommet en cours de visite
- Voisin courant
- Sommet visité

> « À chaque étape, on visitera toujours le sommet non-visité le plus proche du sommet initial »

Algorithme de Dijkstra

Algorithme de Dijkstra

L'algorithme de Dijkstra **est une extension du parcours en largeur** pour les graphes **pondérés**.

Même principe : L'algorithme consiste à découvrir les sommets successivement par ordre croissant de leur distance au sommet initial.

Deux modifications :

- **Sélection du sommet à visiter** : le sommet non-visité le moins distant du sommet initial.
- **Principe de « relaxation d'arc » :** si on découvre une distante plus courte vers un sommet non-visité, on la met à jour.

Proposer un pseudocode de l'algorithme de Dijkstra !

Pseudocode Dijkstra

```
Entrée : Un graphe pondéré G et un sommet initial s
Sortie : La distance minimale entre le sommet initial s et les autres sommets

# initialisation
distance[u] = ∞ pour tout sommet u du graphe
distance[s] = 0
Marquer tous sommets comme non-visités

# boucle principale
Tant qu'il reste des sommets à visiter :
    # sélection du sommet non-visité le plus proche de s
    Déterminer le sommet non-visité u dont distance[u] est minimale
    Marquer u comme visité

    # mise à jour de son voisinage
    Pour chaque voisin v non-visité de u :
        d = distance[u] + poids(u, v)
        Si d < distance[v]:
            distance[v] = d

Retourner distance
```

Sélection du sommet le plus proche

Relaxation d'arc

? Implémenter en Python l'algorithme de Dijkstra ! Modifier ensuite l'algorithme pour retenir les prédécesseurs.