Ayudantía 8

Dijkstra

BFS

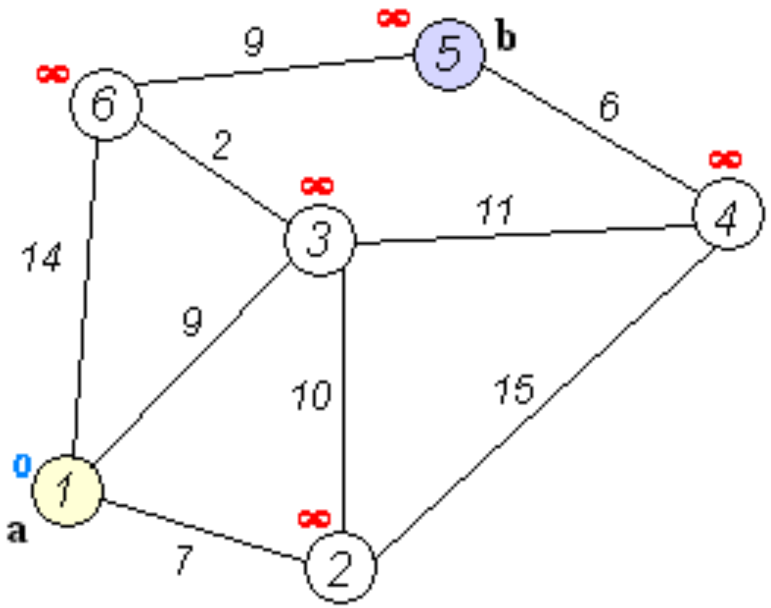
```
Dijkstra(s): —s es el vértice de partida
   for each u in V:
        u.color = white; d[u] = ∞; π[u] = null
   Q = cola
   s.color = gray; d[s] = 0; Q.enqueue(s)
   while !Q.empty():
       u = Q.dequeue()
       for each v in α[u]:
           if v.color == white or v.color == gray:
               if d[v] > d[u]+costo(u,v):
                   d[v] = d[u]+costo(u,v); π[v] = u
               if v.color == white:
                   v.color = gray; Q.enqueue(v)
       u.color = black
```

```
Dijkstra(s):  —s es el vértice de partida
    for each u in V:
         u.color = white; d[u] = ∞; π[u] = null
    Q = cola
    s.c
    whi
```

Dijkstra toma tiempo O((*V*+*E*) log*V*)

Se hacen V inserciones, E-V actualizaciones, y V extracciones

```
                    d[v] = d[u]+costo(u,v); π[v] = u
                if v.color == white:
                    v.color = gray; Q.enqueue(v)
        u.color = black
```



∞
∞
0
∞
∞
∞
1
2
3
4
5
6
a
b
7
9
14
10
15
11
2
9
6

Día de playa. Mediodía con 40°C. El índice UV es ALTO y PELIGROSO para los seres humanos debido al cambio climático. Aparte del bloqueador solar, tu única protección son los quitasoles repartidos por la playa. Asumiendo que cada quitasol proyecta una sombra perfectamente circular, y conoces el centro de cada uno de estos círculos.

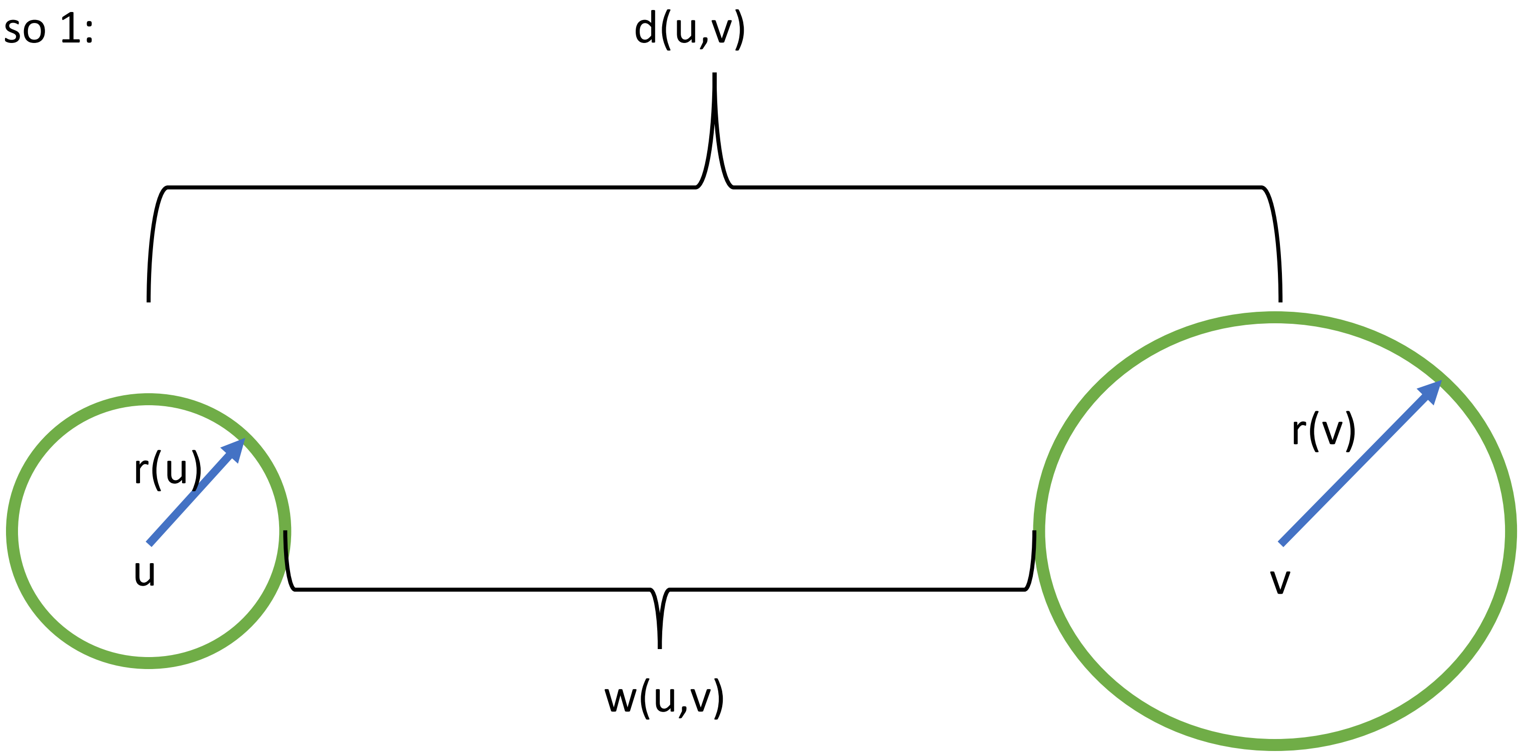
Encuentra el camino desde el quitasol B a los demás quitasoles de la playa, que minimice la exposición al sol.

Propiedades del problema:

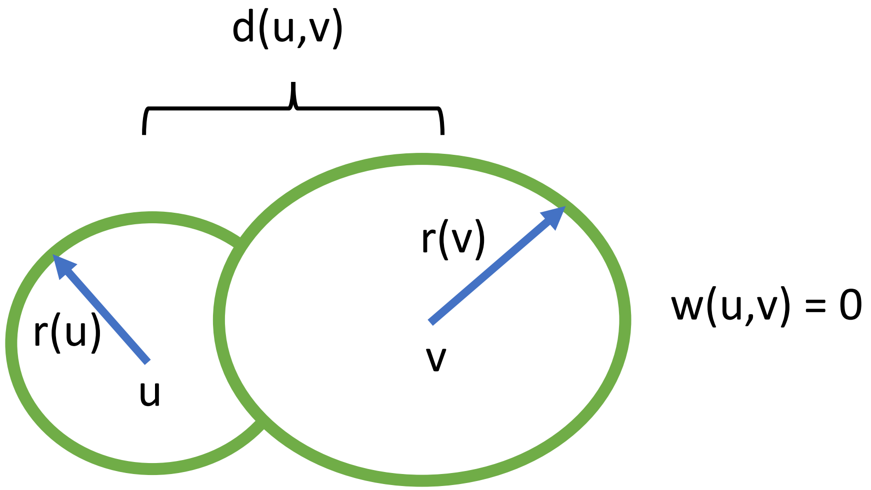
Se define w(u,v) como la cantidad de sol a la que te expones si caminas del nodo u al nodo v.

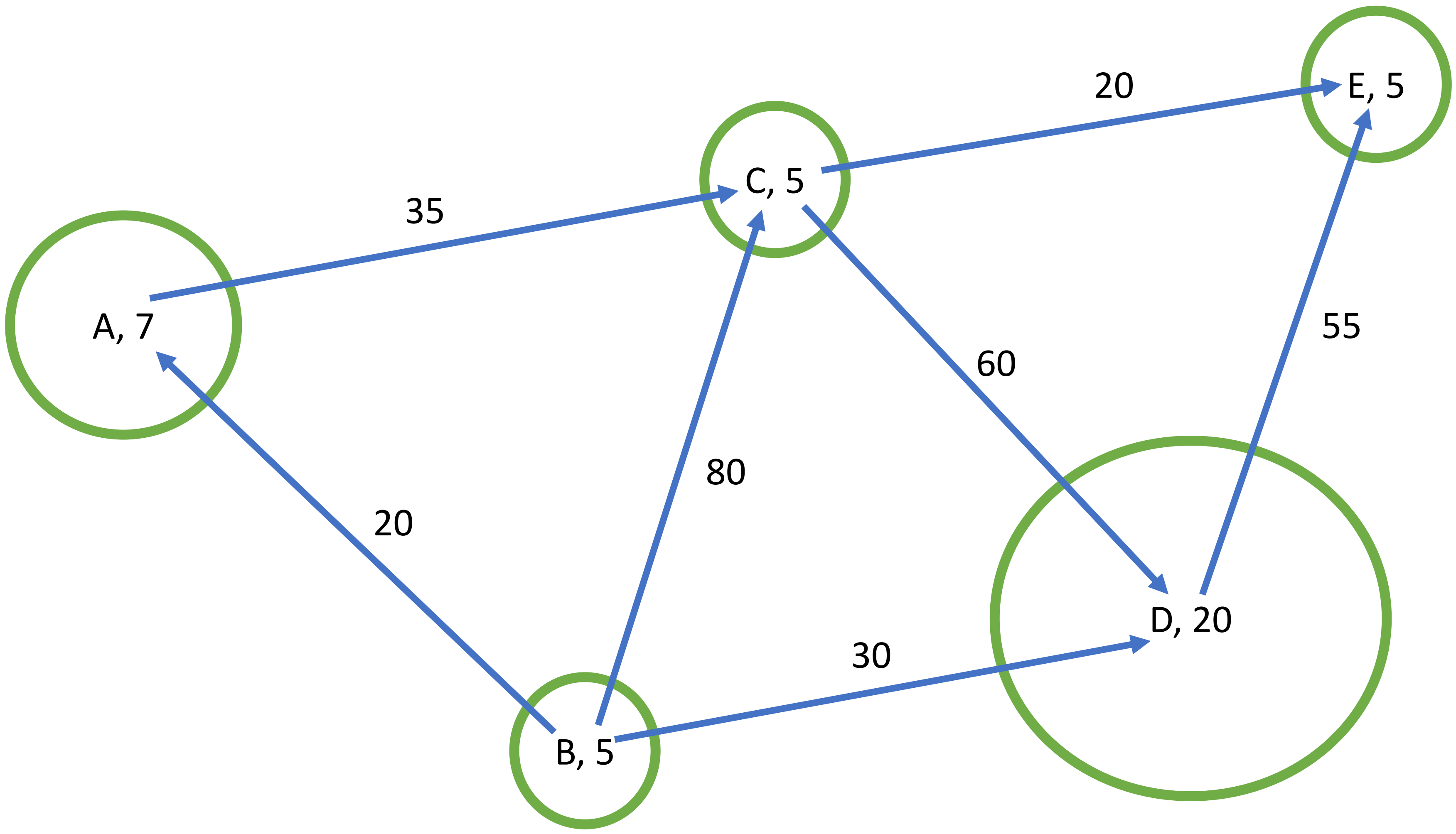
$$w(u,v) = \begin{cases} d(u,v) - r(u) - r(v) & si\!: r(u) + r(v) < d(u,v) \\ 0 & e.o.c. \end{cases}$$

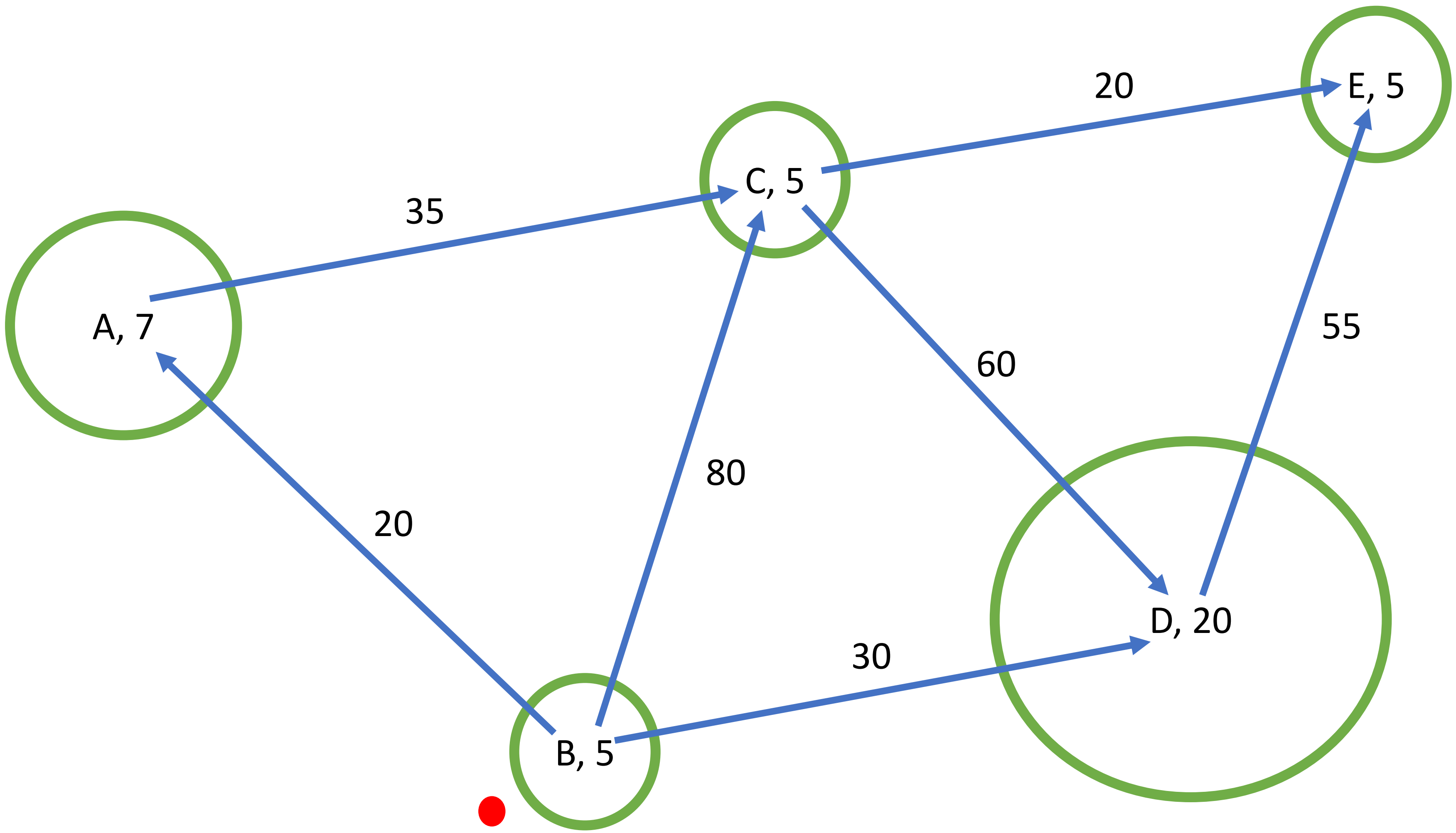
Caso 1:



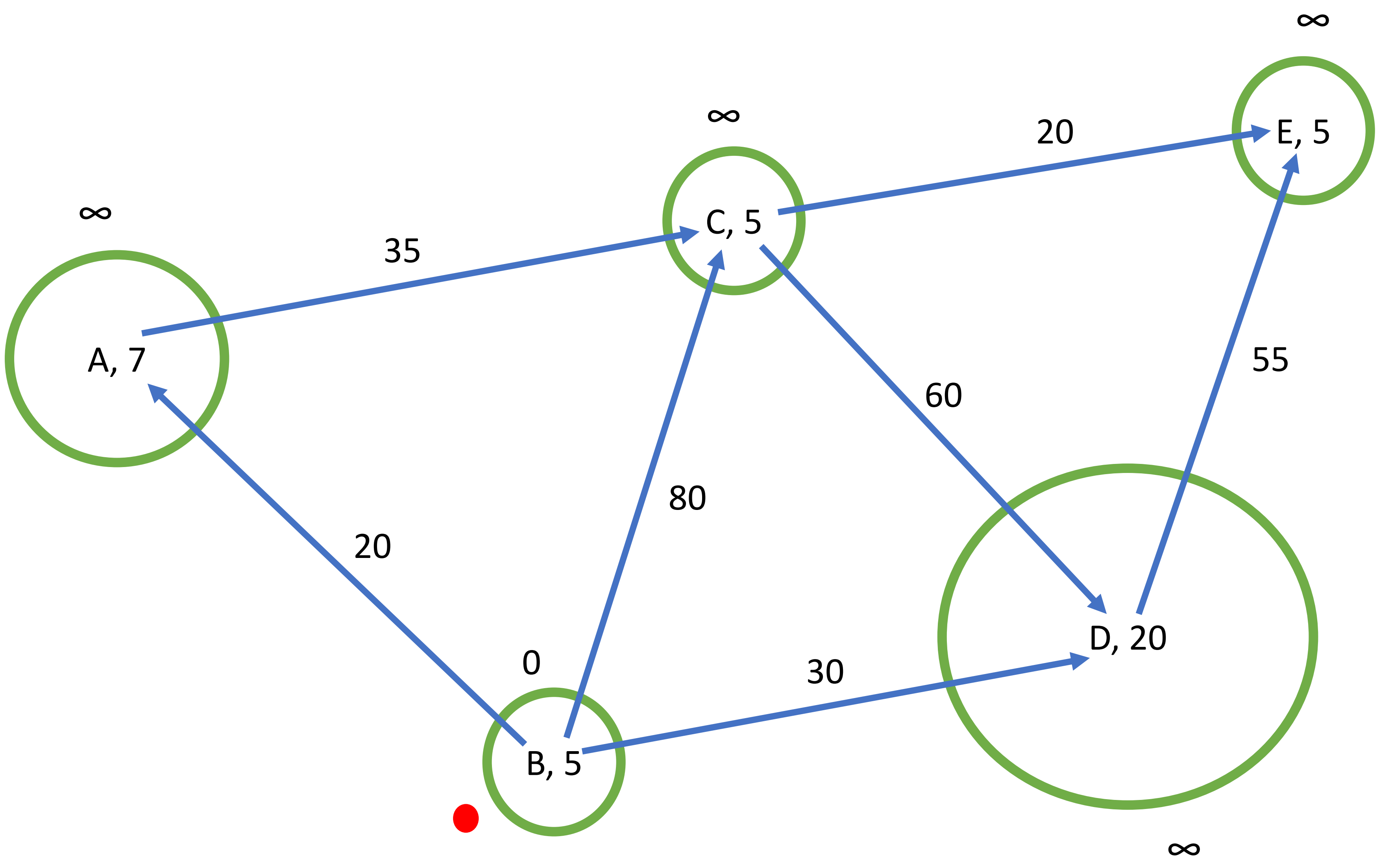
Caso 2:

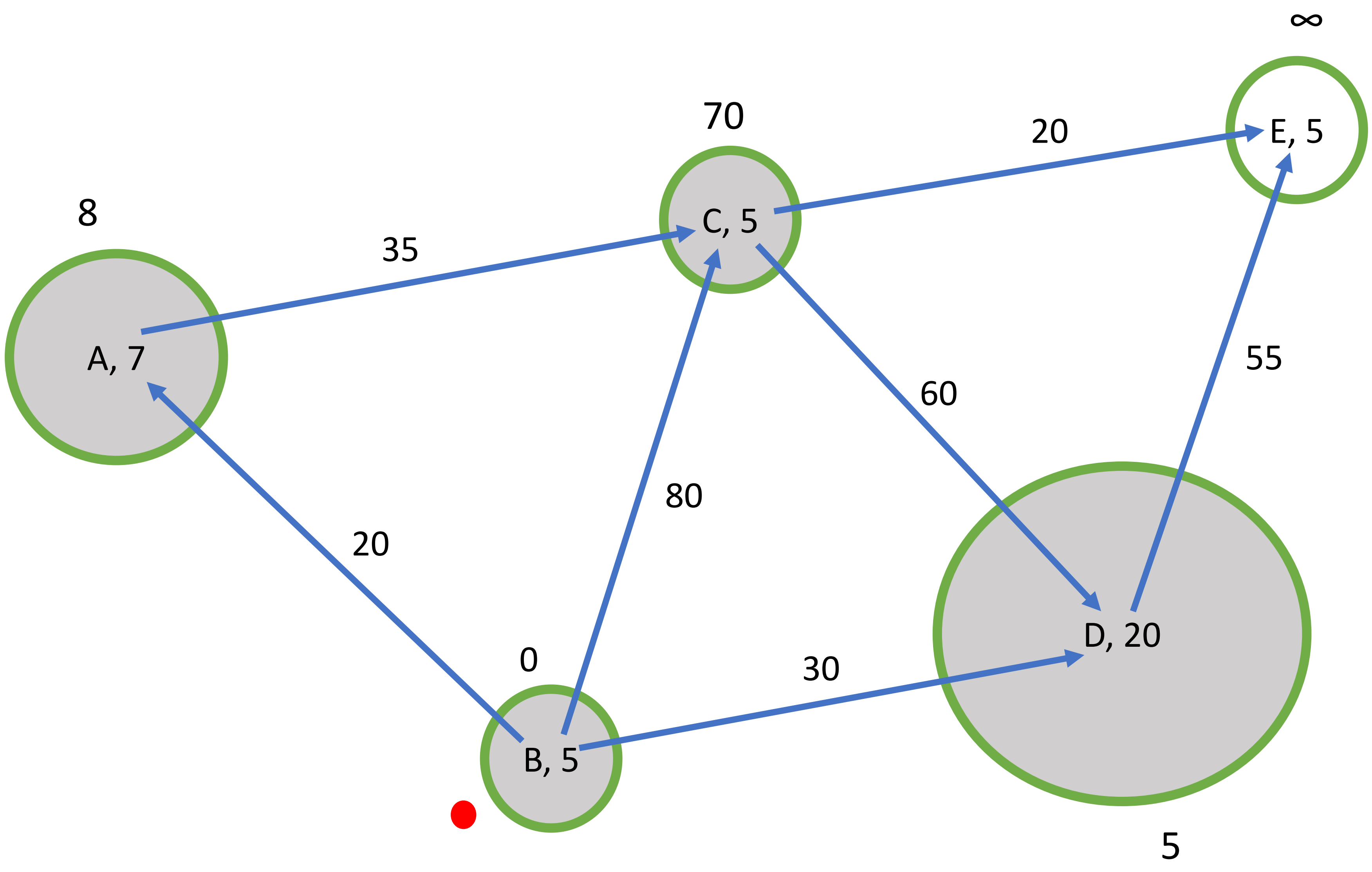



E, 5
20
C, 5
35
A, 7
60
55
80
20
D, 20
30
B, 5

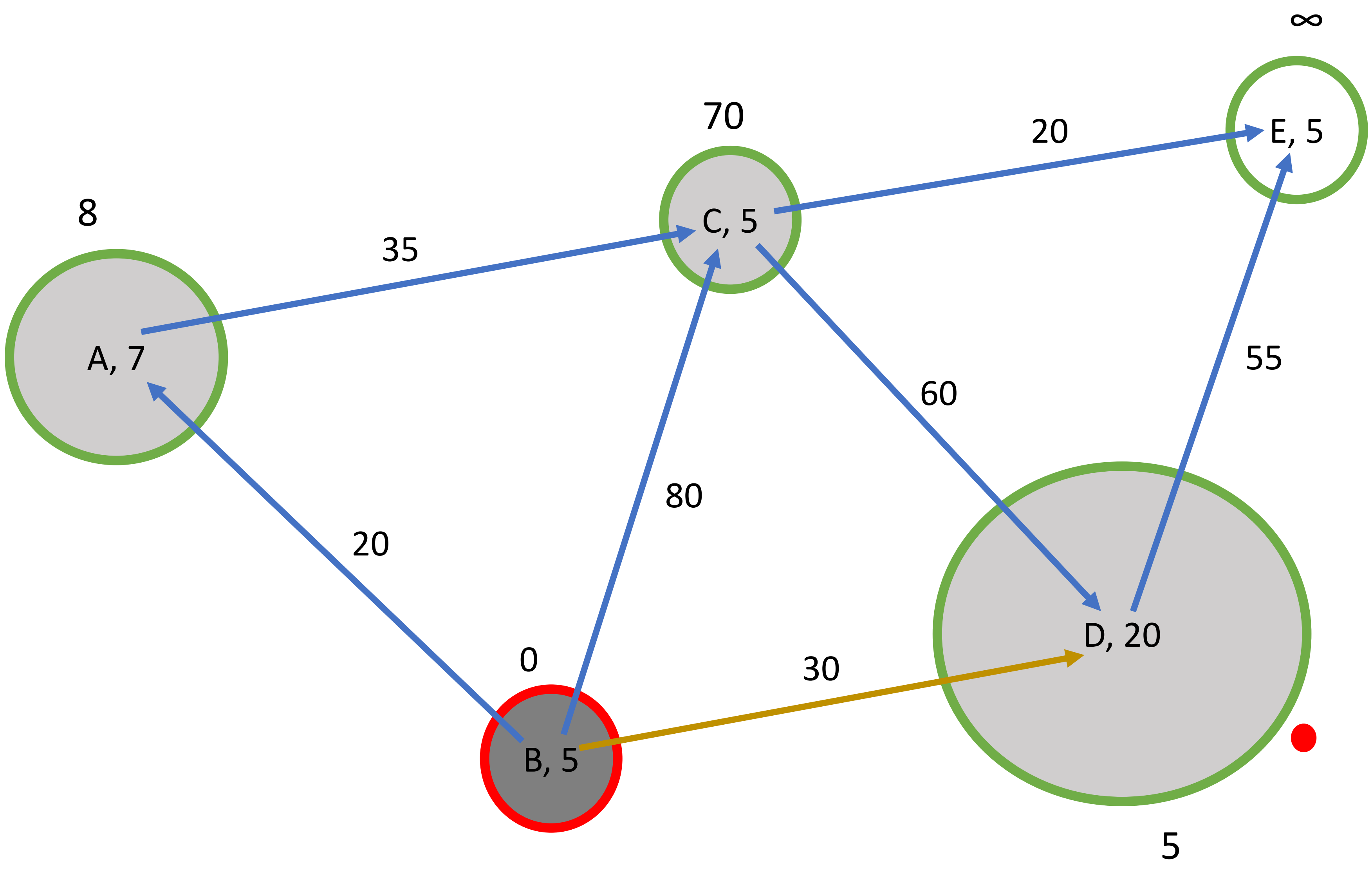


A, 7
B, 5
C, 5
D, 20
E, 5
35
20
80
60
20
55
30

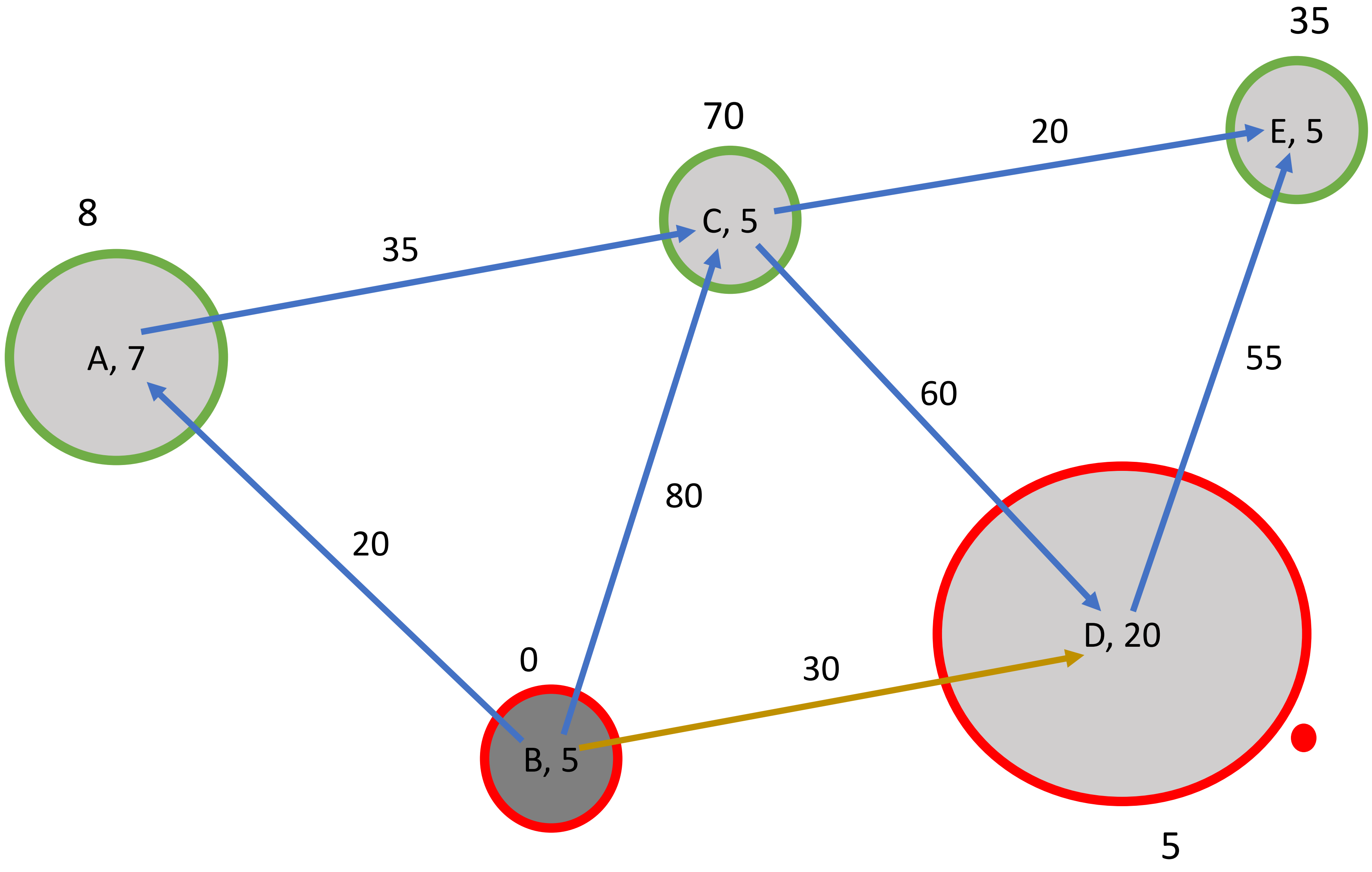

∞
E, 5
20
∞
C, 5
∞
35
A, 7
60
55
80
20
D, 20
0
30
B, 5
∞

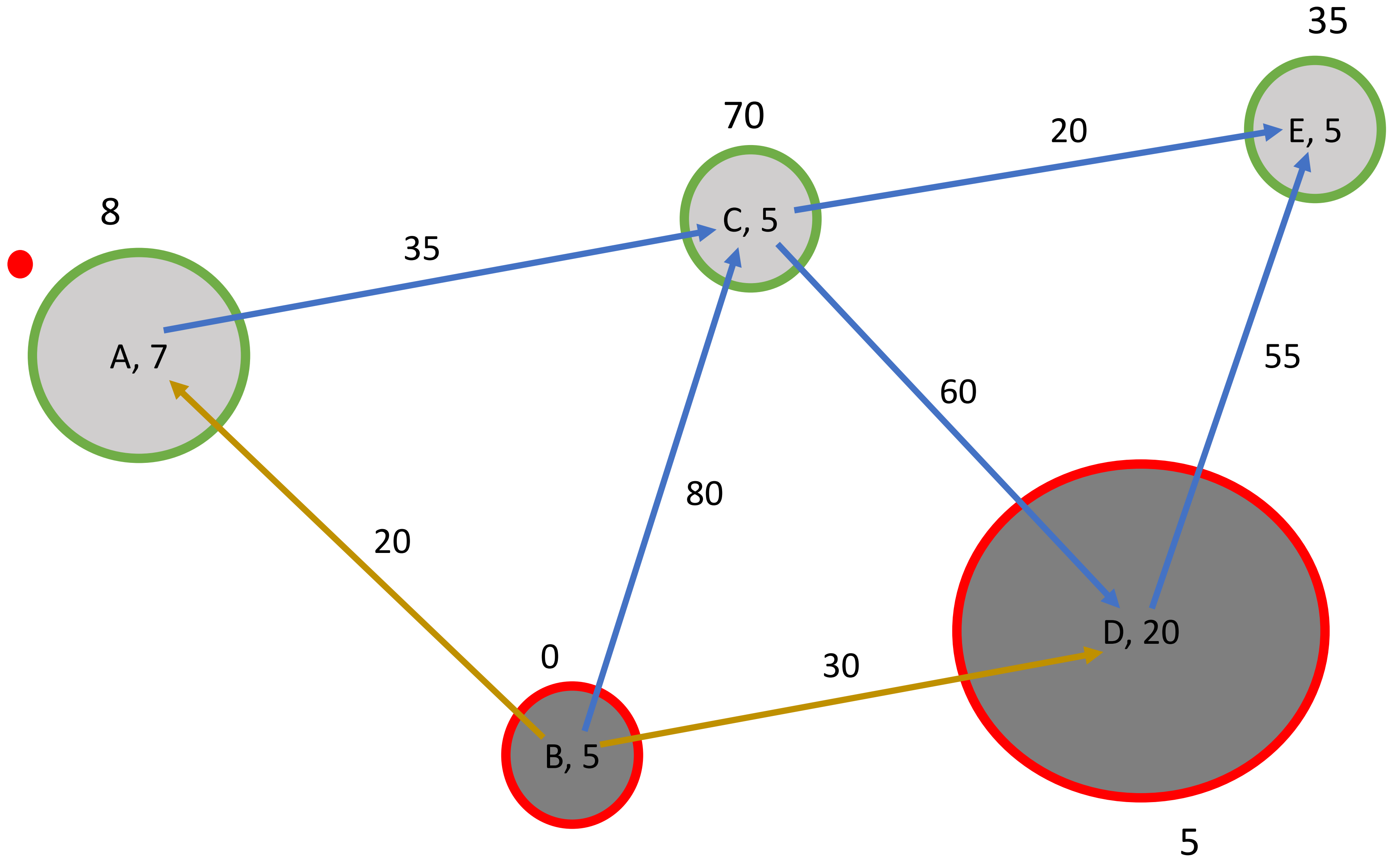


∞
70
20
E, 5
8
C, 5
35
A, 7
60
55
80
20
D, 20
0
30
B, 5
5

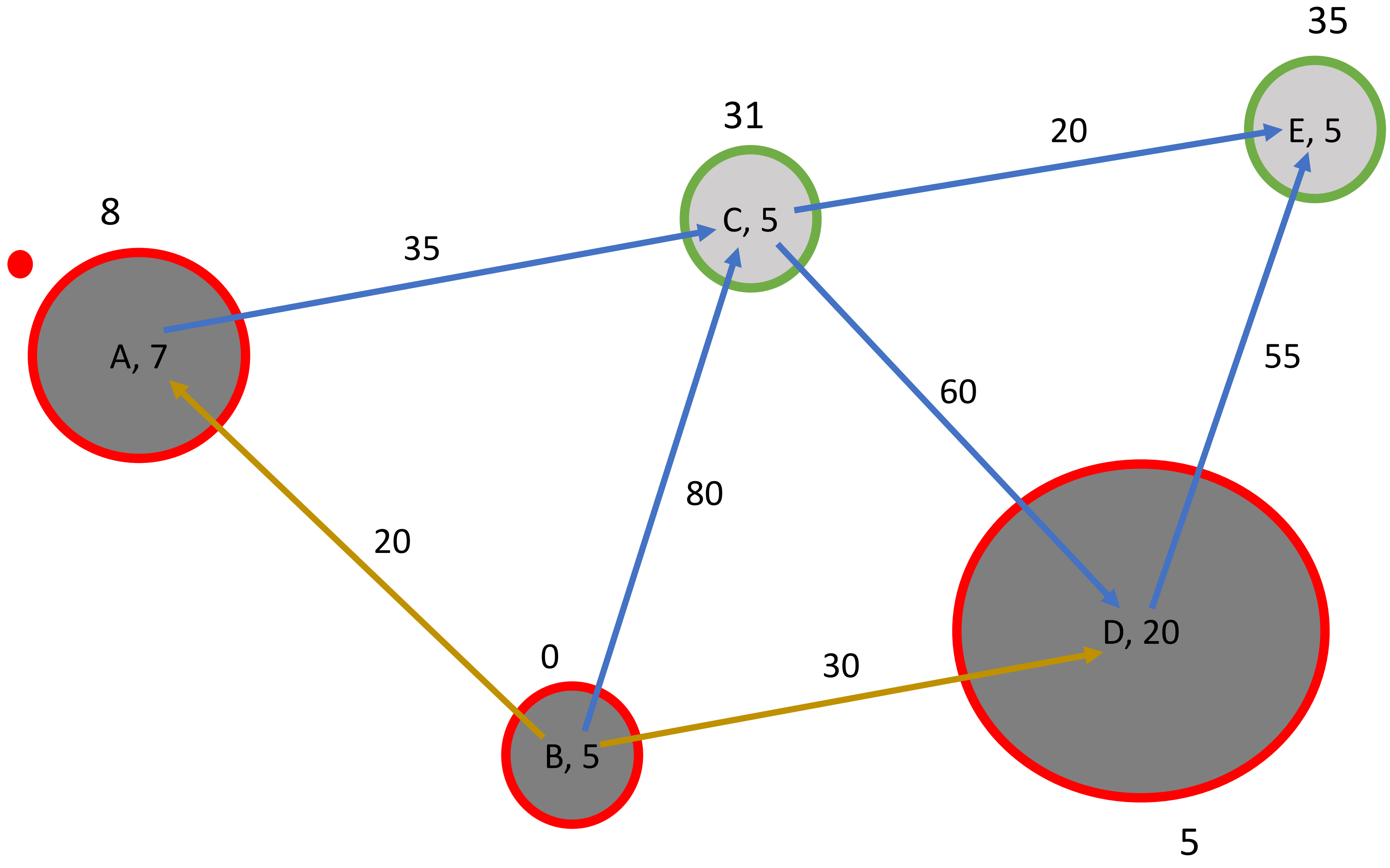


∞
70
20
E, 5
8
35
C, 5
A, 7
55
60
80
20
D, 20
0
30
B, 5
5

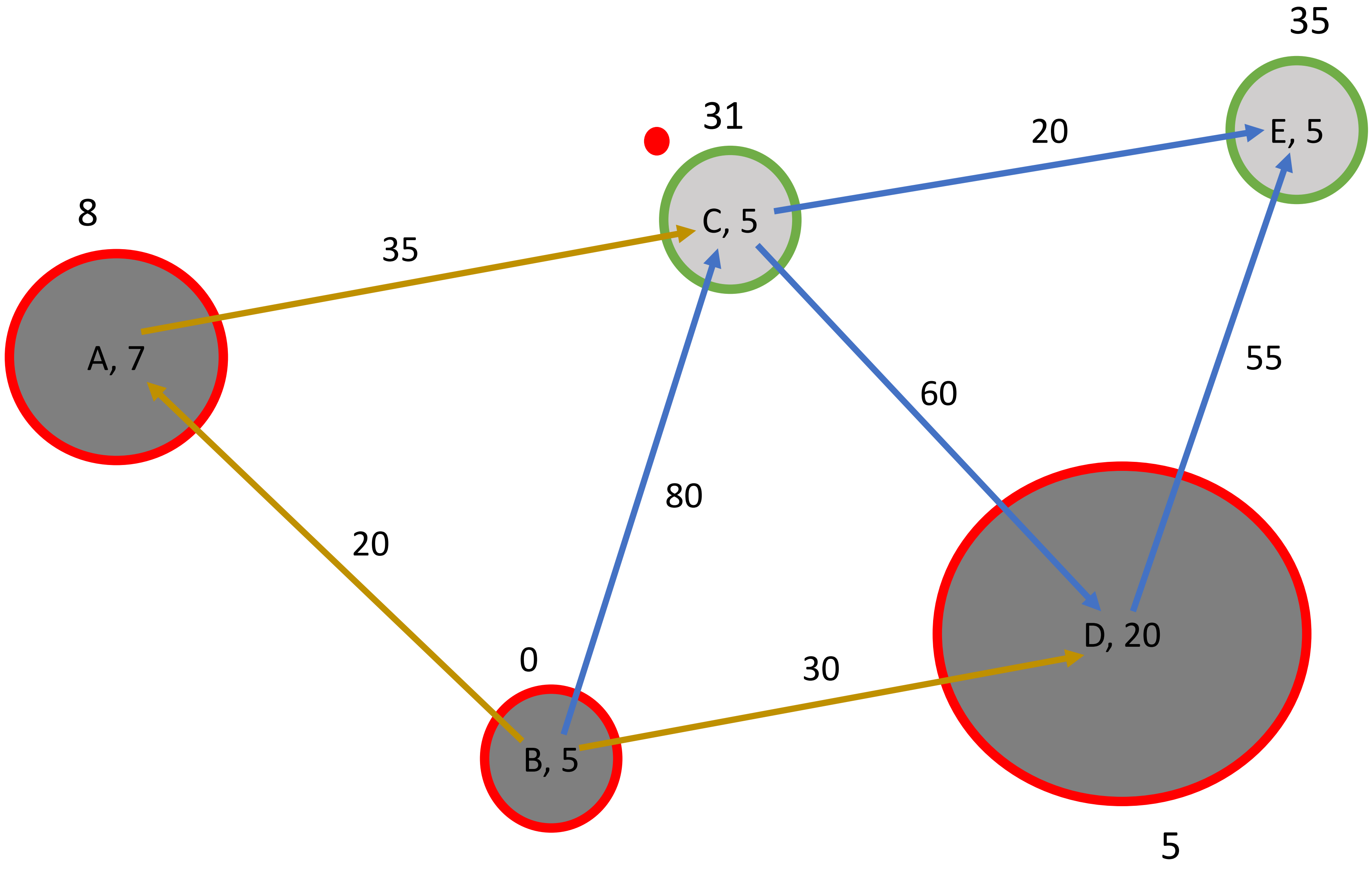

35
70
20
E, 5
8
35
C, 5
A, 7
60
55
80
20
0
30
D, 20
B, 5
5

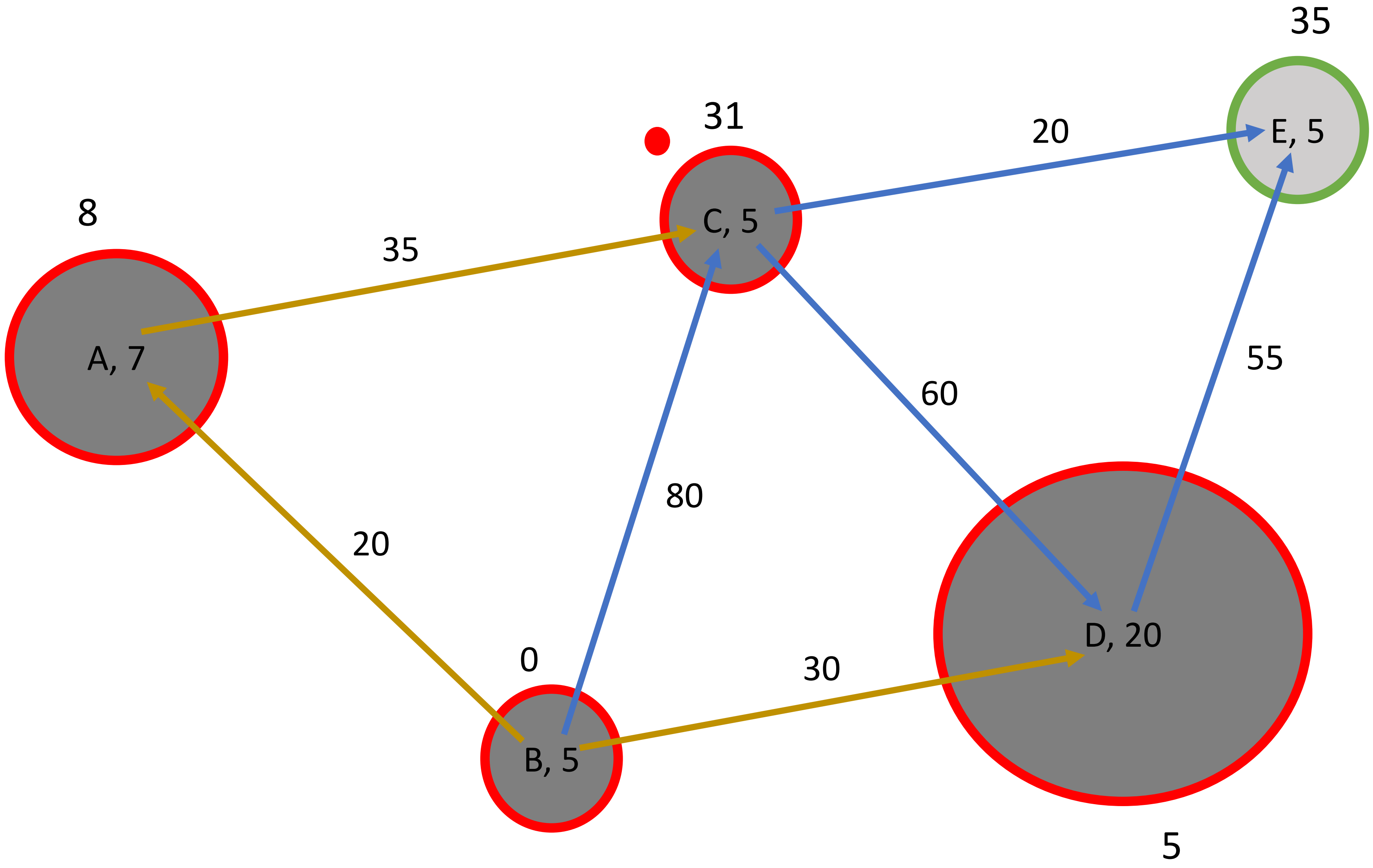


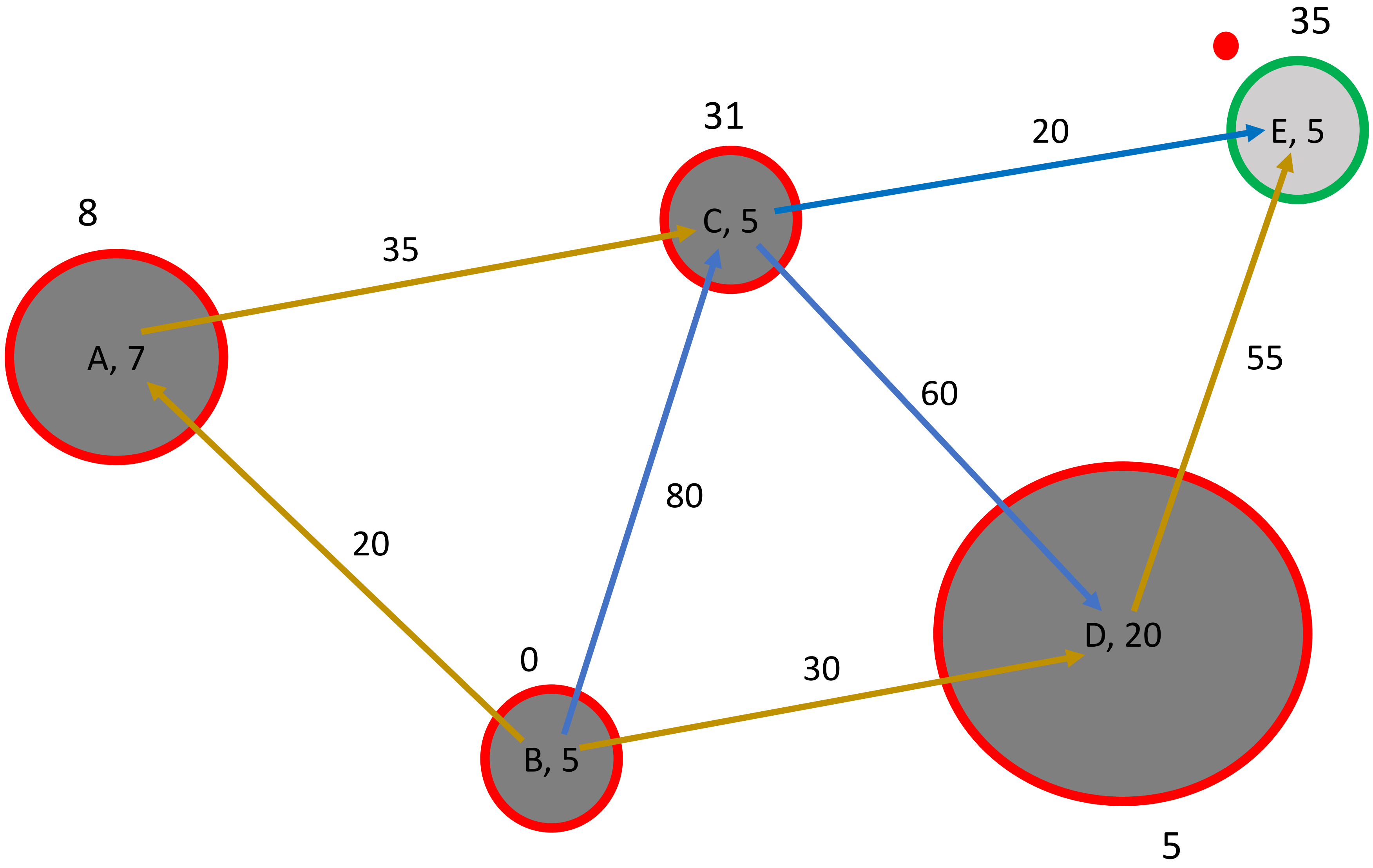
35
70
20
E, 5
8
35
C, 5
A, 7
60
55
80
20
0
30
D, 20
B, 5
5

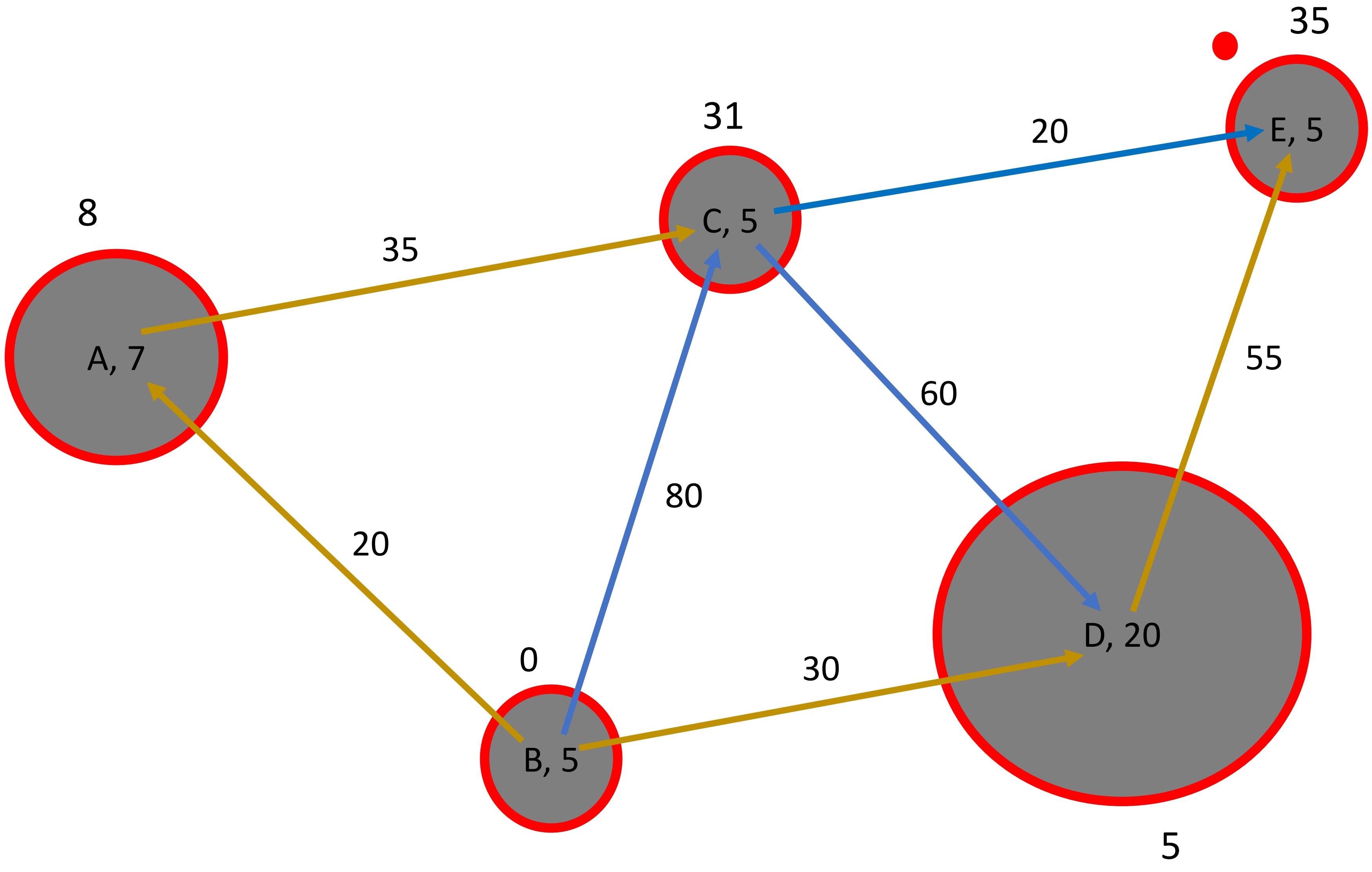


35
31
20
E, 5
8
35
C, 5
A, 7
60
55
80
20
0
30
D, 20
B, 5
5

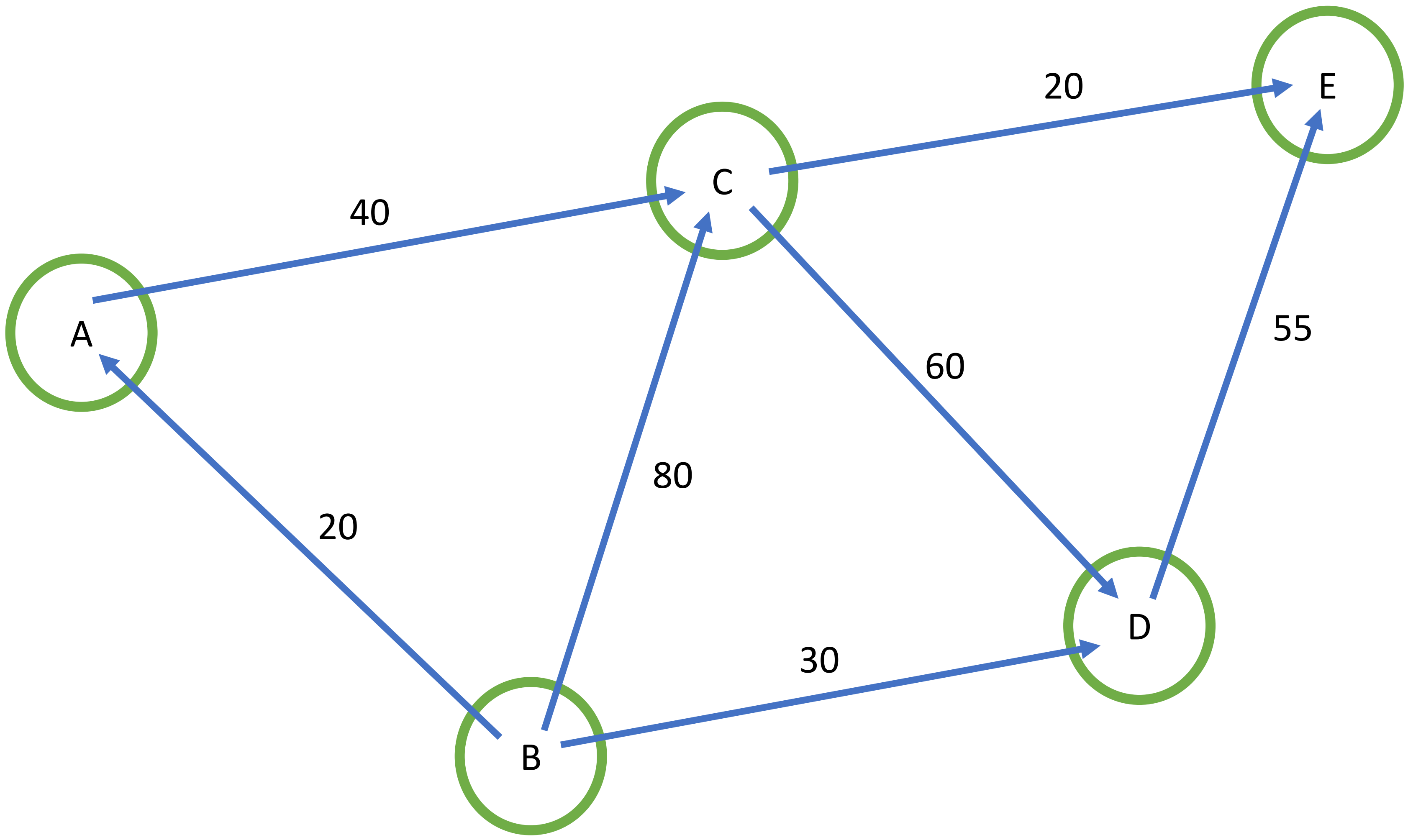

35
31
20
E, 5
8
C, 5
35
A, 7
60
55
80
20
0
D, 20
30
B, 5
5


35
31
20
E, 5
8
35
C, 5
A, 7
60
55
80
20
0
30
D, 20
B, 5
5

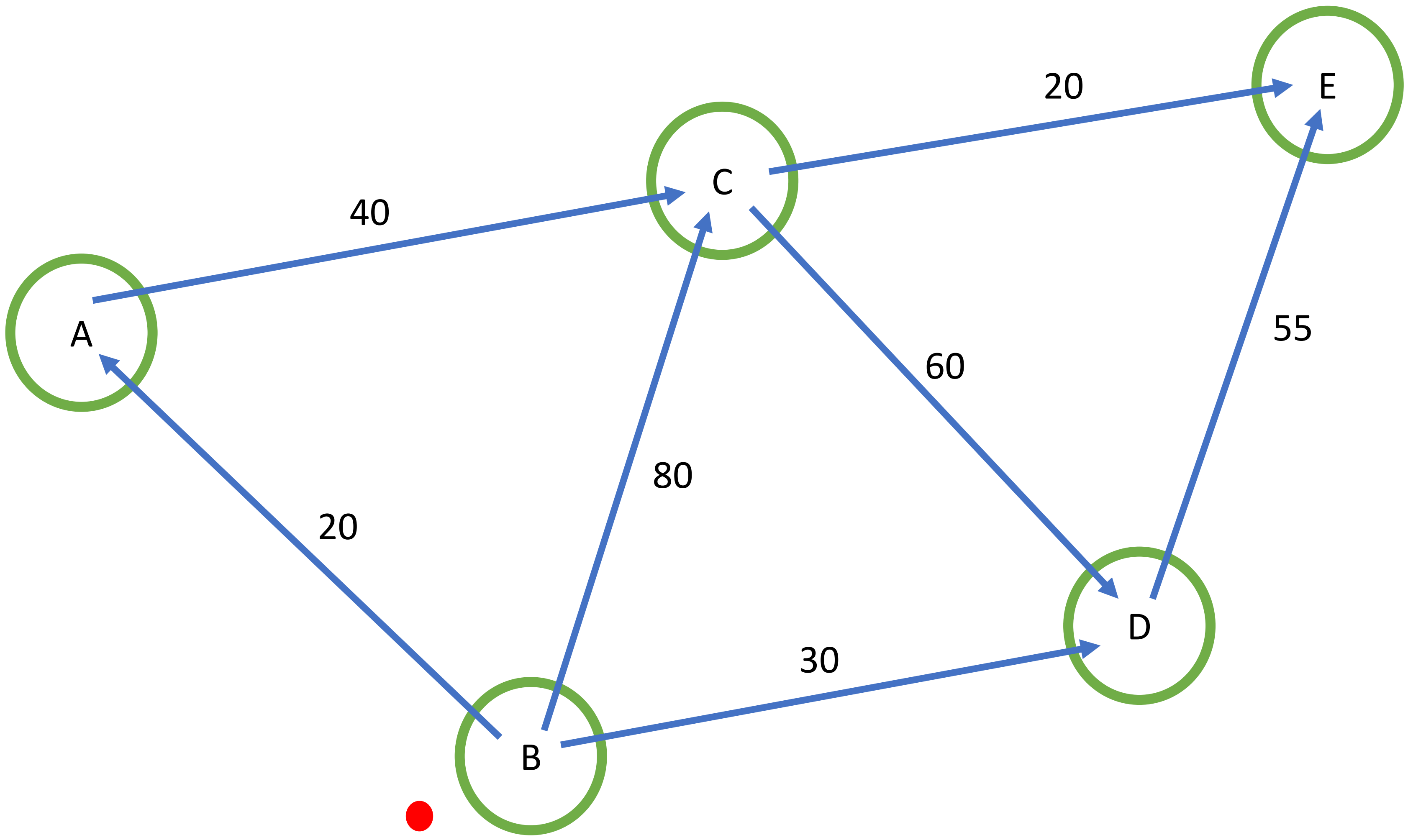

35
31
20
E, 5
8
35
C, 5
A, 7
60
55
80
20
0
30
D, 20
B, 5
5


35
31
20
E, 5
8
35
C, 5
A, 7
60
55
80
20
0
30
D, 20
B, 5
5

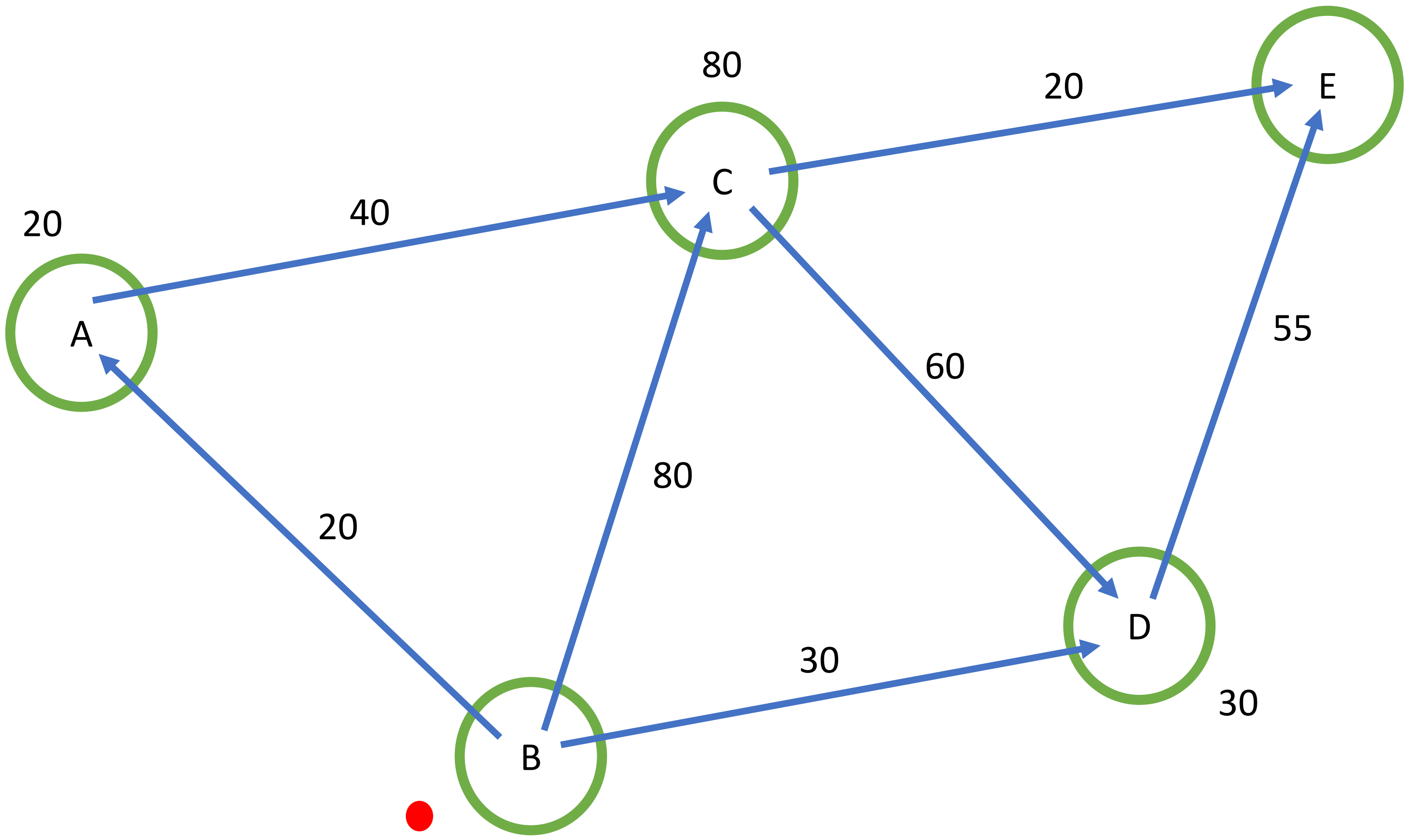
Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla



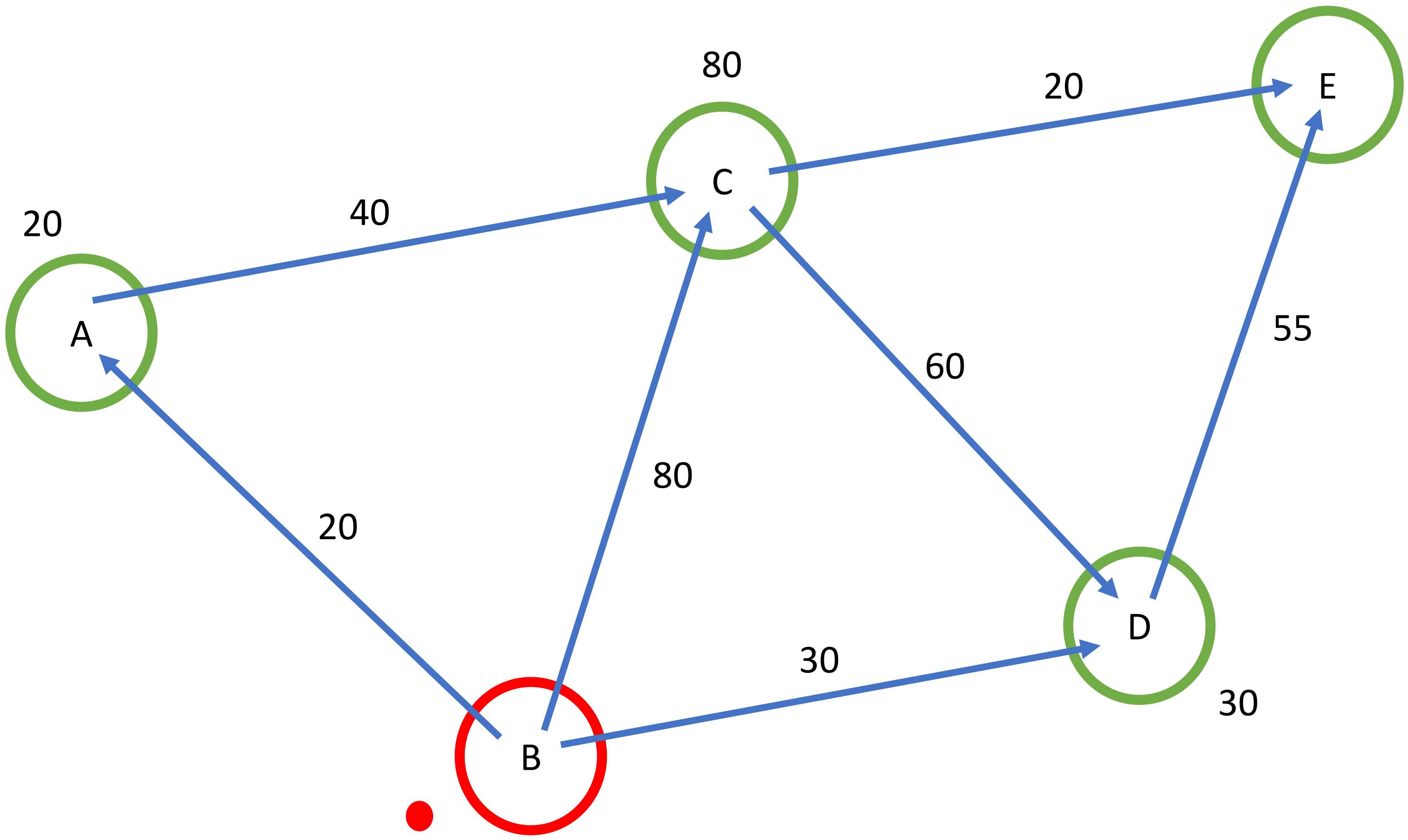
Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla



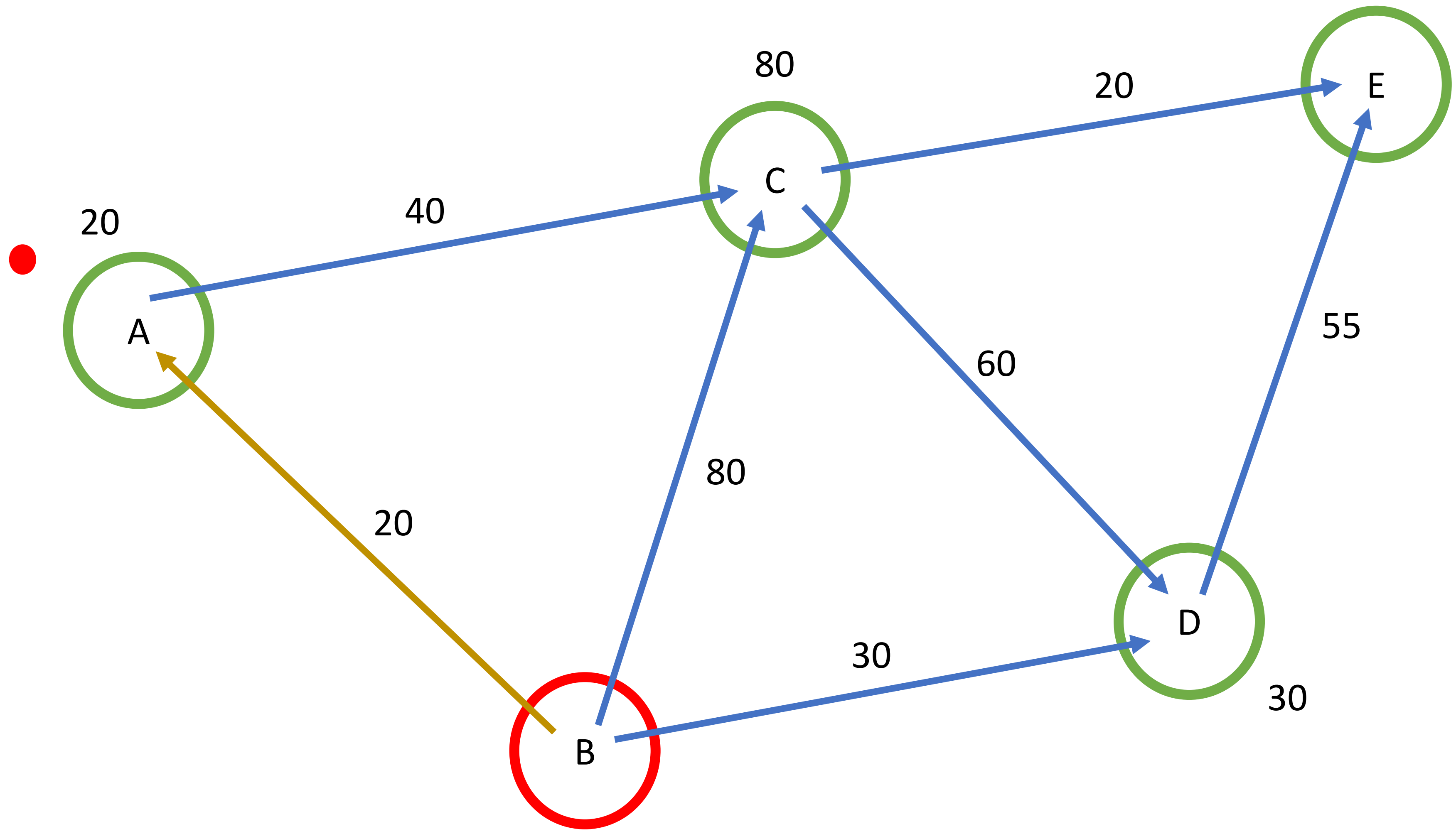
Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla



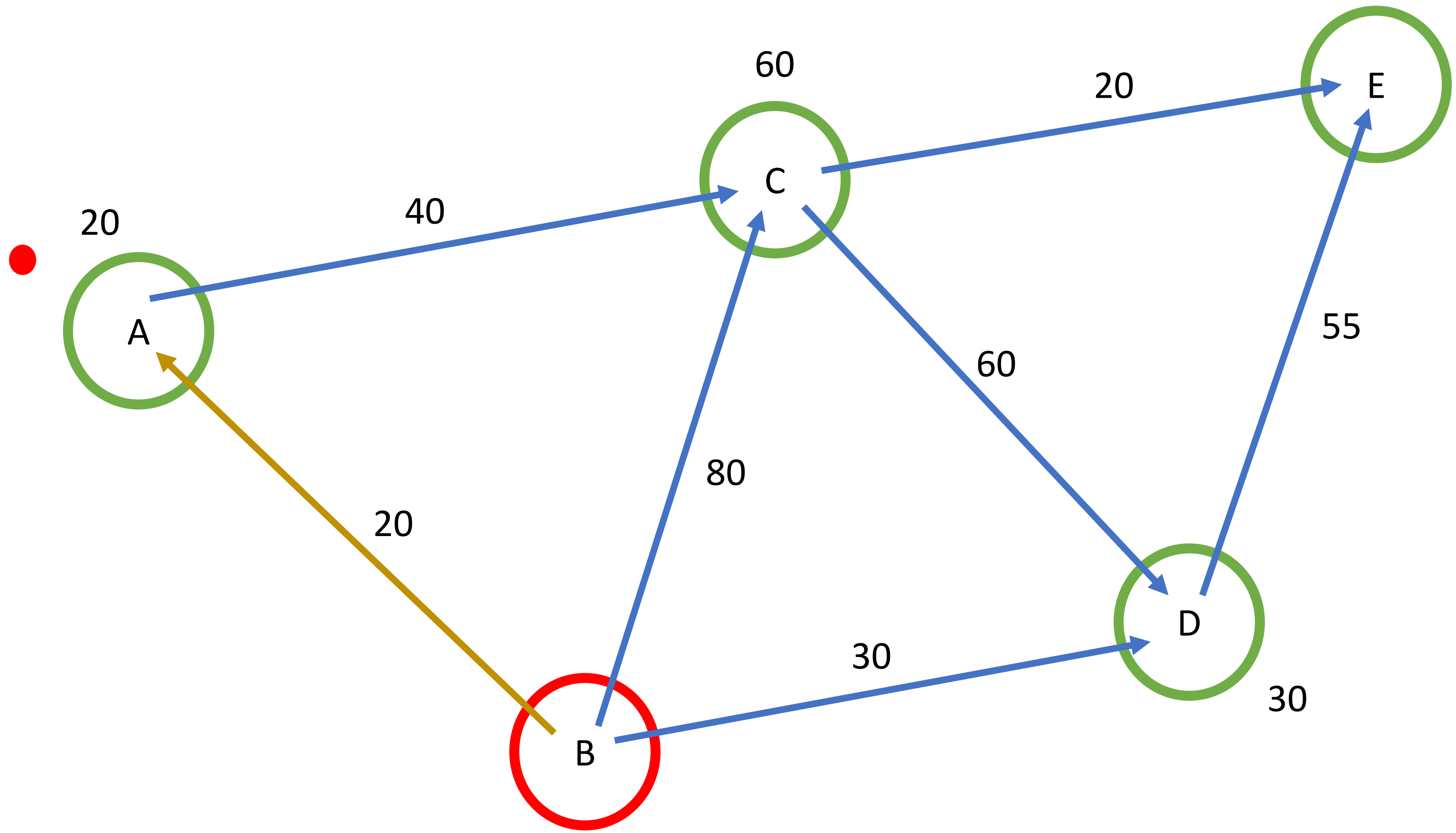
Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla



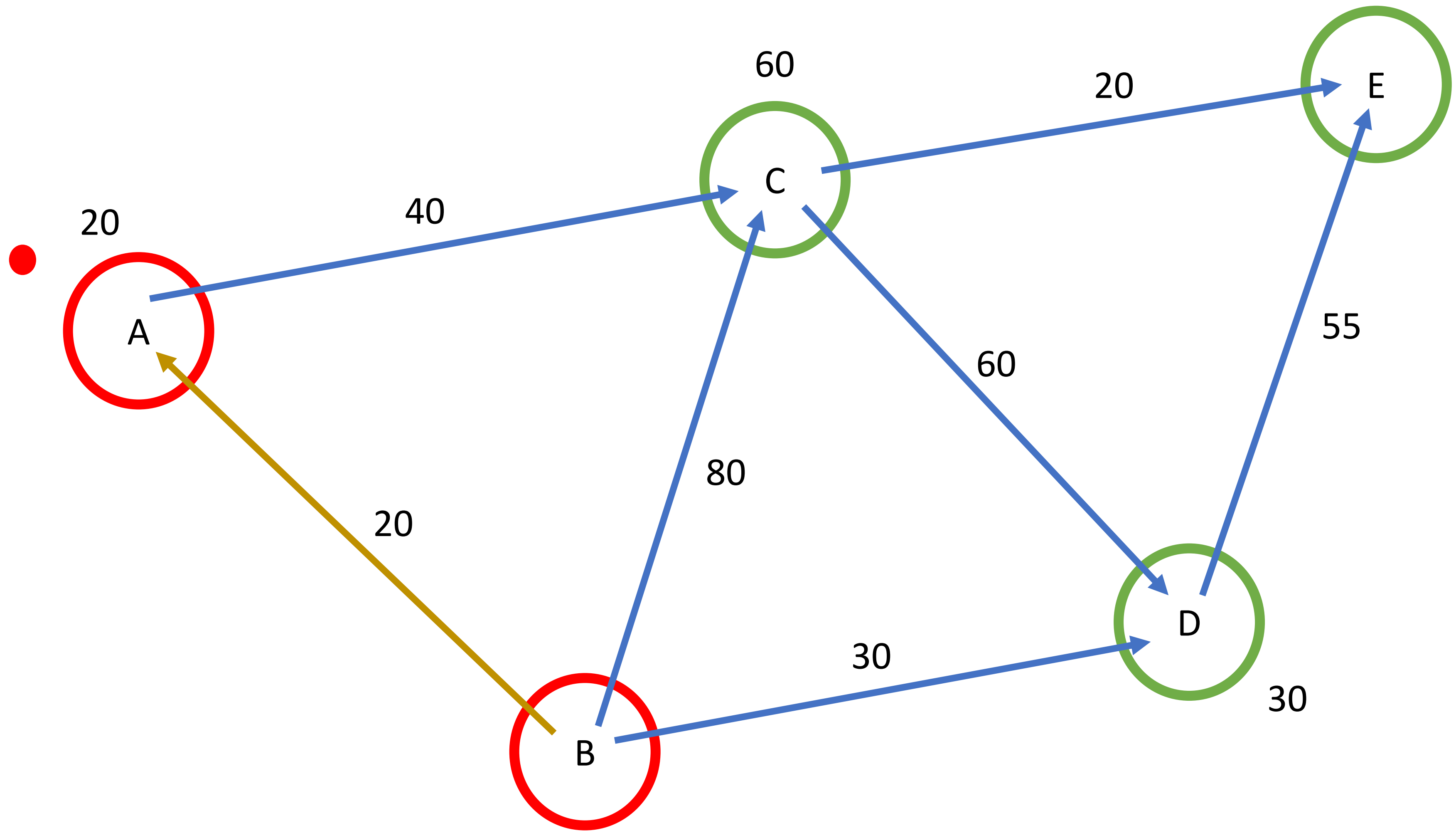
Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla



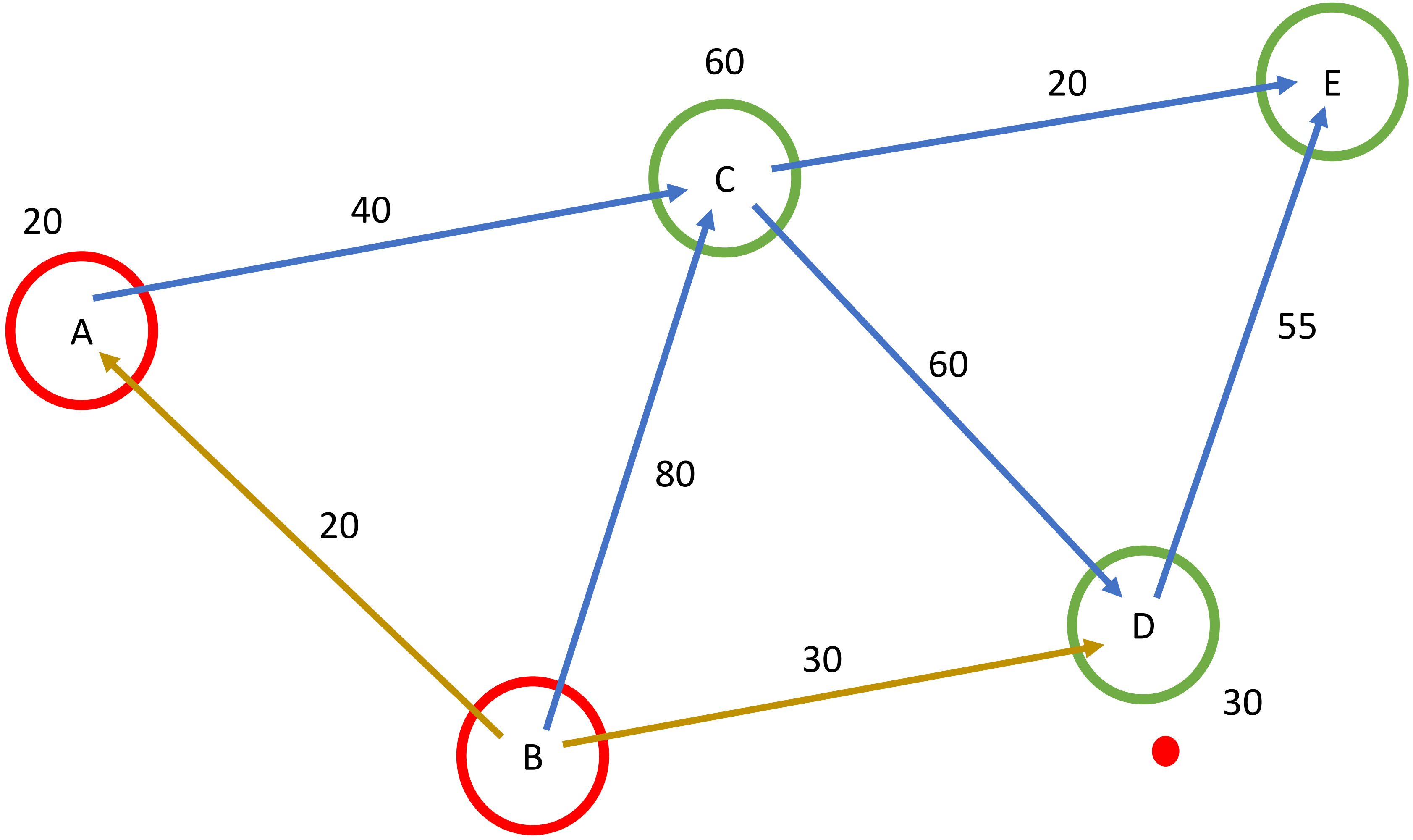
Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla



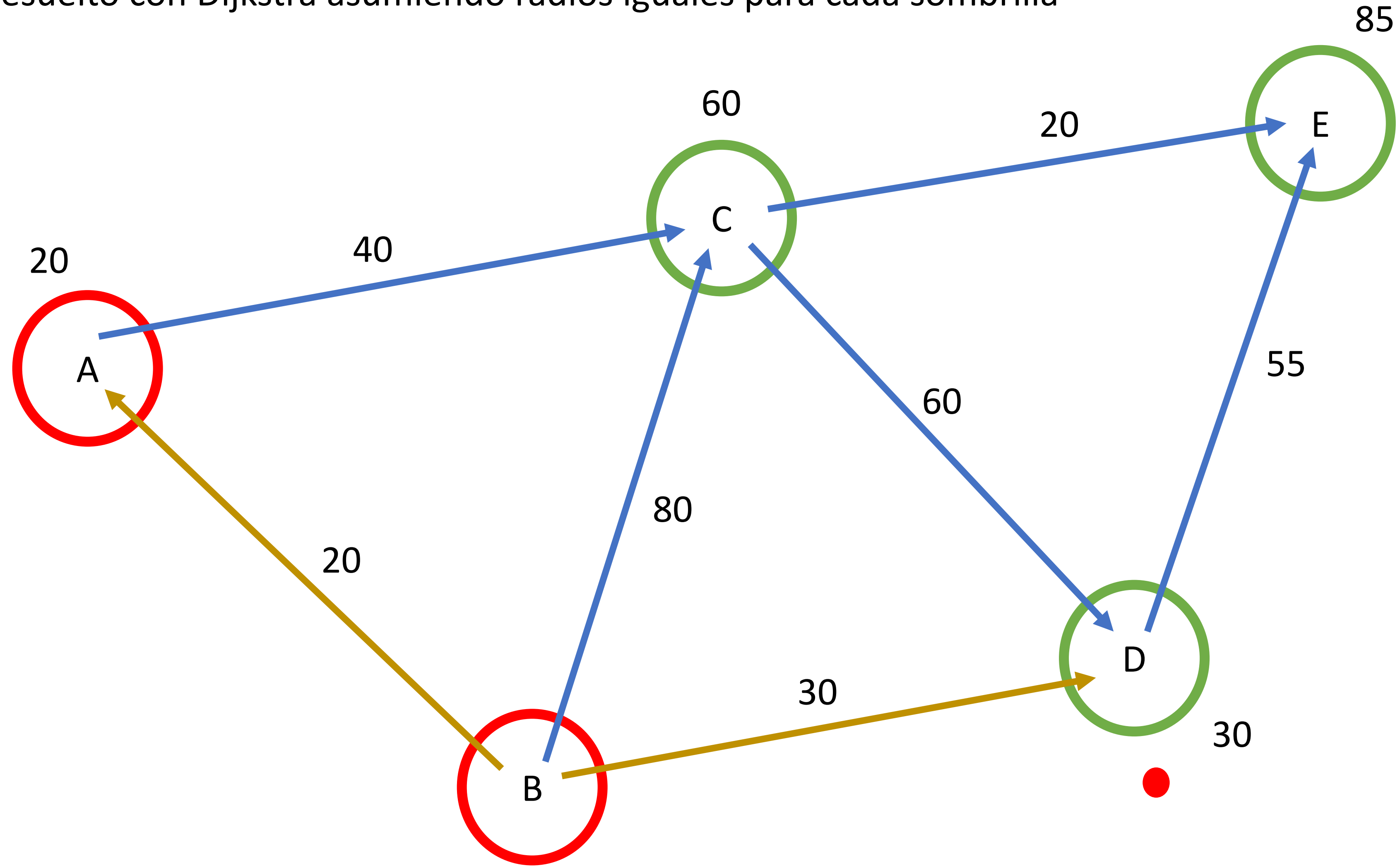
Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla



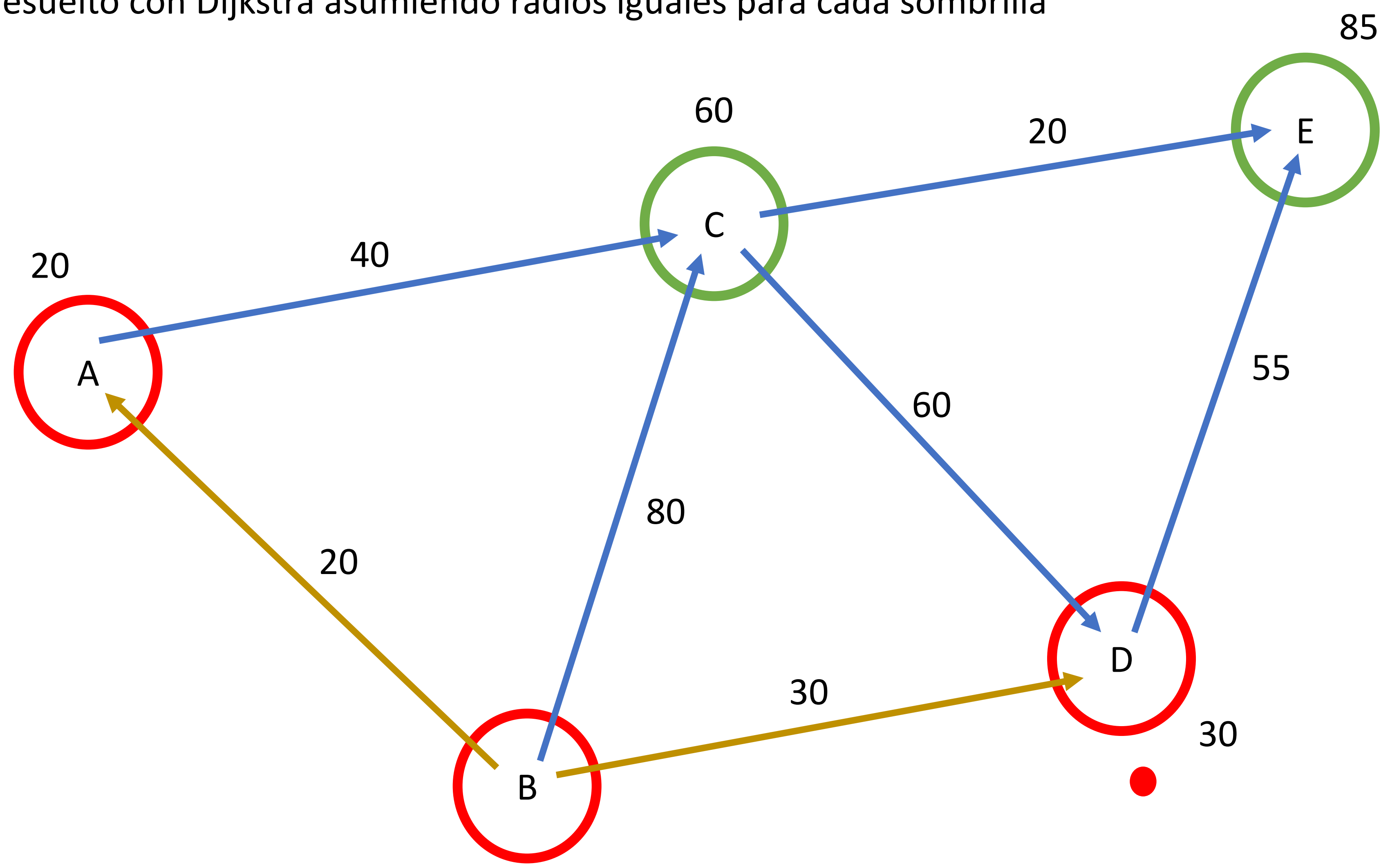
Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla



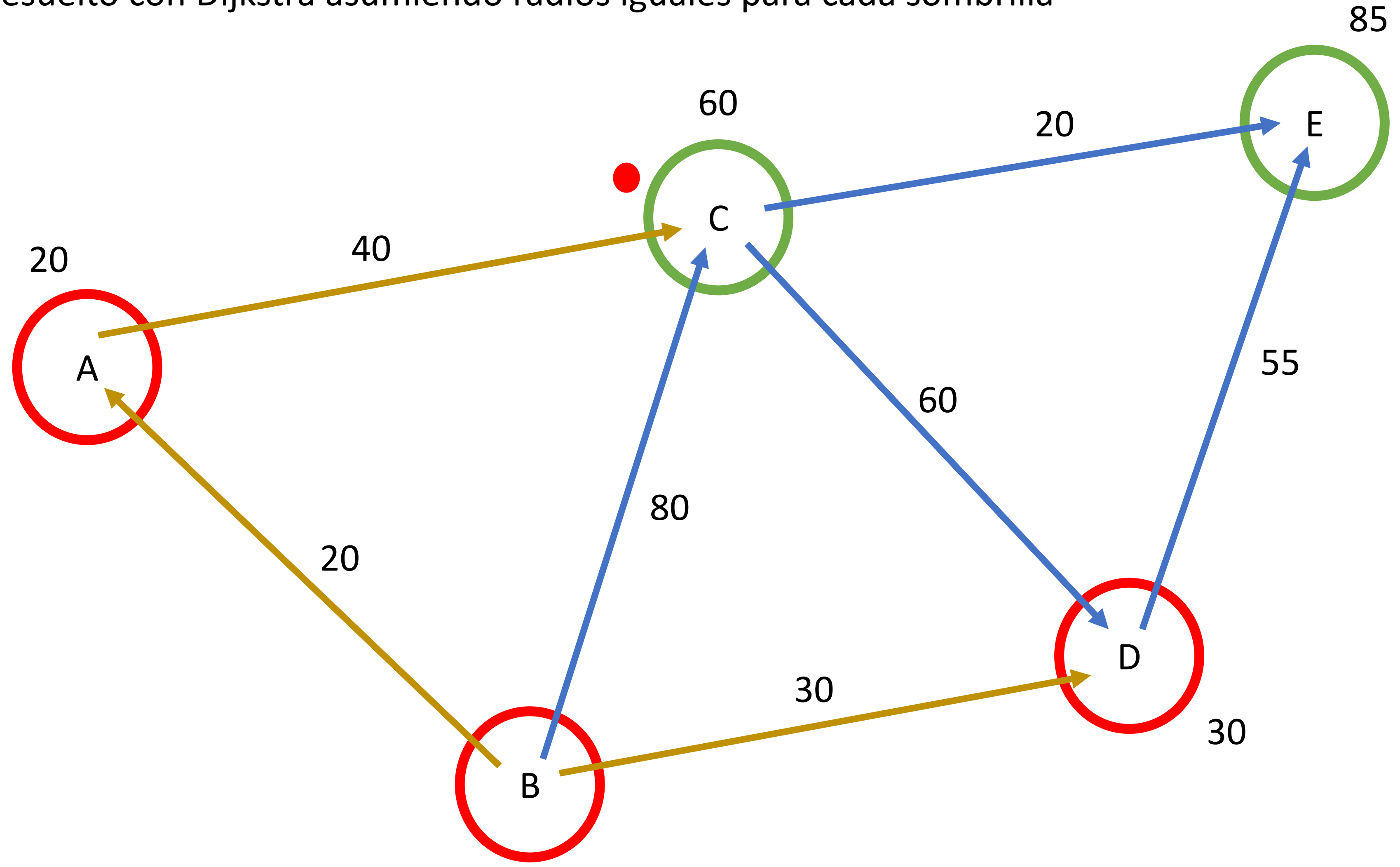
Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla



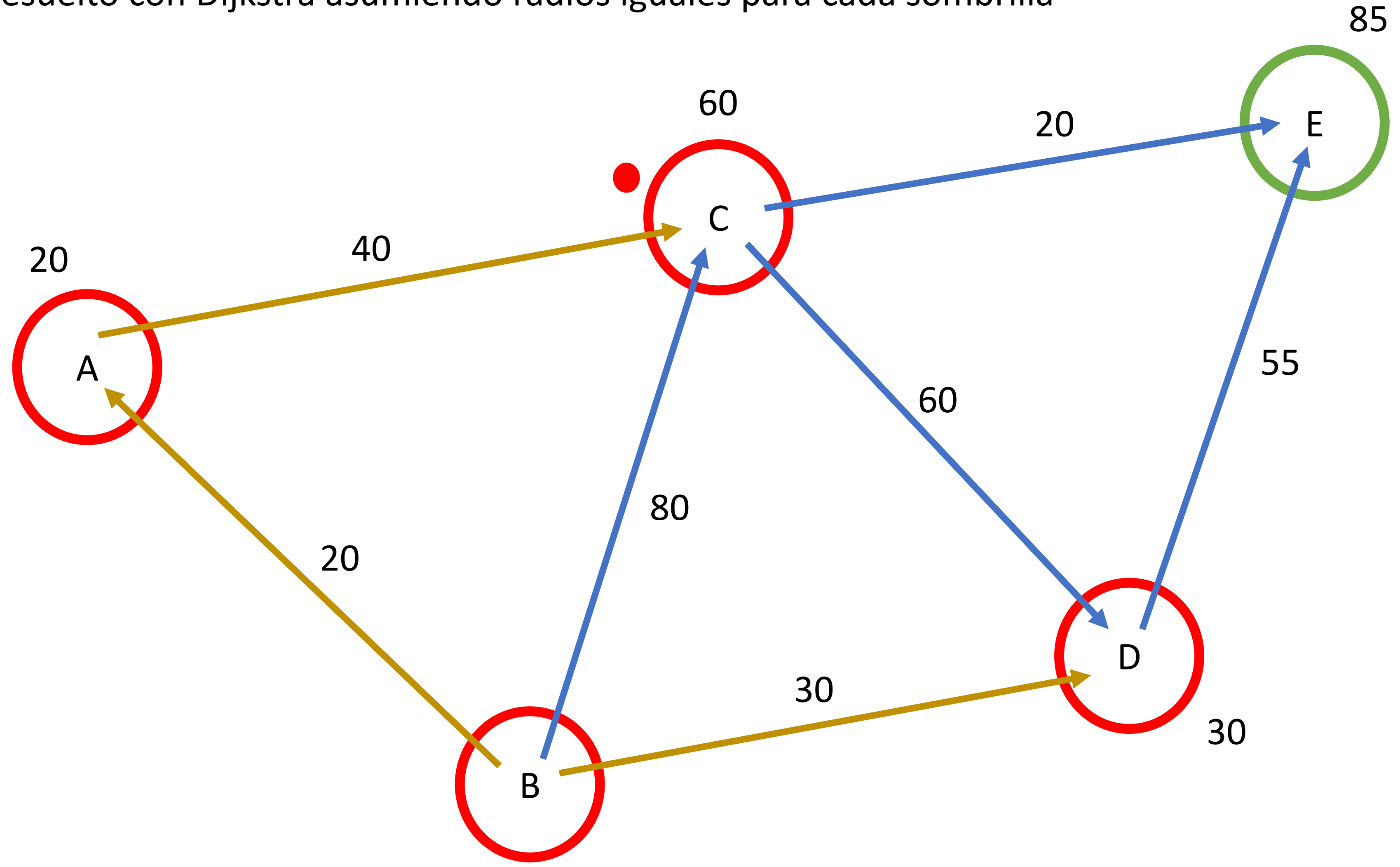
Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla



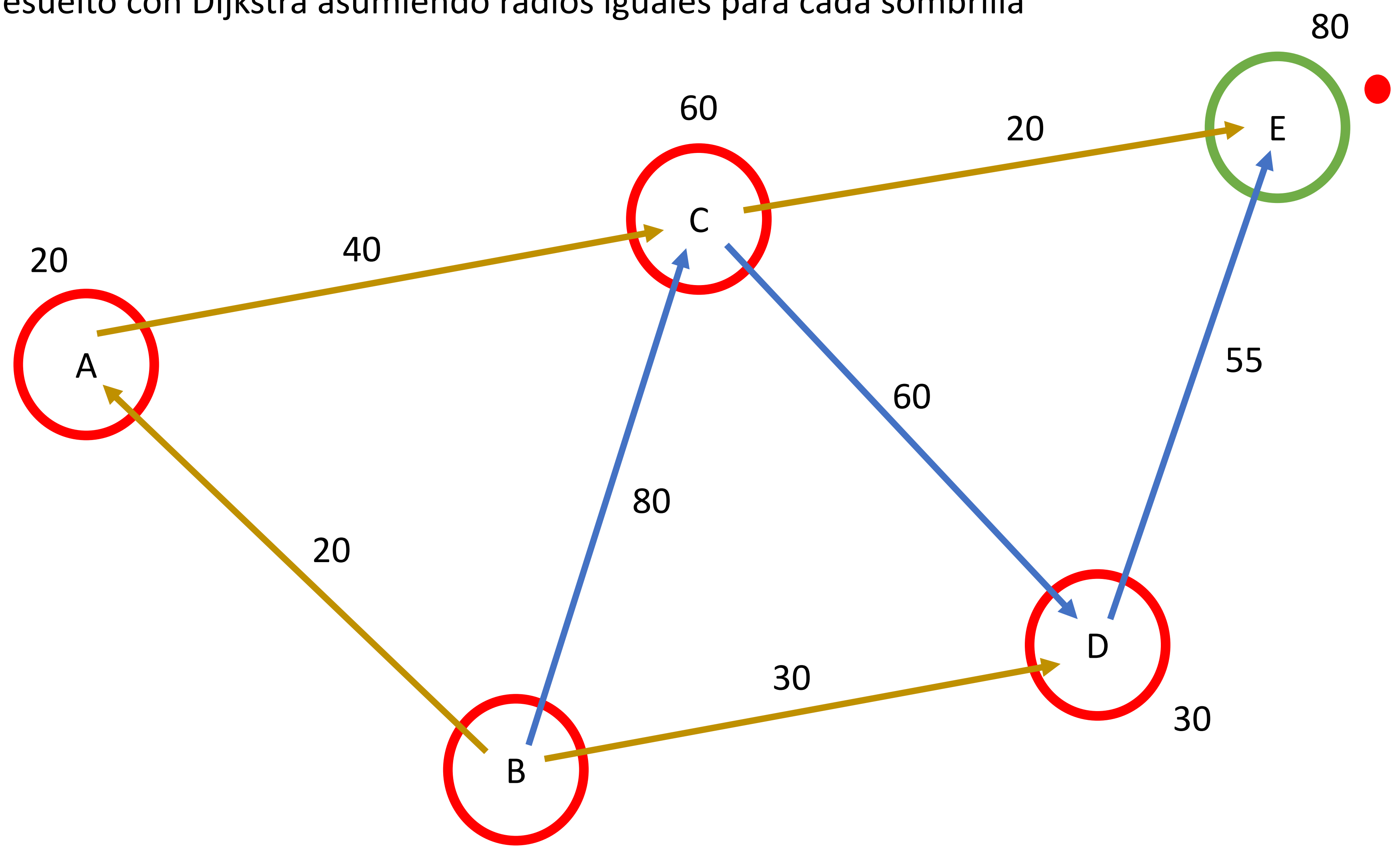
Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla



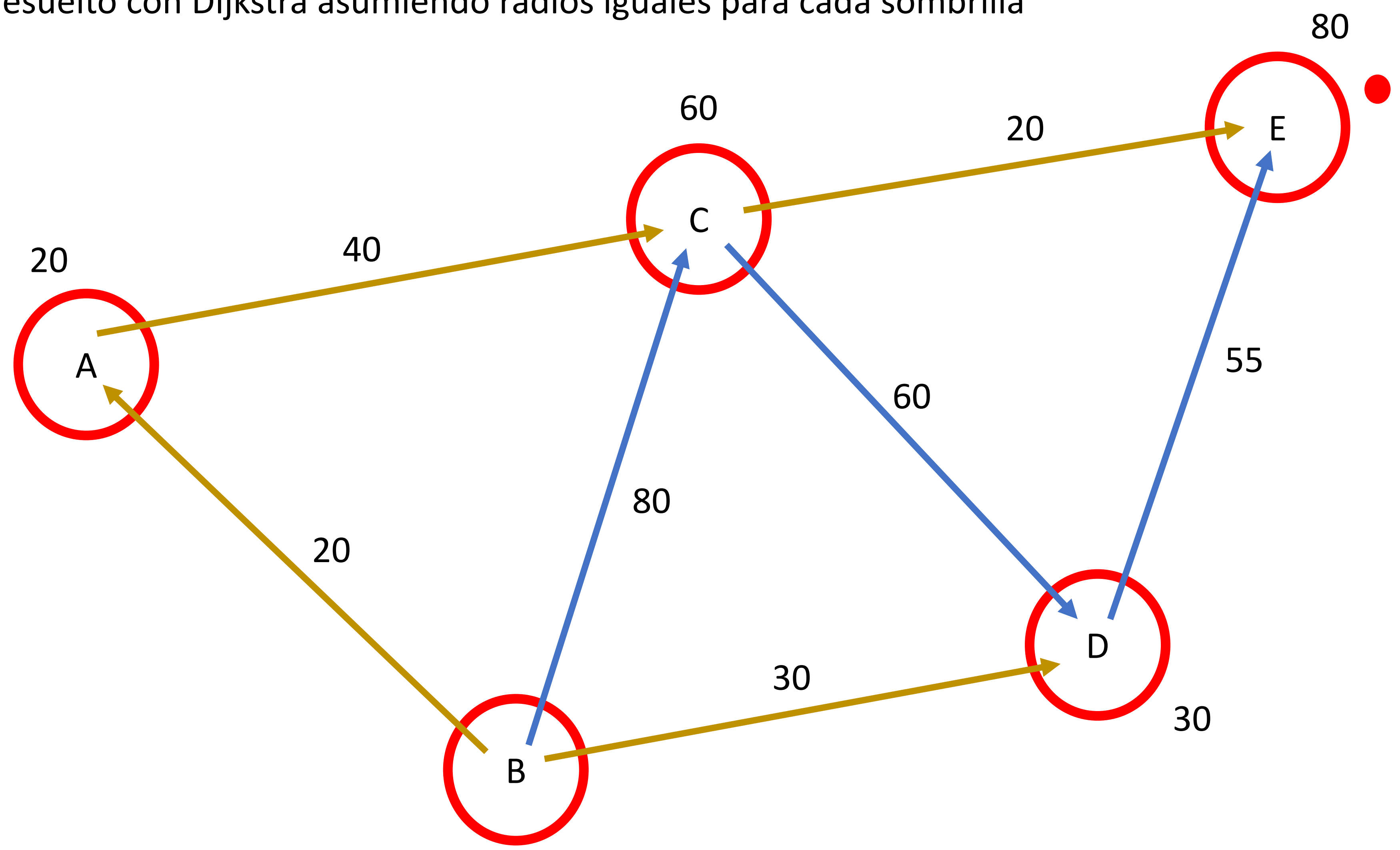
Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla



Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla

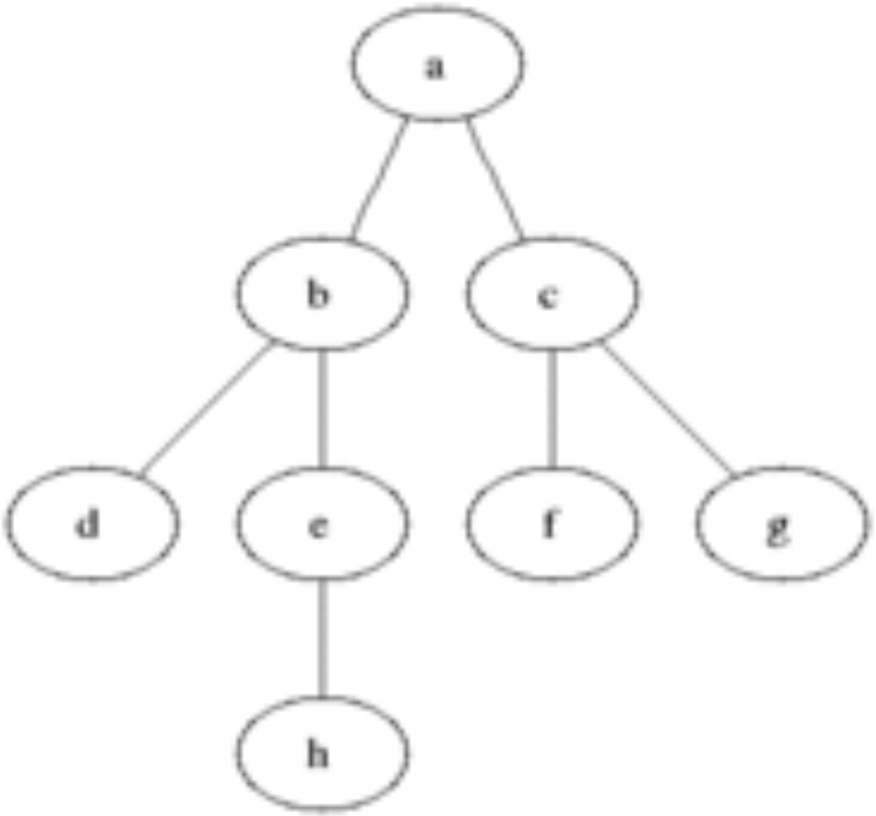


Ejercicio resuelto con Dijkstra asumiendo radios iguales para cada sombrilla



Breadth-First Search

```
BFS(s): —s es el vértice de partida
  for each u in V−{s}:
     u.color = white; u.δ = ∞; π[u] = null
  s.color = gray; s.δ = 0; π[s] = null
  Q = cola; Q.enqueue(s)
  while !Q.empty():
     u = Q.dequeue()
     for each v in α[u]:
        if v.color == white:
           v.color = gray; v.δ = u.δ+1
           π[v] = u; Q.enqueue(v)
     u.color = black
```


a
b
c
d
e
f
g
h

Demostración de que BFS entrega los caminos más cortos

- Definimos un nodo *u* que descubre a un nodo *v*
- $d[v] = d[u] + 1$
- $d[s] = 0$
- $\delta(s,x)$ es el camino más corto desde *s* hasta un nodo *x*

Lema 1: Si u entra en la cola antes que v durante la ejecución de BFS(s), entonces d[u] ≤ d[v]

PD: en cualquier momento, dada una cola $Q = (v_1, v_2, ..., v_n)$, entonces $d[v_1] \leq d[v_2], ..., \leq d[v_n] \leq d[v_1] + 1$.

Caso base: La cola consiste sólamente de s. Por lo tanto, se cumple.

Hipótesis inductiva: Se cumple en la $k-$ésima iteración, que en una cola $Q = (v_1, ..., v_n)$, se cumple que $d[v_1] \leq ... \leq d[v_n] \leq d[v_1] + 1$.

Paso inductivo: En la iteración siguiente se elimina el primer elemento de la cola (v_1), y se agregan al final de la cola los hijos de v_1. Al eliminar v_1, la cola se mantiene como $(v_2, ..., v_n)$, por lo que, por la **HI**, $d[v_1] \leq d[v_2]$, y por lo tanto $d[v_1] + 1 \leq d[v_2] + 1$, por lo que, $d[v_2] \leq ... \leq d[v_n] \leq d[v_1] + 1 \leq d[v_2] + 1$, por lo que la propiedad se mantiene.

Lema 2: Luego de la ejecución de BFS(s), para todo nodo v, d[v] ≥ δ(s, v)

La demostración de este lema es directa. Luego de la ejecución de BFS(s), $d[v]$ corresponderá al largo de *algún camino* de s a v, mediante el cuál, se descubre v, comenzando desde s. Por la definición de $\delta(s, v)$, no existe un camino más corto, ya que justamente lo definimos como tal. Por lo tanto, $d[v] \geq \delta(s, v)$

Teorema: Luego de la ejecución de BFS(s), para cualquier nodo v, d[v] = δ(s, v)

Por contradicción:

- *d[x] ≠ δ(s,v)*
- *v* es el nodo más cercano que cumple esta propiedad
- Por lema 2, *d[v] > δ(s, v)*
- Consideremos *(u,v)* como la última arista del camino más corto de *(s,v)*
- Notemos que *δ(s,v) = δ(s,u)* + 1

$$d[v] > \delta(s, v) = \delta(s, u) + 1 = d[u] + 1$$

$$d[v] > d[u] + 1 \ (*)$$

3 Casos

1. *v* aun no es descubierto
2. *v* ya fue descubierto y explorado
3. *v* fue descubierto, pero aún no ha sido explorado

v aun no ha sido descubierto

En este caso, v es descubierto *durante* la exploración de u, y, por lo tanto, $d[v] = d[u] + 1$, lo que es una contradicción con (*).

v ya fue descubierto y explorado

En este caso, v entró a la cola (y fue removido) antes que u entrara a la cola. Por el **Lema 1**, $d[v] \leq d[u]$, lo que es una contradicción con (*)

v fue descubierto, pero aún no ha sido explorado

- Sea *w* el nodo que decubre a *v*
- *w* entró antes a la cola que *u*
- Por Lema 1 $d[w] \leq d[u]$ y $d[w] + 1 \leq d[u] + 1$
- Como *v* fue descubierto por *w* $d[v] = d[w] + 1$

- Nos queda reemplazando que

$$d[v] \leq d[u] + 1$$

Contradicción con

$$\mathrm{d}[\mathrm{v}] > \mathrm{d}[\mathrm{u}] + 1$$