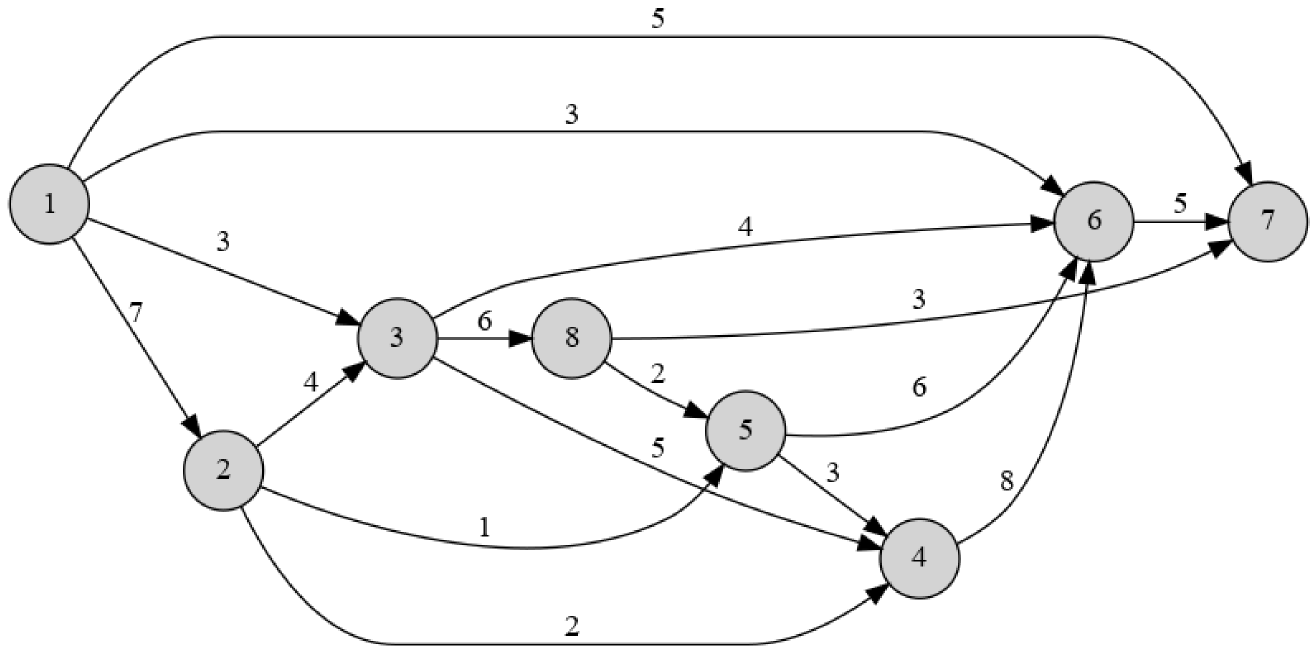
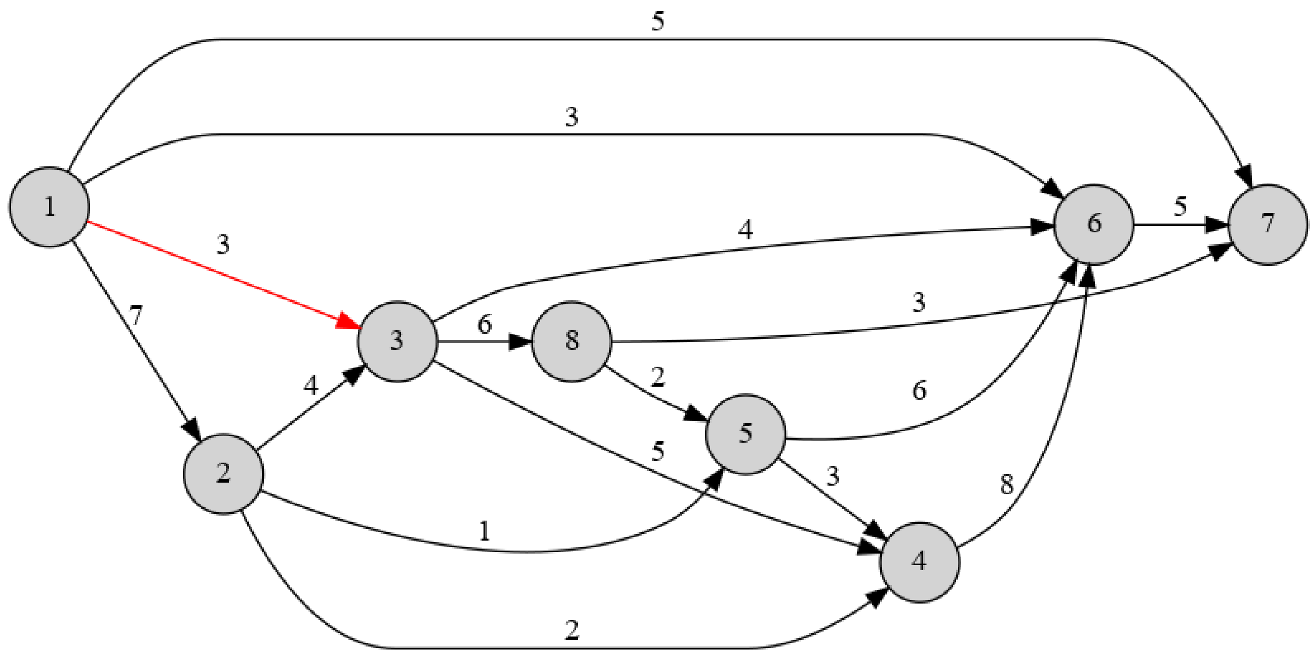
1

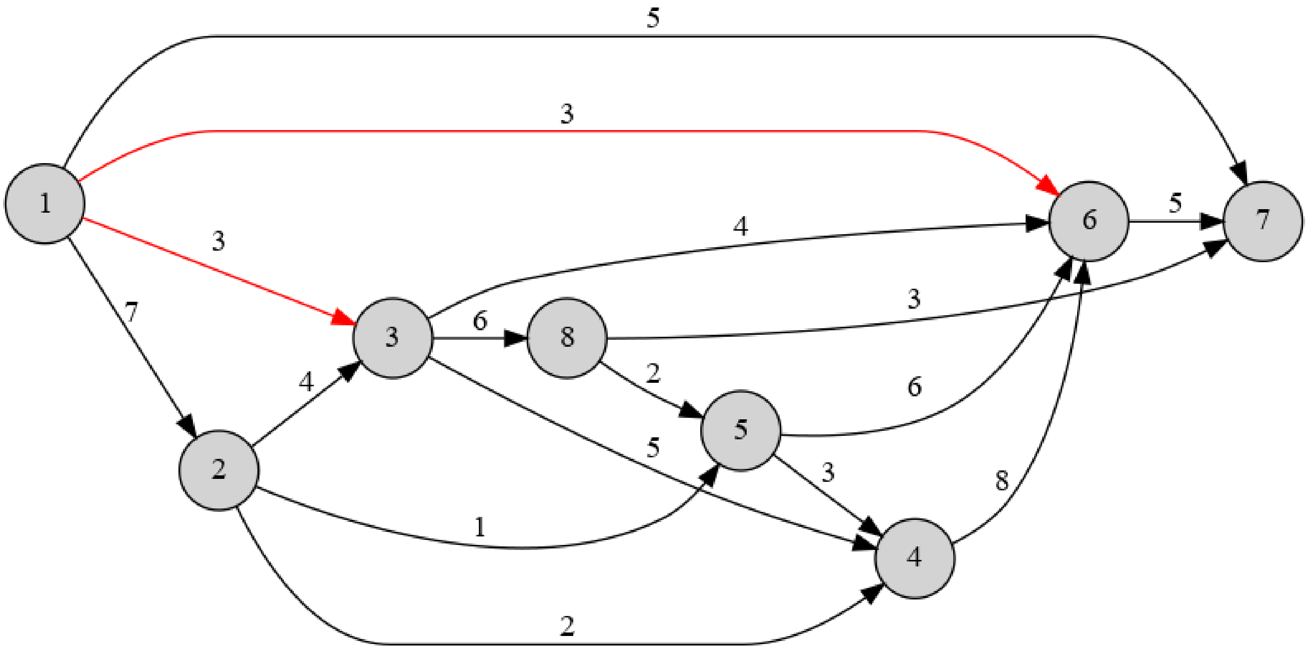
Ejecutar paso a paso, graficando las soluciones parciales, el algoritmo de $Prim$ que computa el *árbol generador mínimo* sobre los grafos con nodos $\{1, 2, \ldots, 8\}$ y costos dados por una función w:

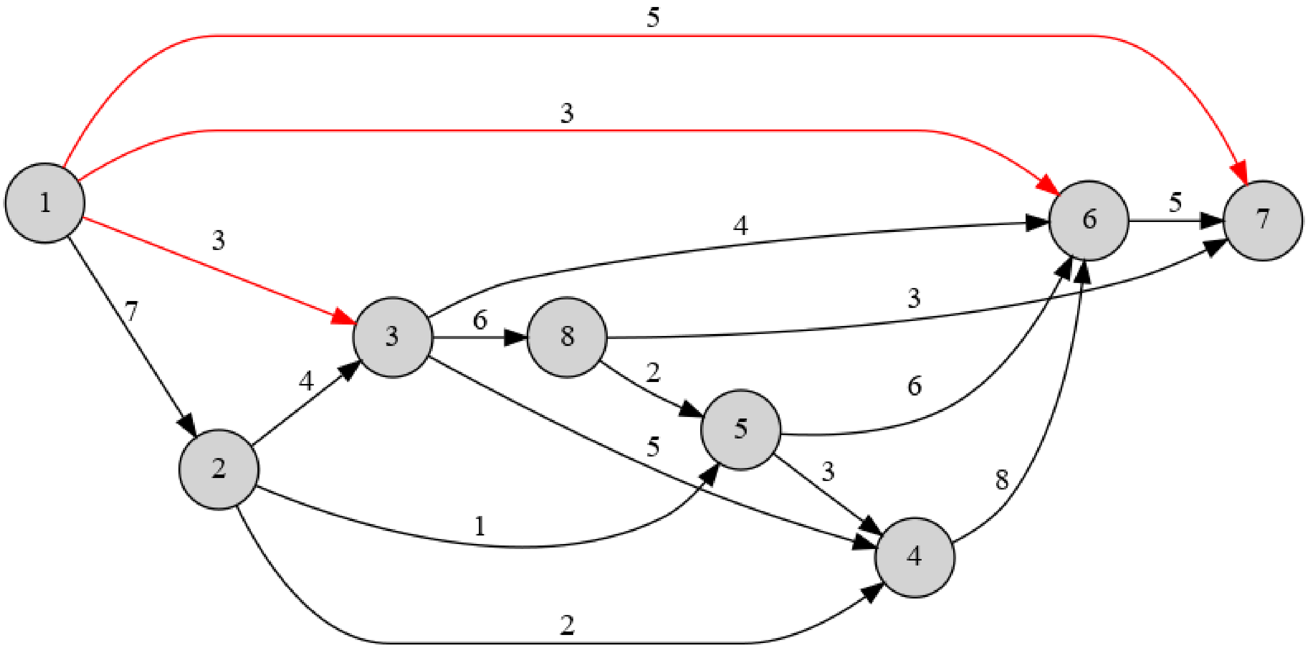
a

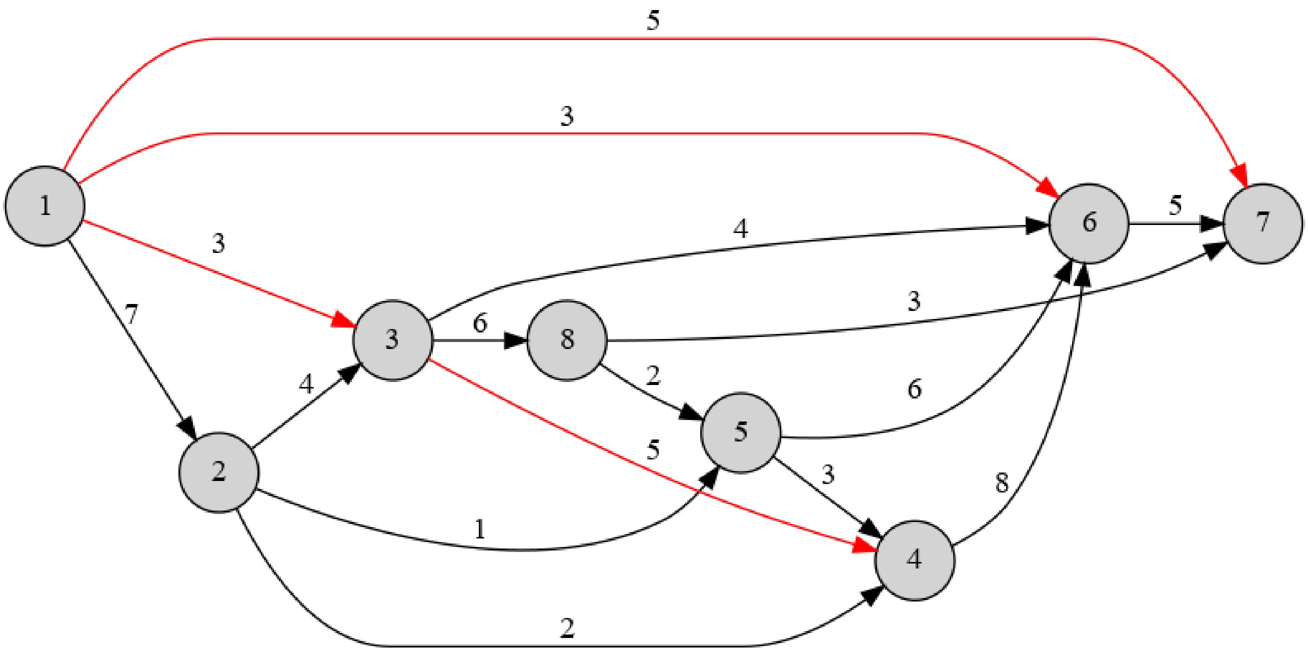
$$\begin{array}{llll}
w((1,2)) = 7 & w((2,3)) = 4 & w((3,6)) = 4 & w((5,6)) = 6 \\
w((1,6)) = 3 & w((2,4)) = 2 & w((3,8)) = 6 & w((6,7)) = 5 \\
w((1,7)) = 5 & w((2,5)) = 1 & w((4,6)) = 8 & w((8,5)) = 2 \\
w((1,3)) = 3 & w((3,4)) = 5 & w((5,4)) = 3 & w((8,7)) = 3
\end{array}$$

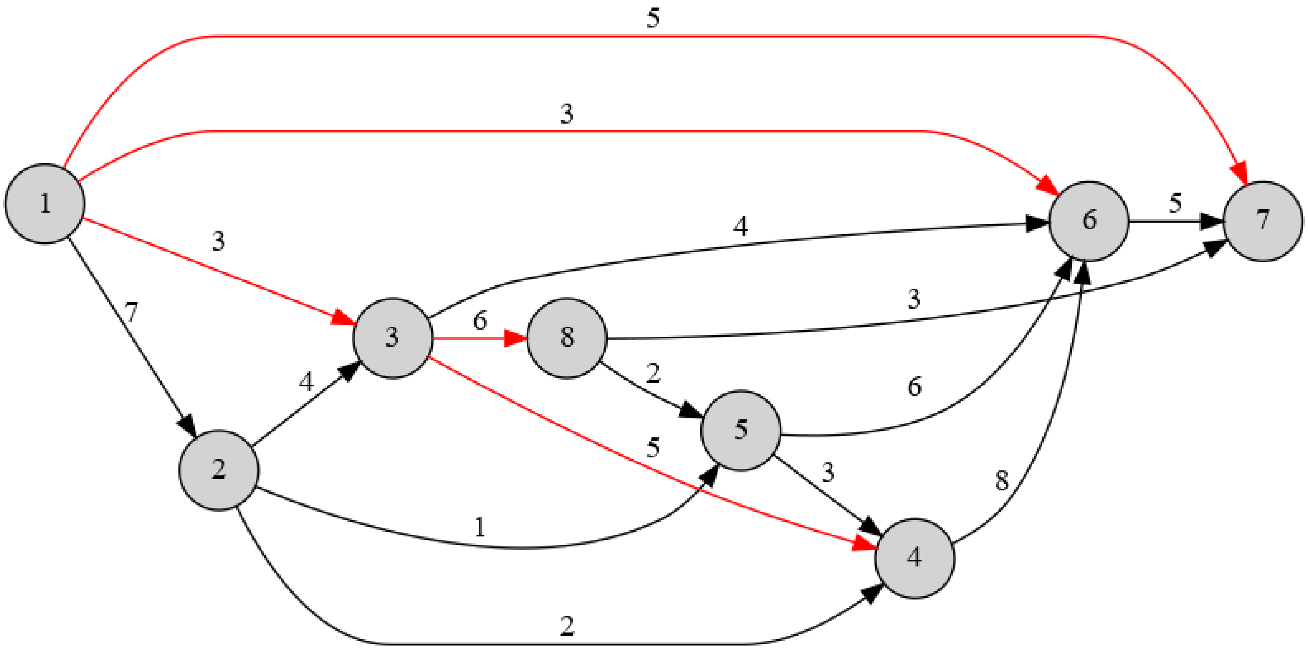


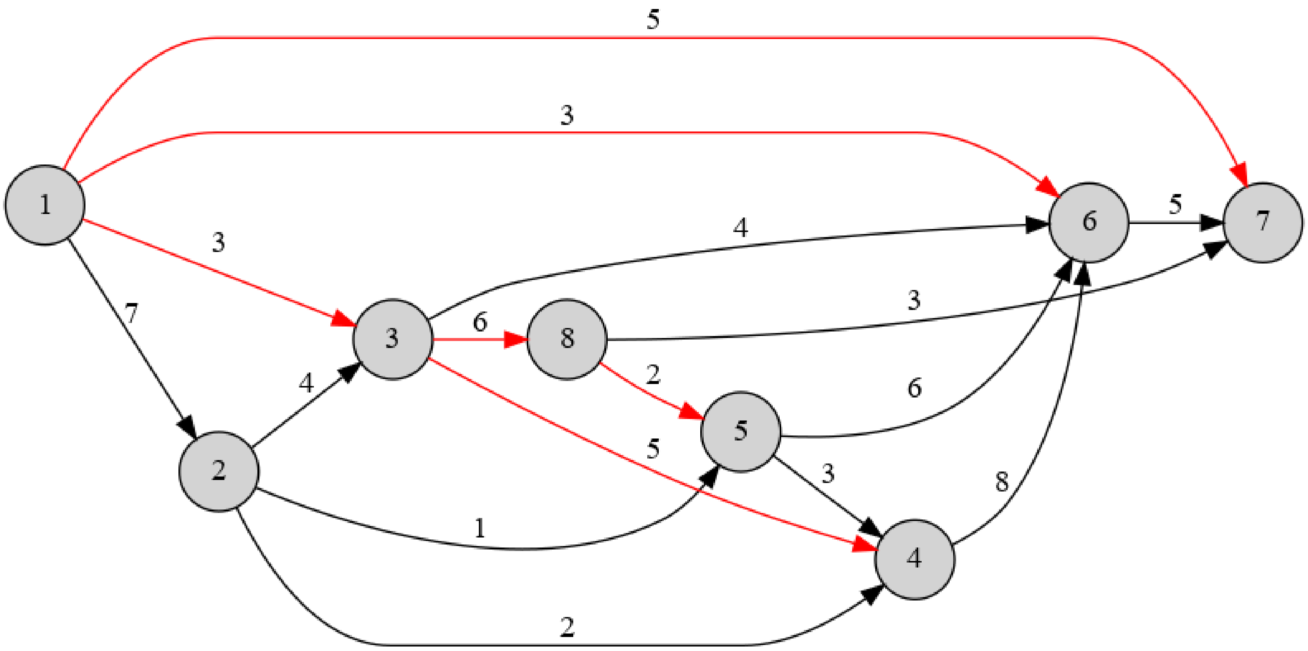


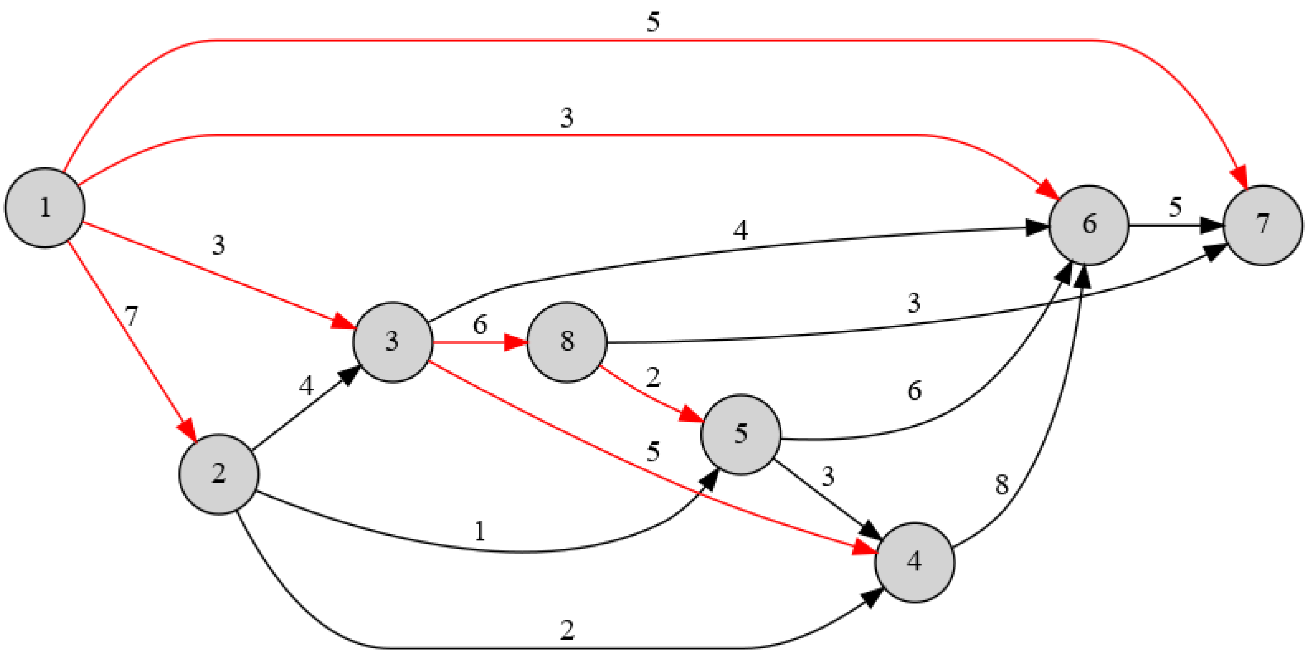

5
3
1
3
7
4
6
5
7
3
6
8
3
4
2
6
5
2
5
3
8
1
4
2


5
3
1
3
7
4
6
5
7
3
6
8
3
4
2
6
5
2
5
3
8
1
4
2


5
3
1
3
7
4
6
5
7
3
6
8
3
4
2
6
5
2
5
3
8
1
4
2

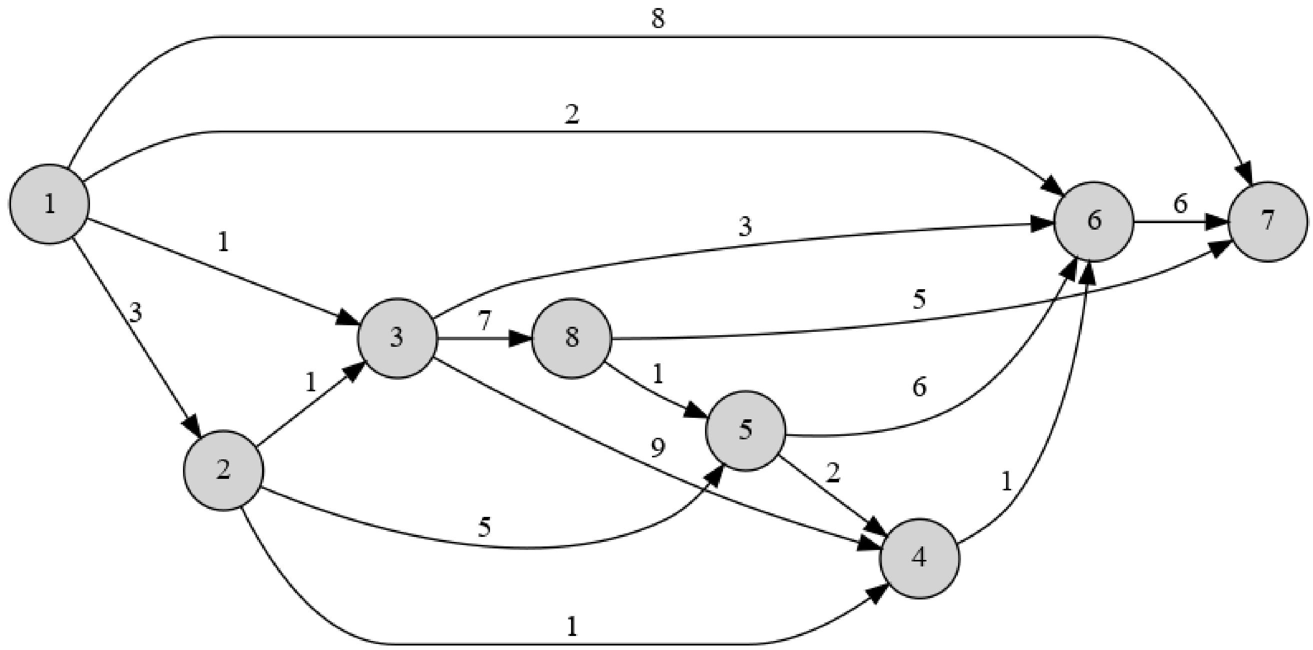

5
3
1
6
5
7
3
4
7
3
6
8
3
4
2
6
5
5
3
2
8
1
4
2

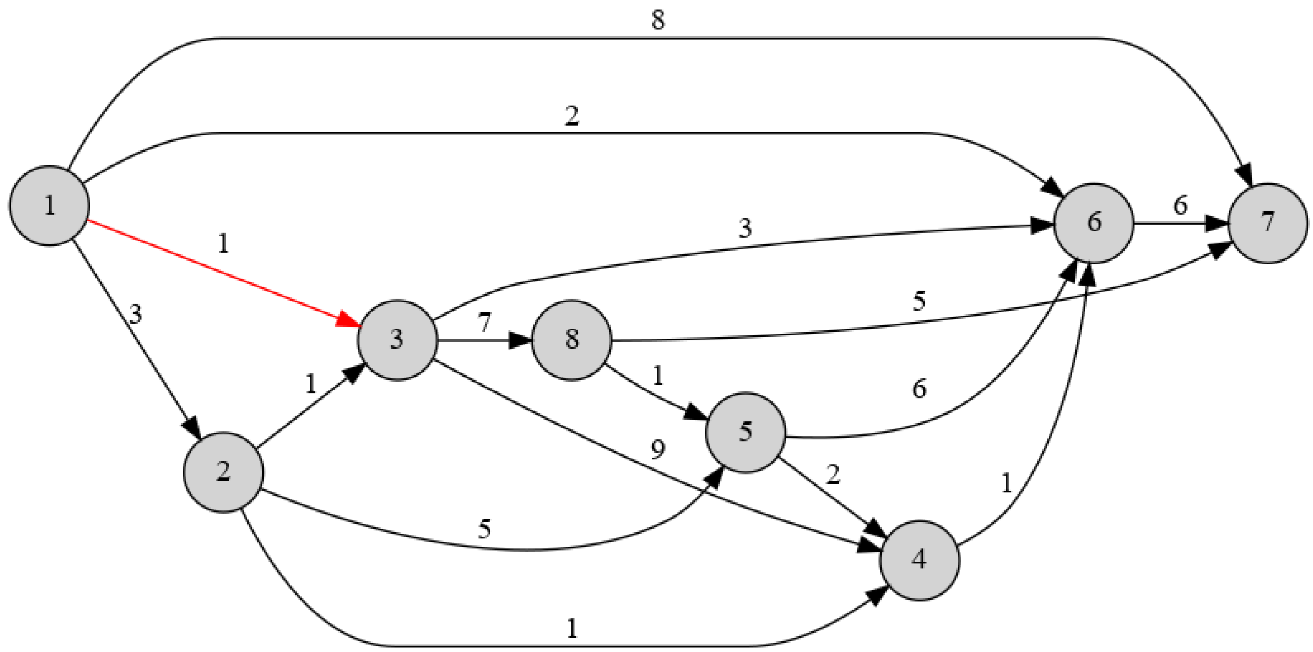

5
3
1
6
5
7
3
4
7
3
6
8
3
4
2
6
5
5
3
2
8
1
4
2

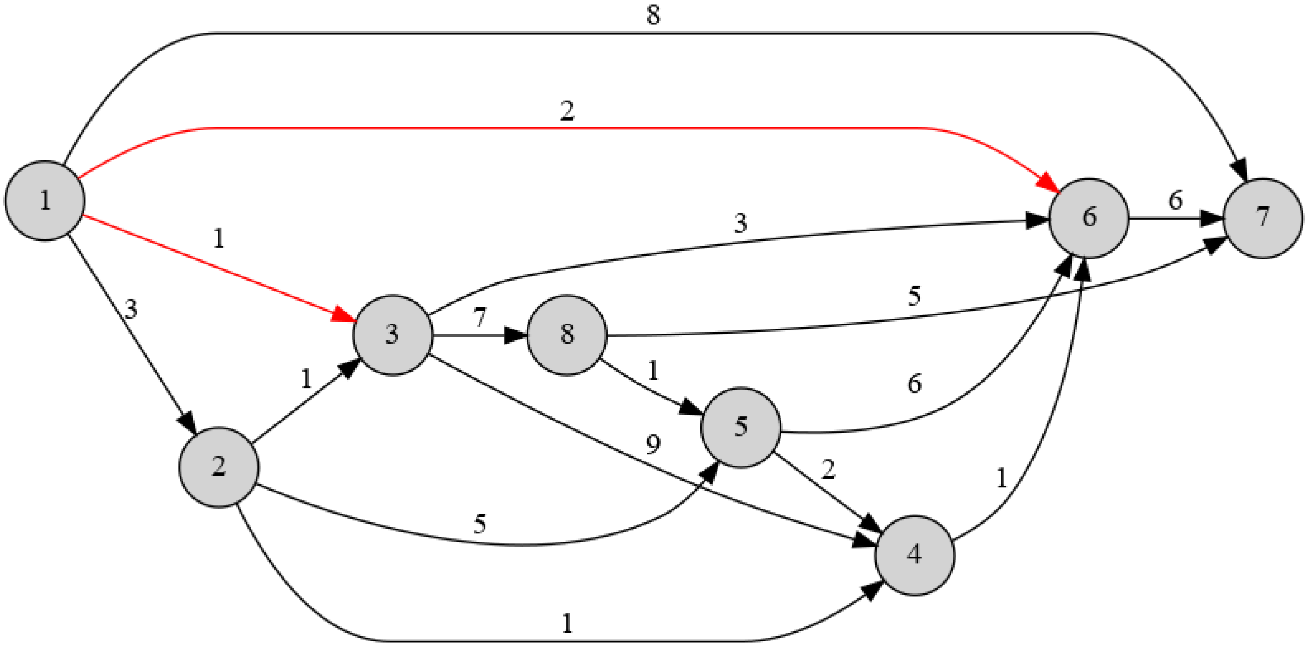

5
3
1
6
5
7
3
4
7
3
6
8
3
4
2
6
5
5
3
2
8
1
4
2

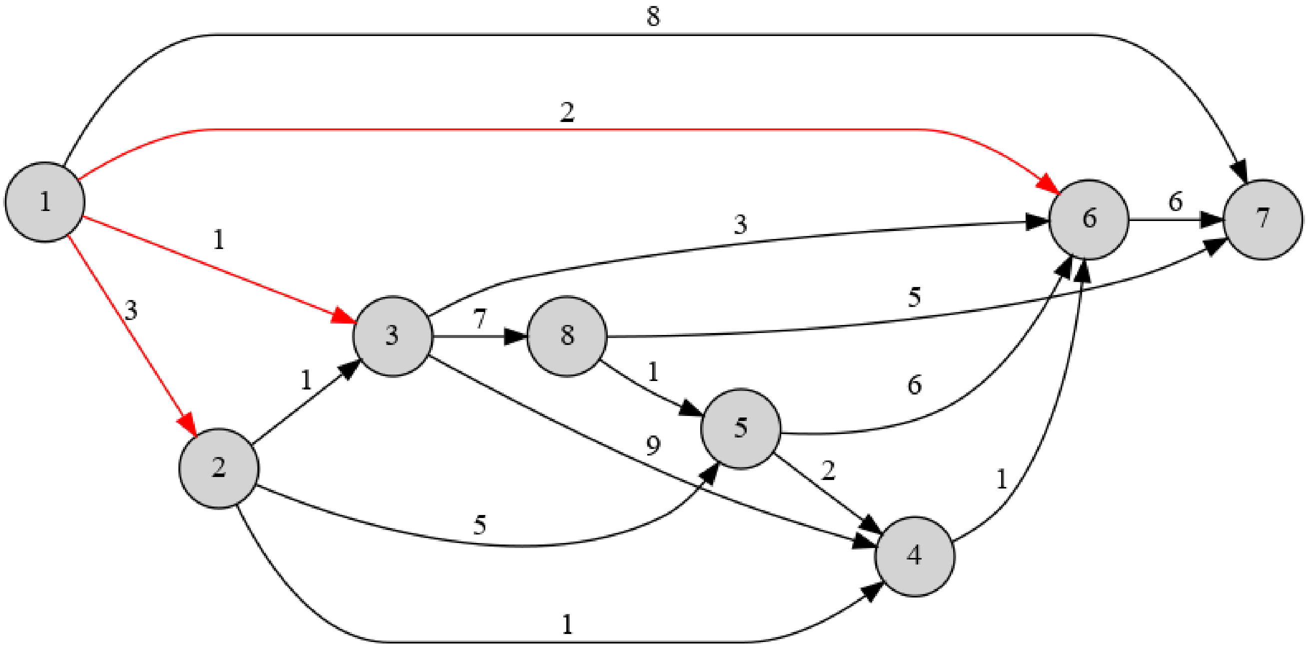
b

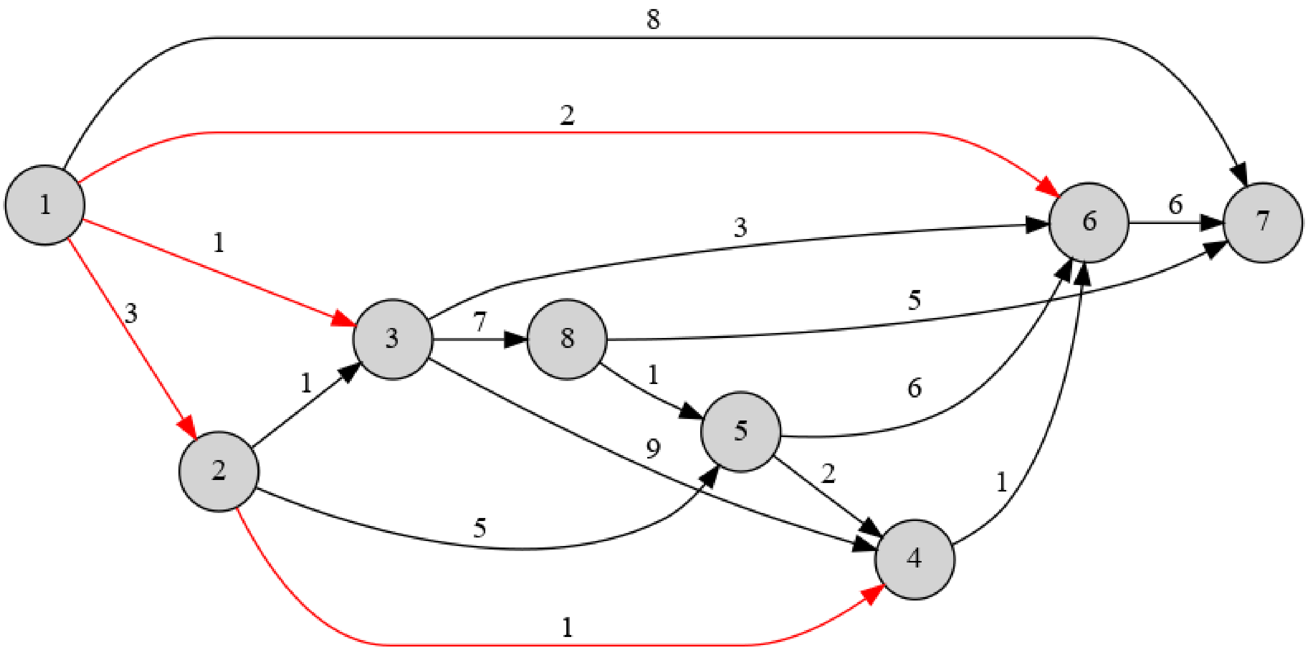
$$\begin{array}{llll} w((1,2)) = 3 & w((2,3)) = 1 & w((3,6)) = 3 & w((5,6)) = 6 \\ w((1,6)) = 2 & w((2,4)) = 1 & w((3,8)) = 7 & w((6,7)) = 6 \\ w((1,7)) = 8 & w((2,5)) = 5 & w((4,6)) = 1 & w((8,5)) = 1 \\ w((1,3)) = 1 & w((3,4)) = 9 & w((5,4)) = 2 & w((8,7)) = 5 \end{array}$$

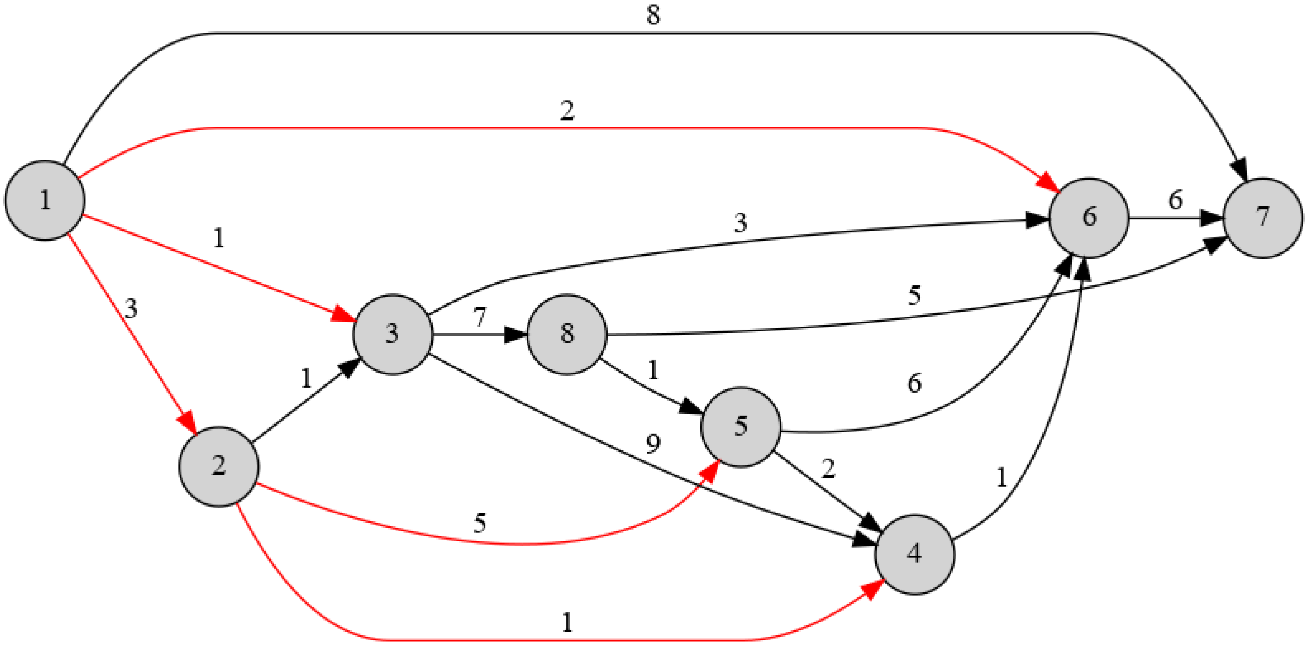


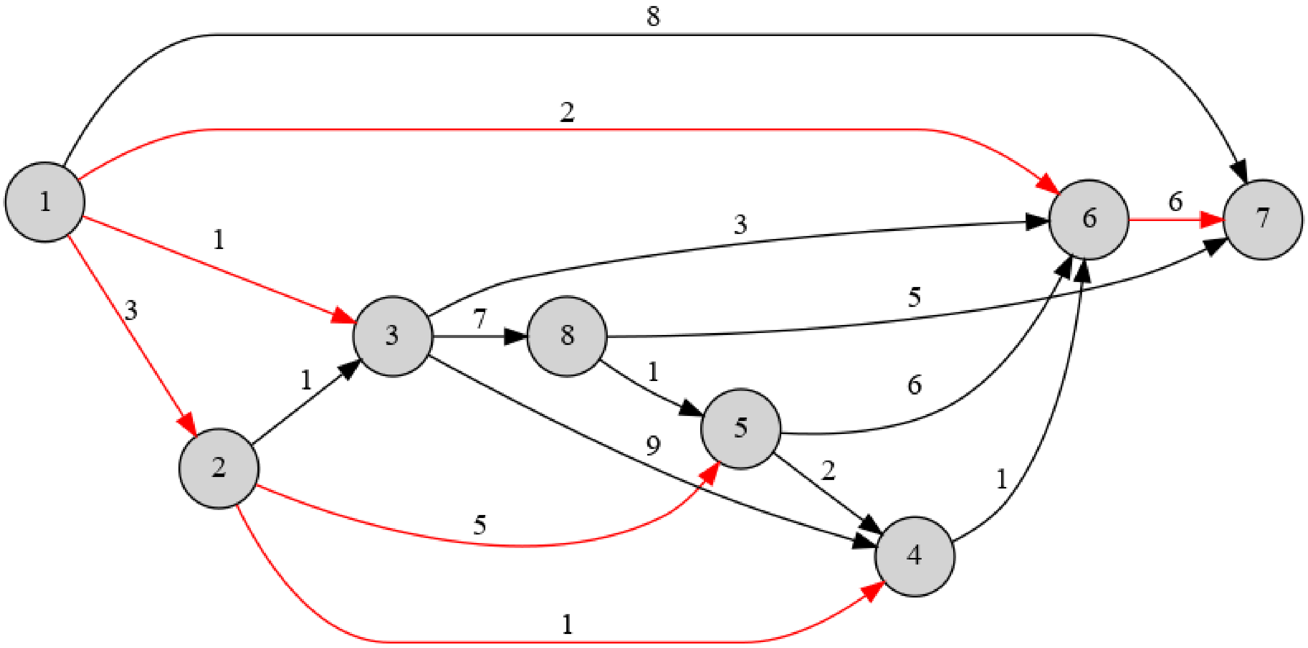


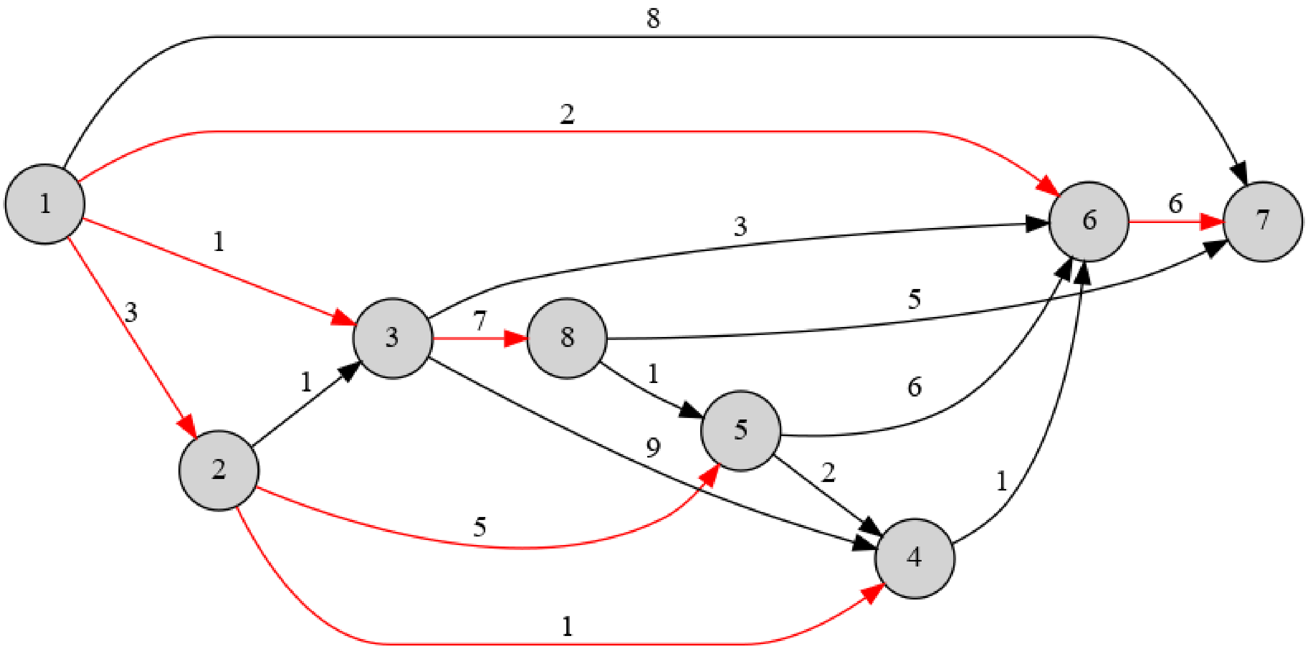

8
2
1
6
6
7
1
3
3
3
7
8
5
1
1
6
5
9
2
2
1
5
4
1


8
2
1
6
6
7
1
3
3
3
7
8
5
1
1
6
5
9
2
2
1
5
4
1


8
2
1
6
6
7
1
3
3
3
7
8
5
1
1
6
5
9
2
2
1
5
4
1





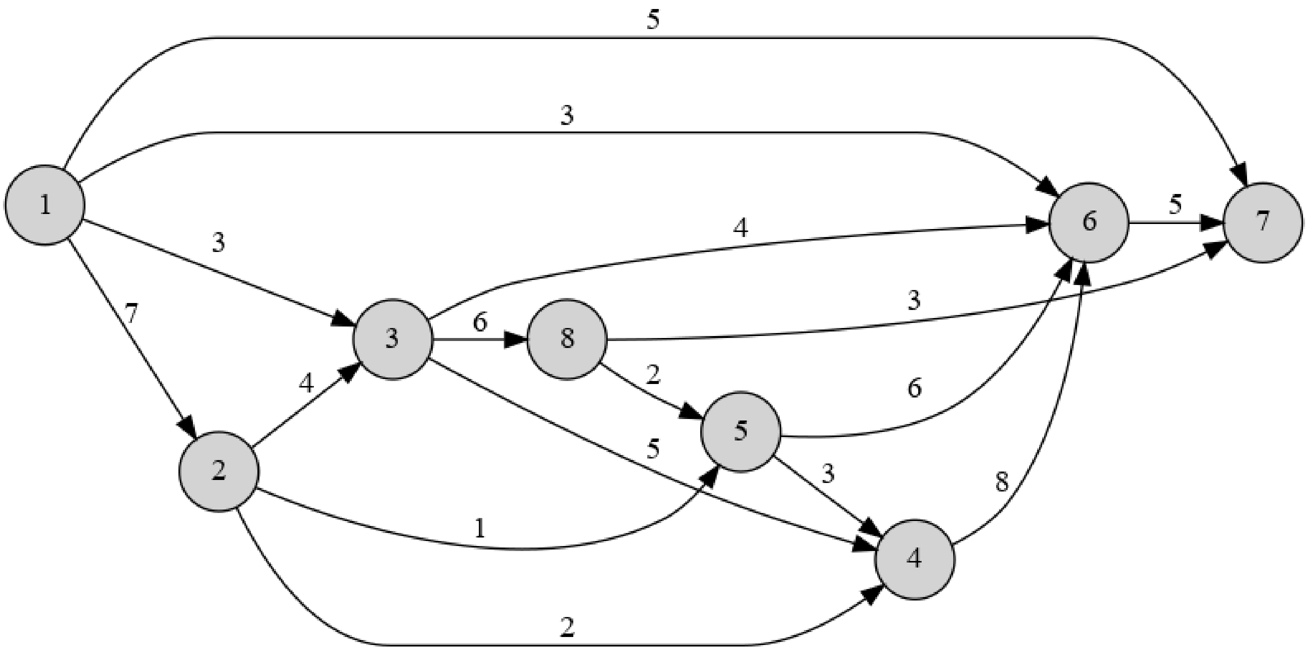


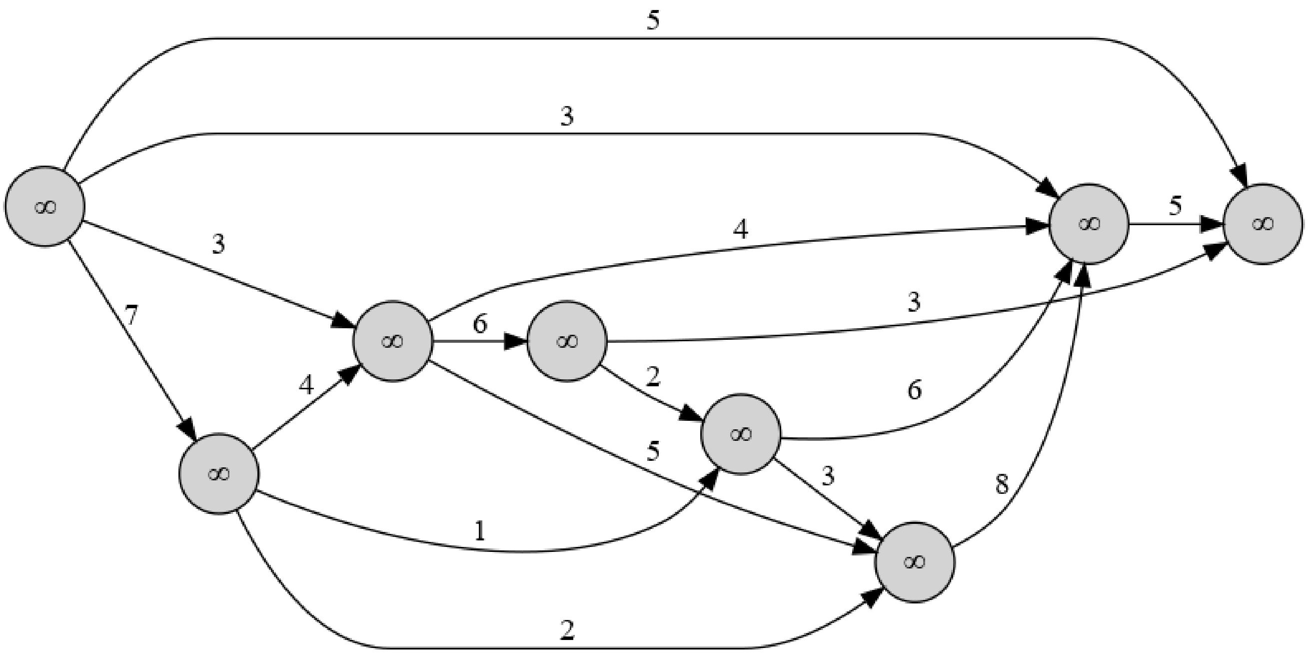
2

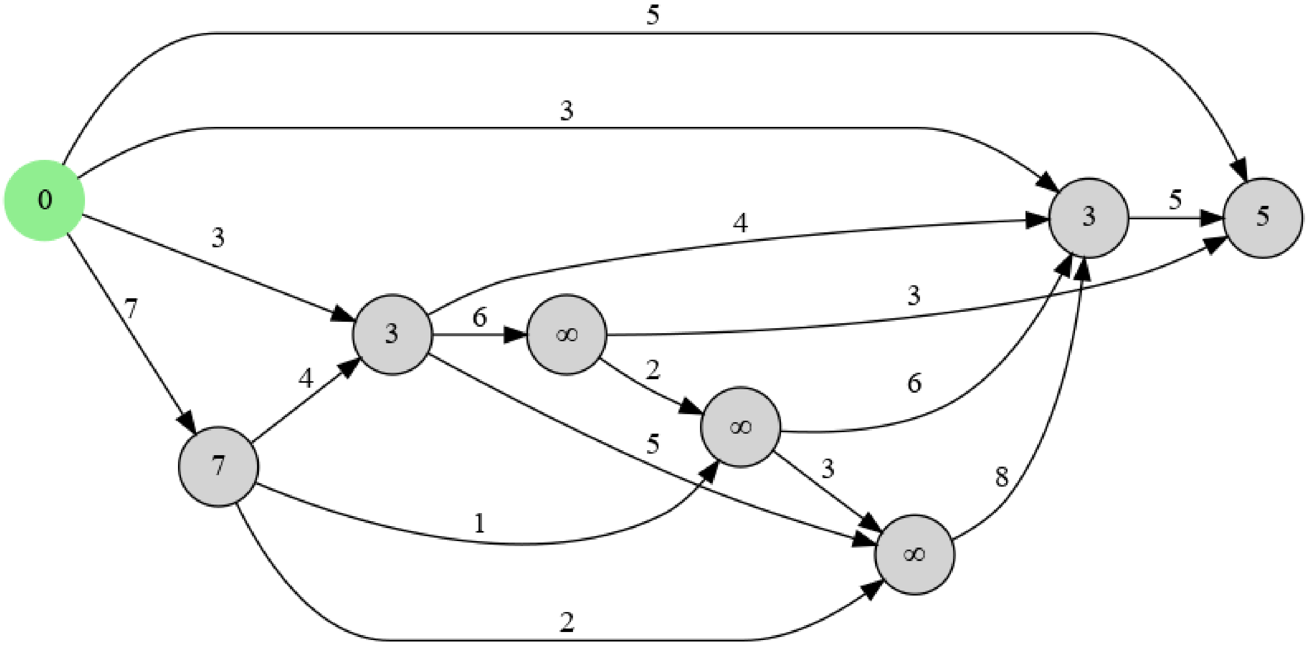
Ejecutar paso a paso el algoritmo de Dijkstra que computa el *camino de costo mínimo* entre un nodo dado y los restantes nodos de un grafo, sobre los dos grafos especificados en el ejercicio anterior.

Considerar 1 como el nodo inicial. Explicitar en cada paso el conjunto de nodos para los cuales ya se ha computado el costo mínimo y el arreglo con tales costos.

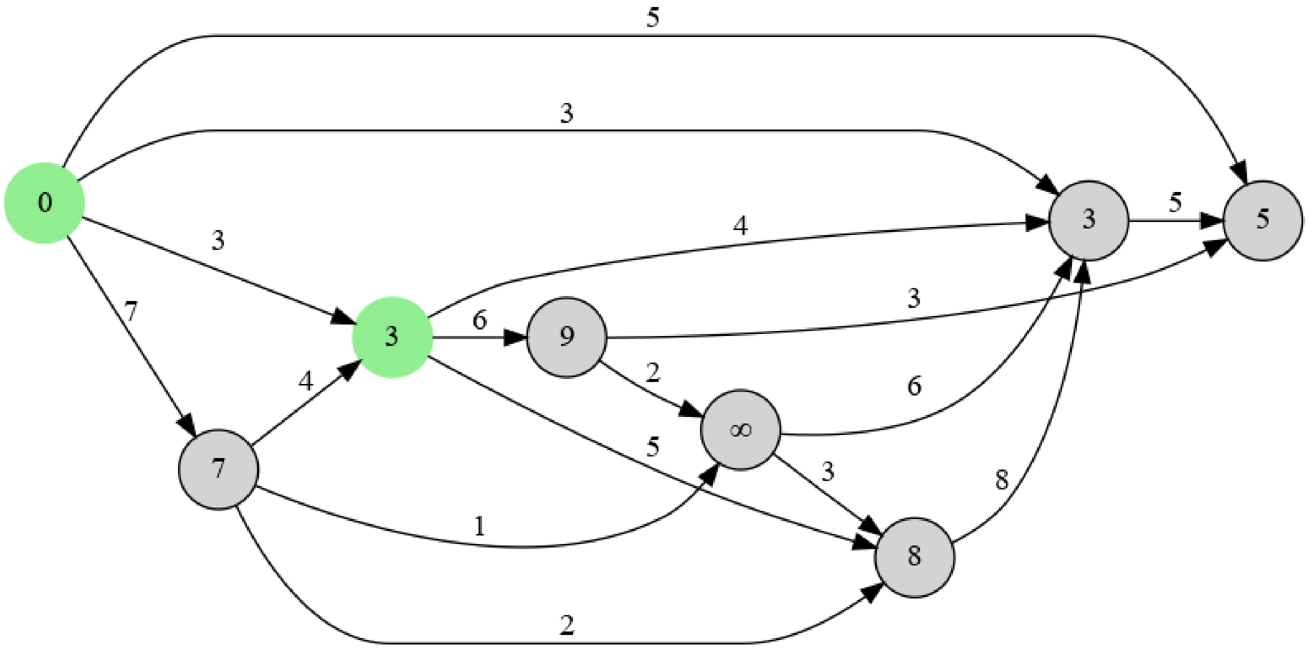
a



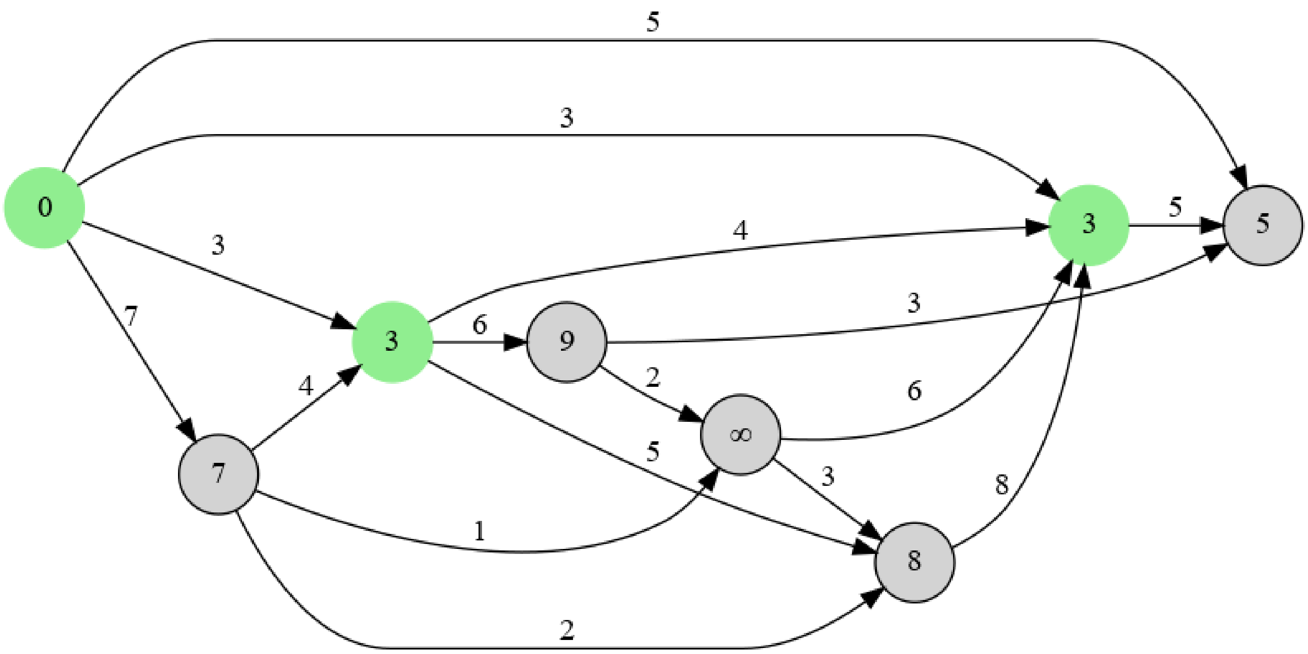




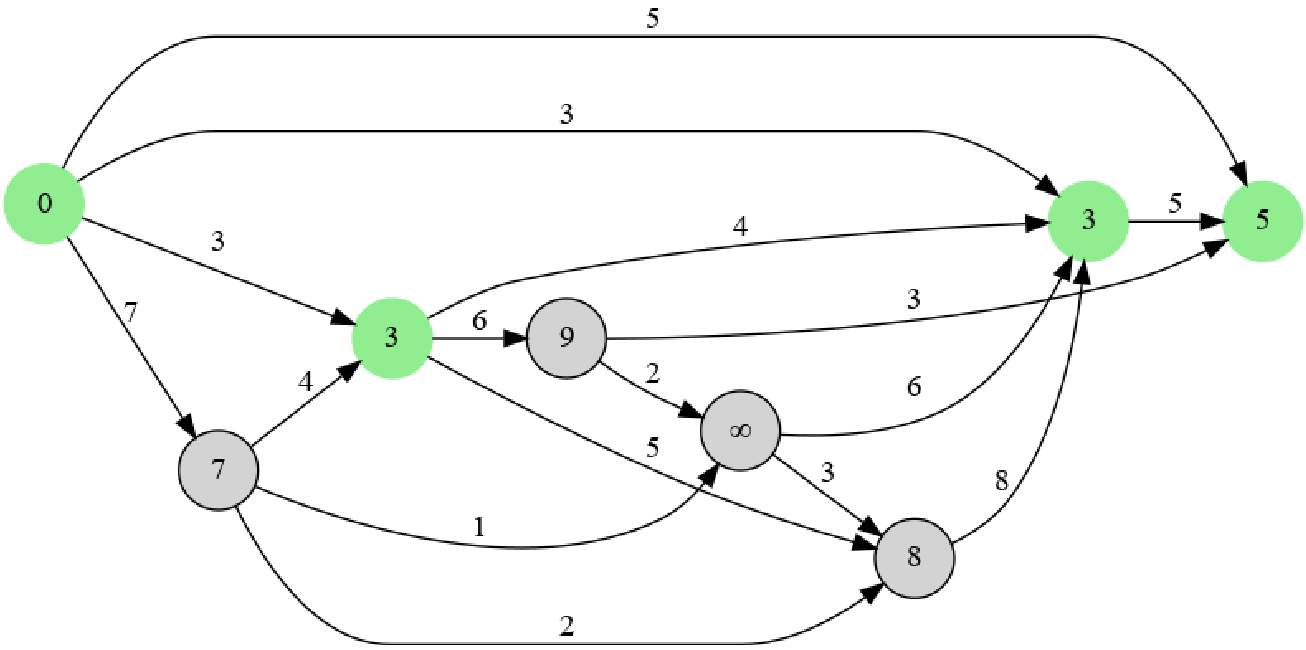
5
3
0
3
5
3
4
7
3
6
∞
3
4
2
∞
6
7
5
3
8
1
∞
2

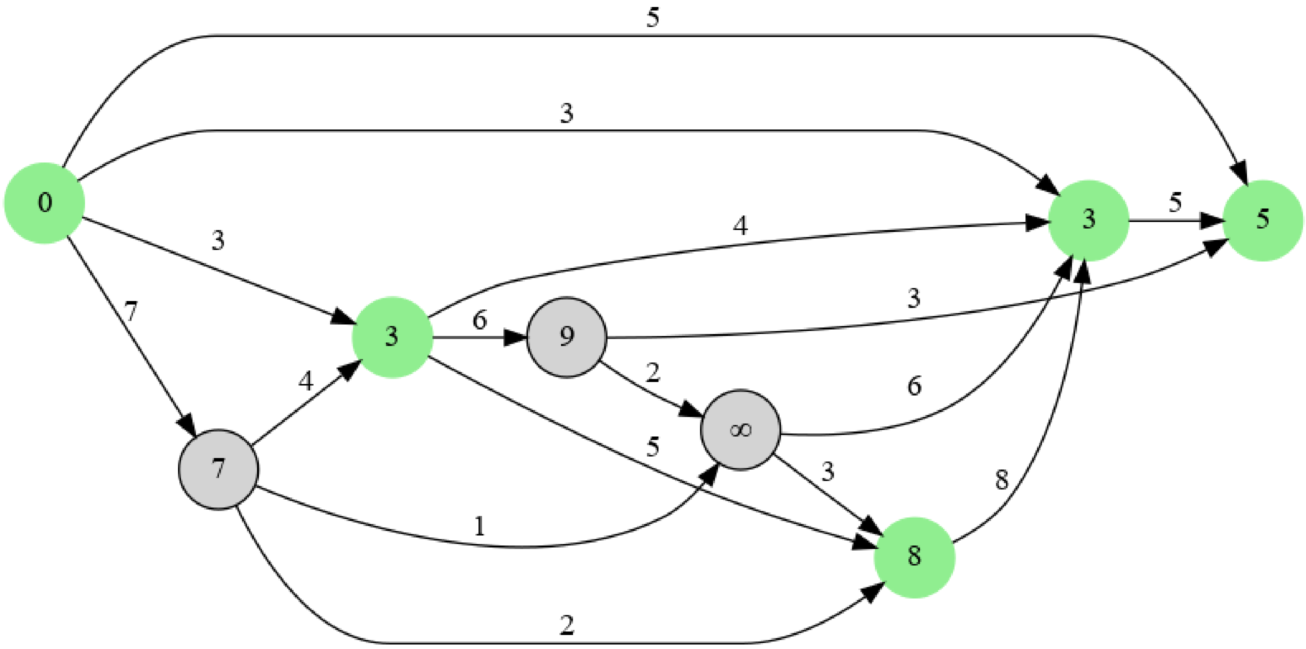


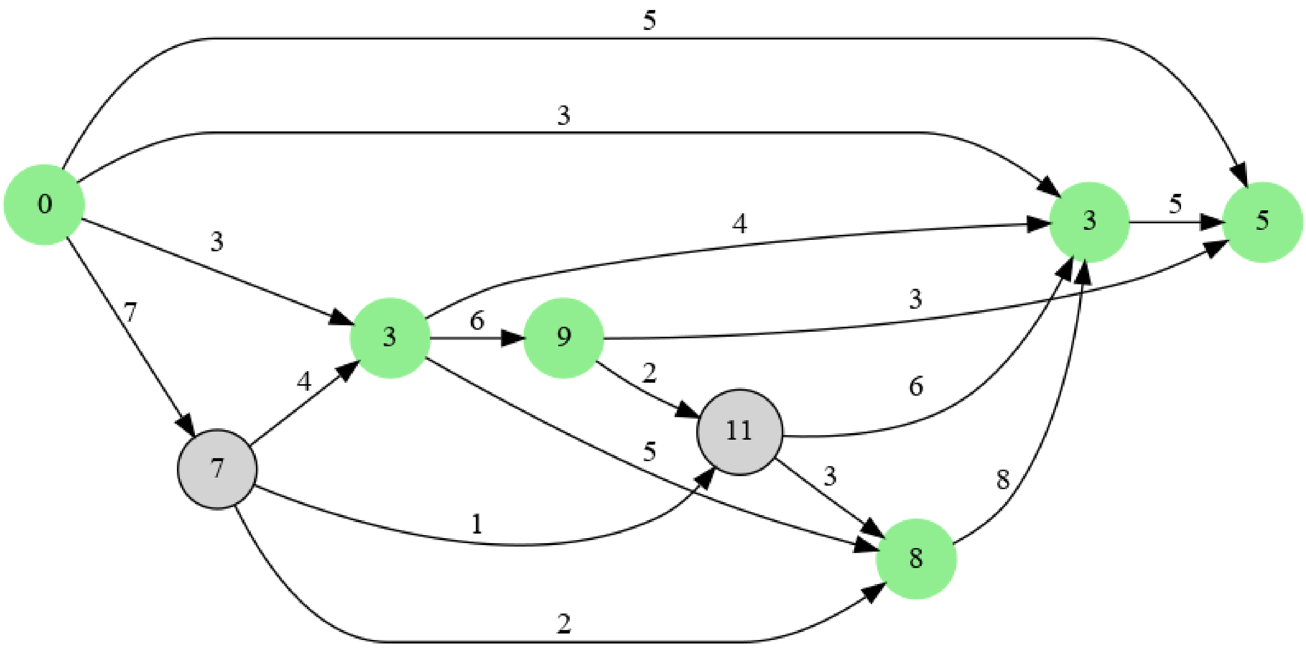
5
3
0
3
5
3
4
7
3
6
9
3
4
2
∞
6
7
5
3
8
1
8
2

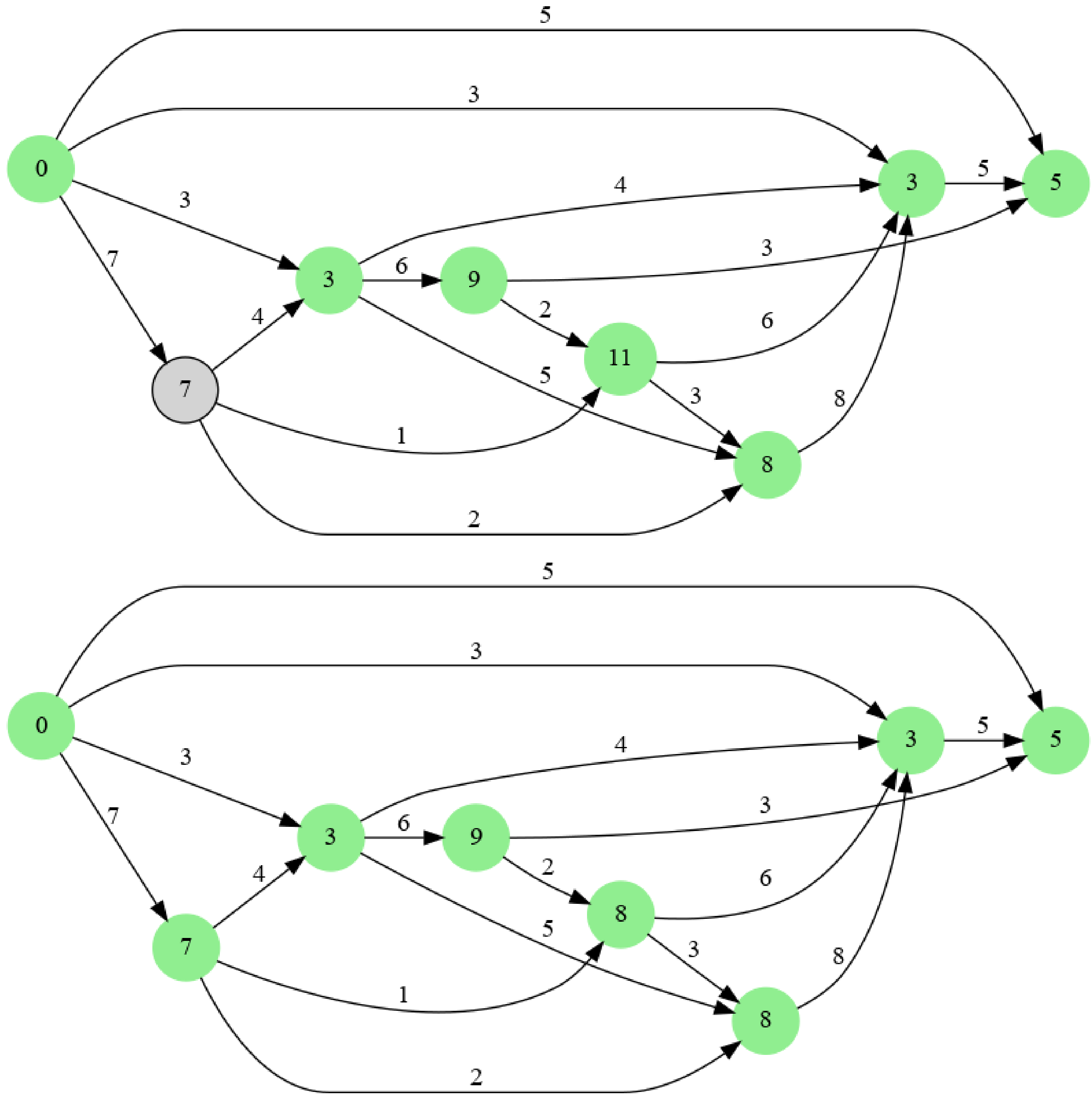


5
3
0
3
5
3
4
7
3
6
9
3
4
2
∞
6
7
5
3
8
1
8
2

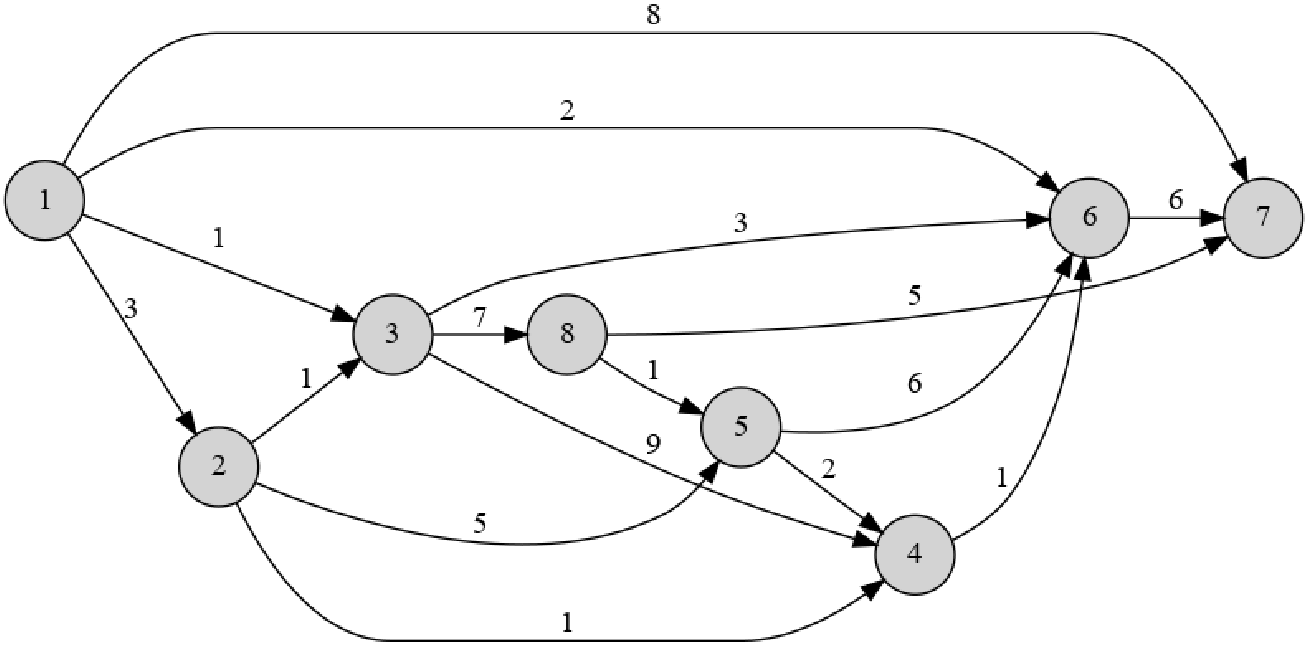

5
3
0
3
3
5
5
4
3
7
3
6
9
3
4
2
∞
6
5
7
3
1
8
8
2

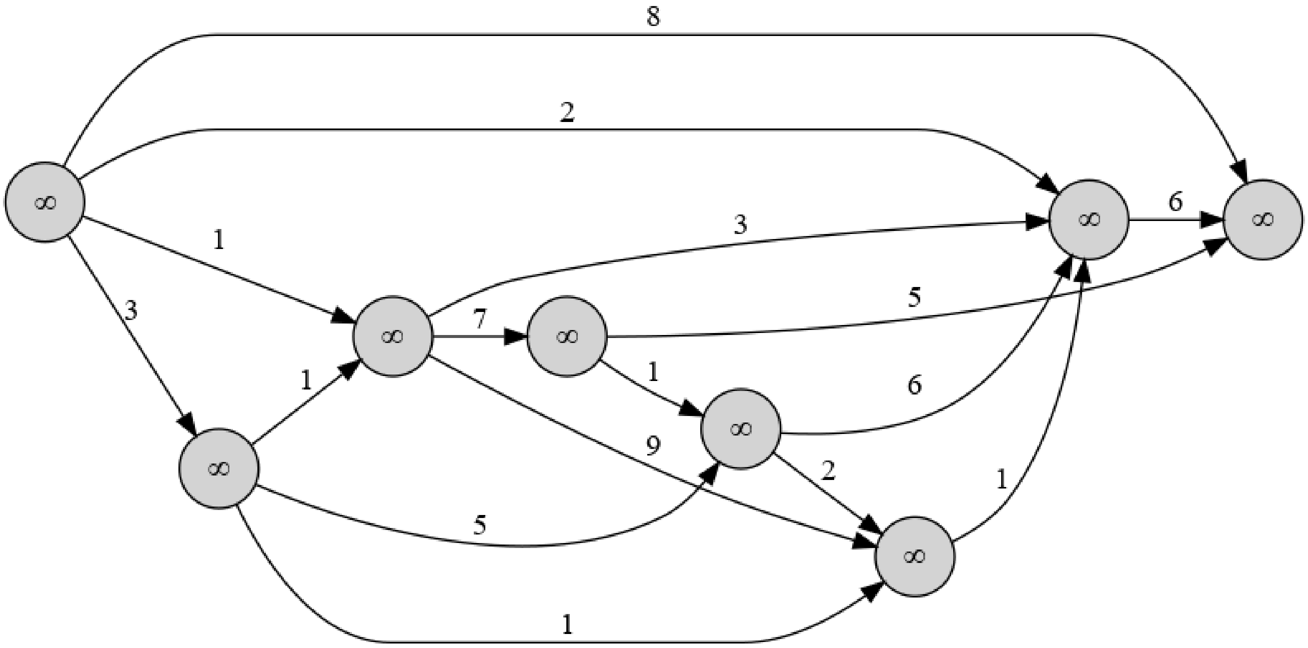

5
3
0
3
5
5
3
4
3
7
3
6
9
3
4
2
∞
6
5
7
3
1
8
8
2

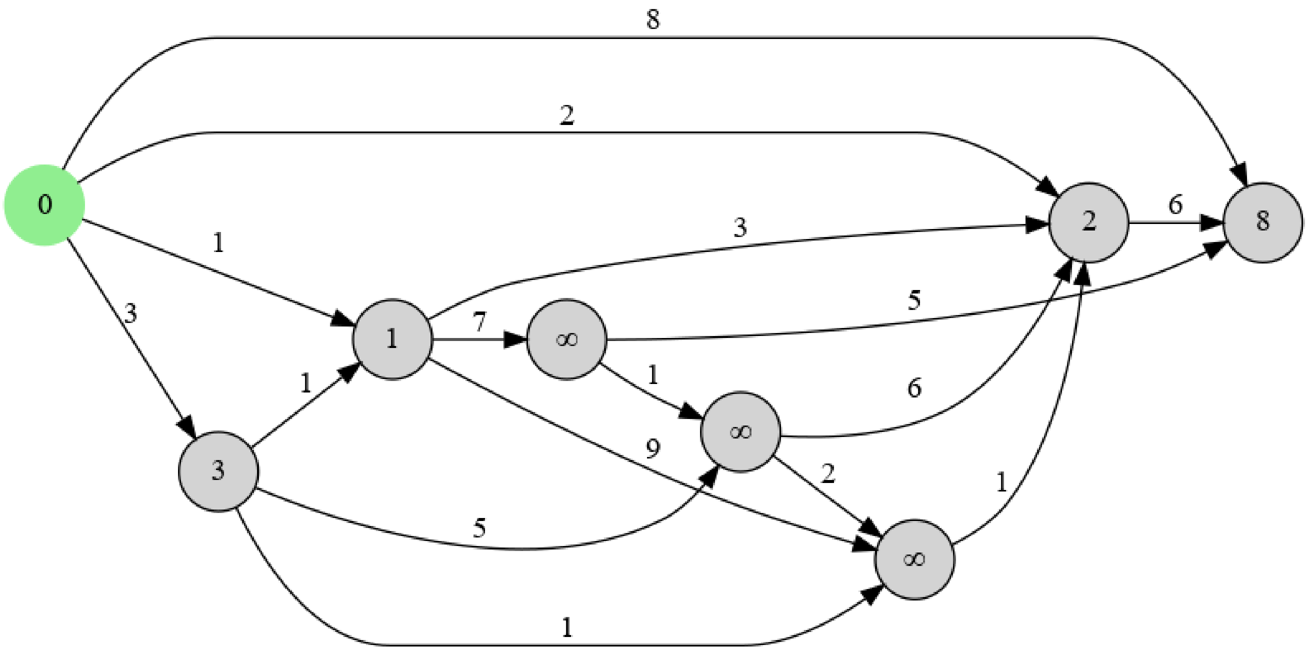

5
3
0
3
5
5
3
4
3
7
3
6
9
3
4
2
11
6
5
7
3
1
8
8
2

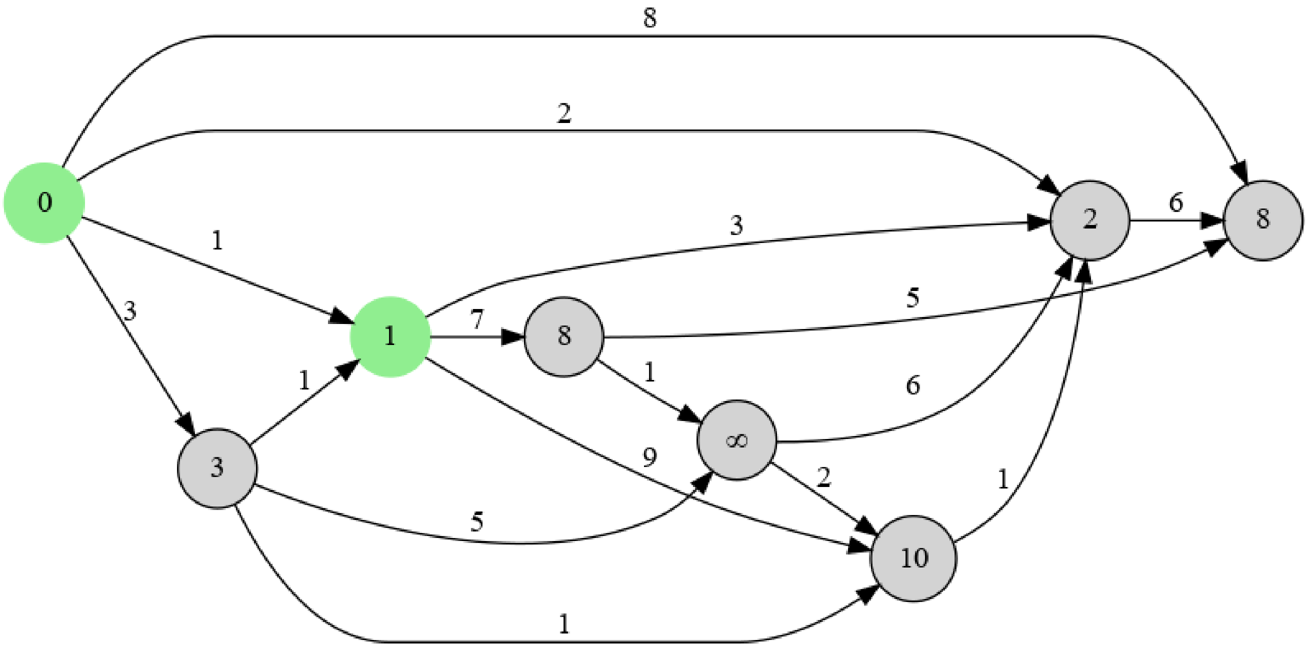


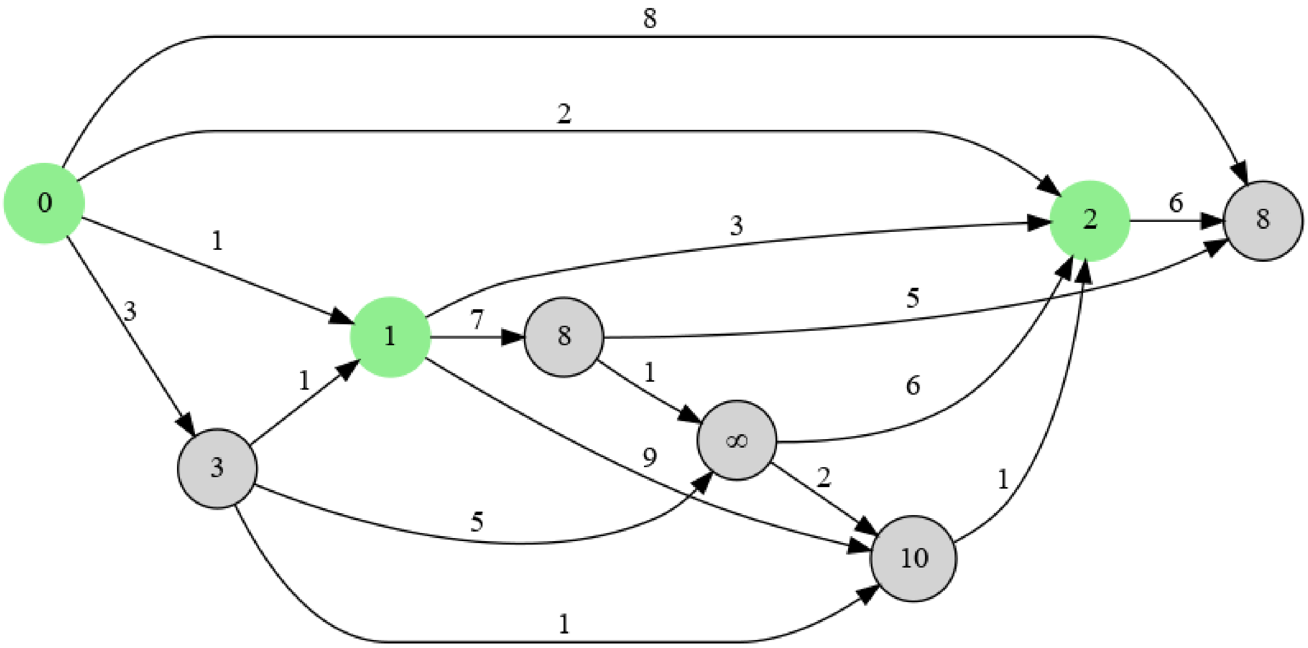
b

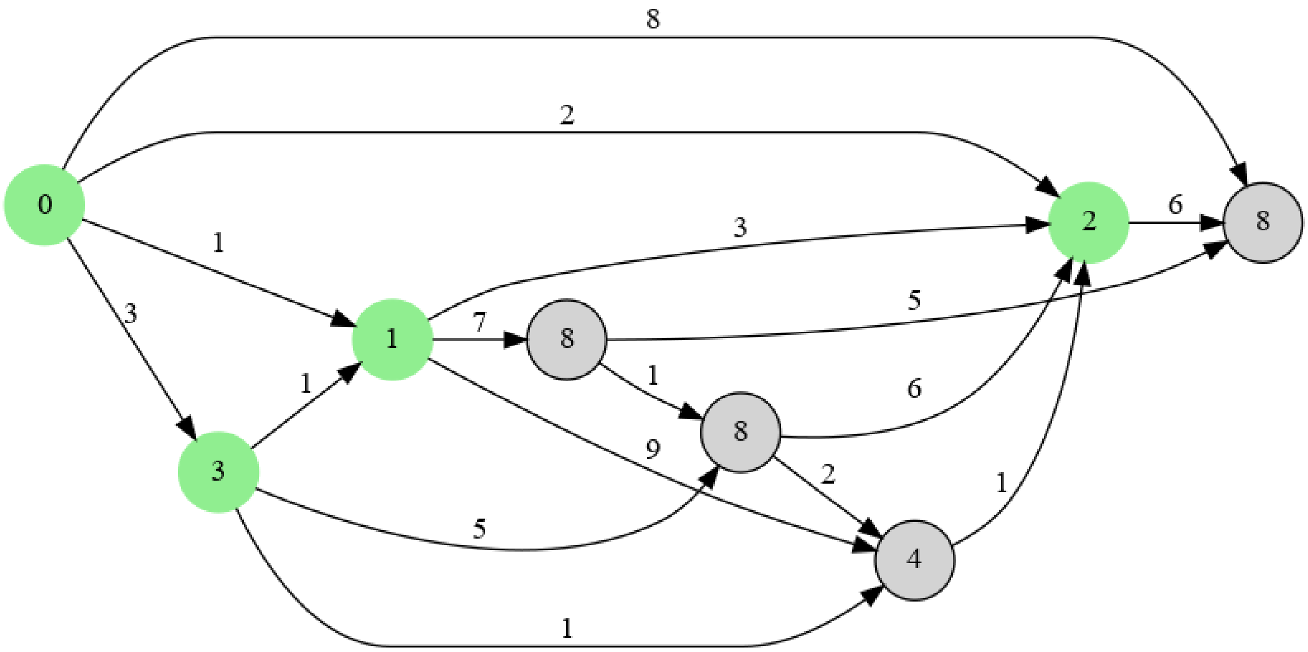

8
2
1
6
6
7
3
1
5
3
3
7
8
1
6
5
1
9
2
2
1
5
4
1

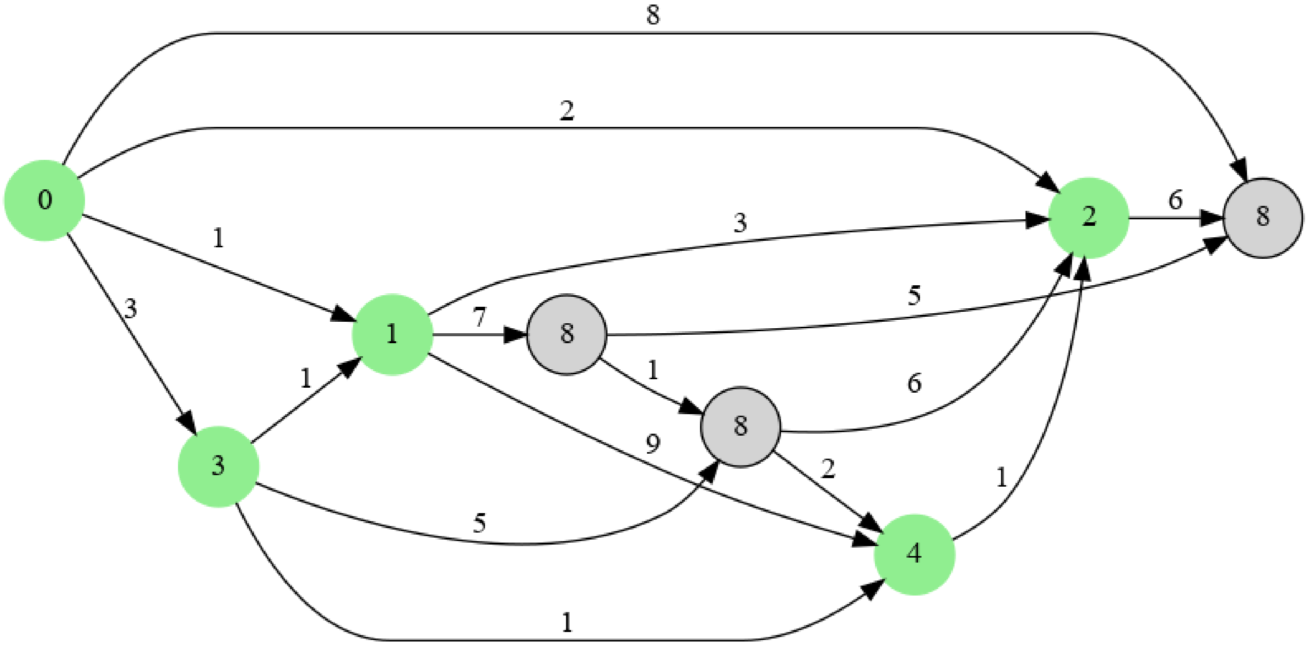

8
2
∞
∞
6
∞
3
1
5
3
∞
7
∞
1
6
∞
1
9
∞
2
1
5
∞
1

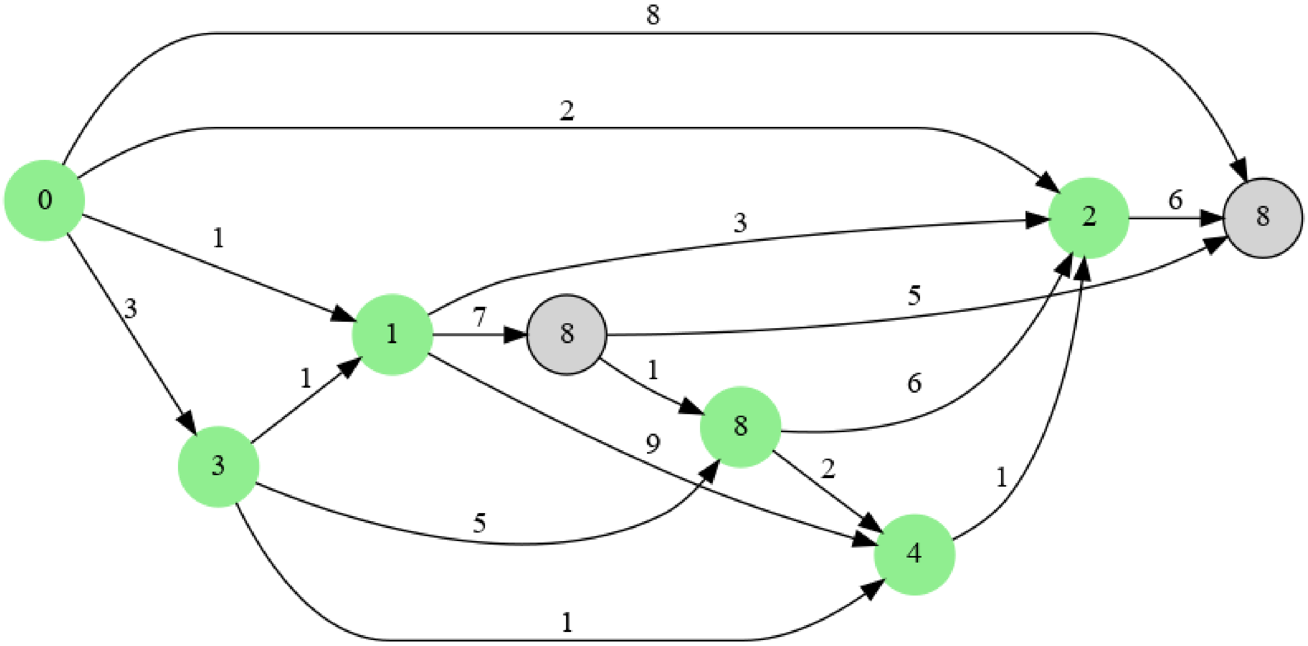

8
2
0
2
6
8
3
1
5
3
1
7
∞
1
6
∞
1
9
3
2
1
5
∞
1

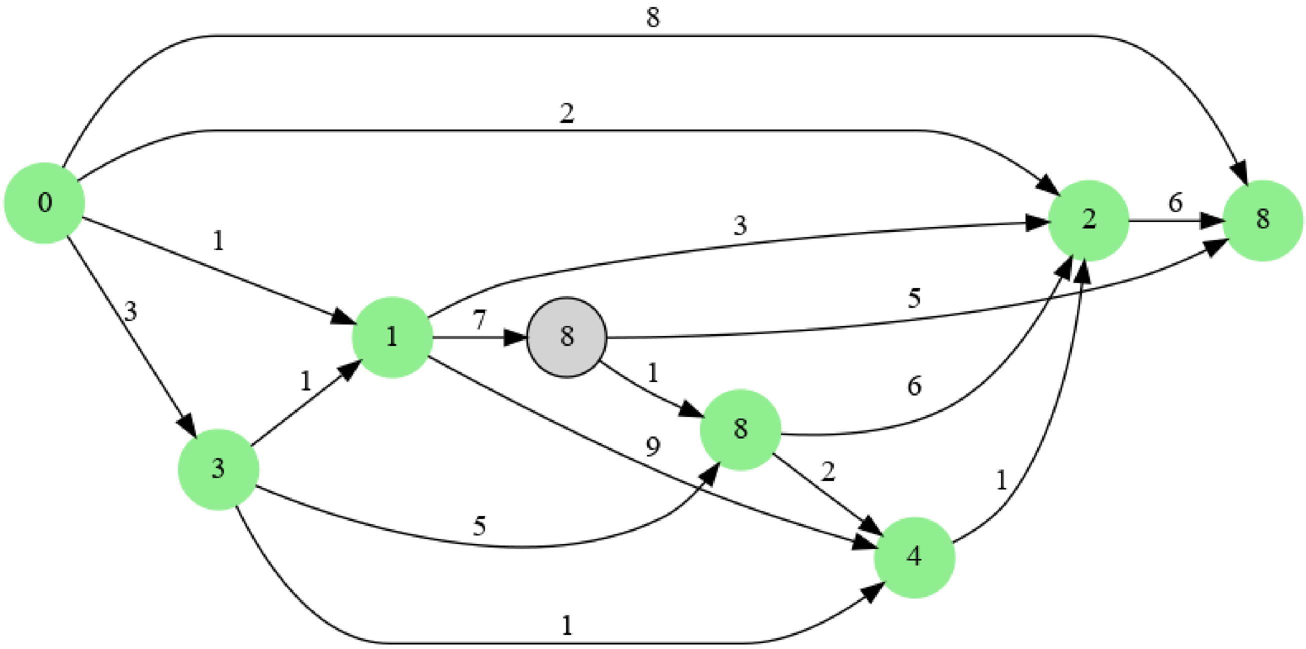

0
1
3
8
2
∞
10
2
8
8
2
1
3
1
7
3
6
5
1
6
9
2
1
5
1

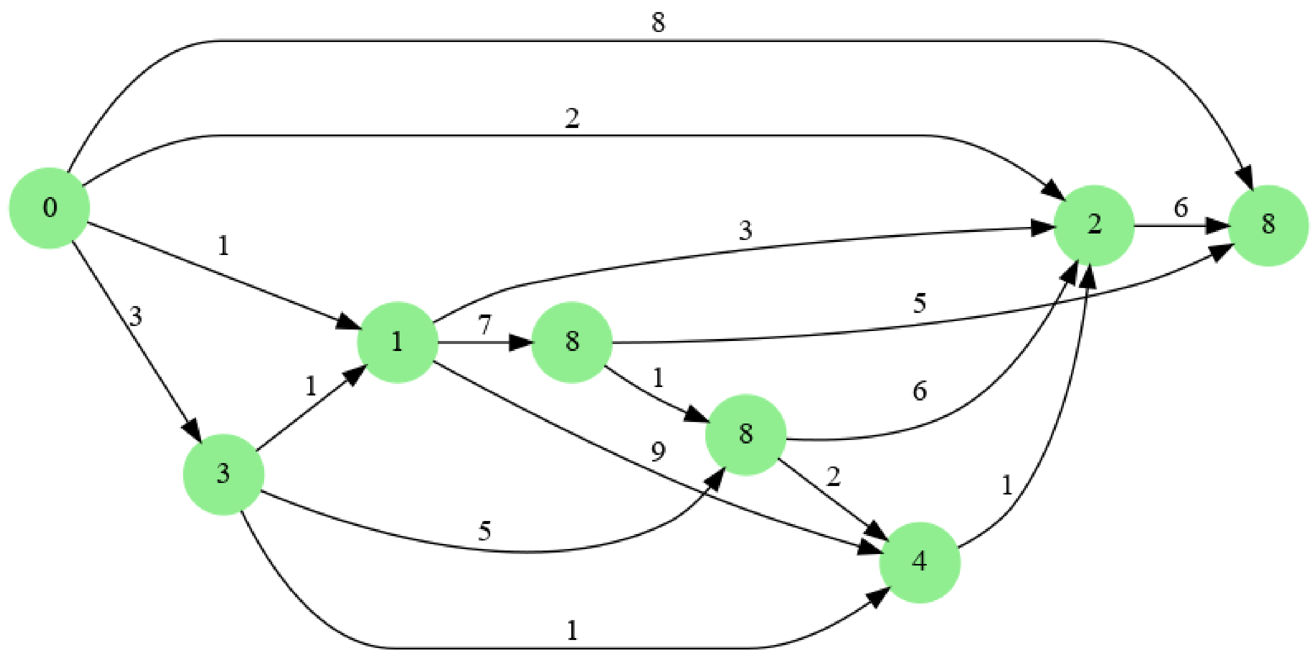

0
1
3
8
2
∞
10
2
8
8
2
1
3
1
7
3
6
5
1
6
9
2
1
5
1


0
1
3
8
8
4
2
8
8
2
1
3
1
7
3
6
5
1
6
9
2
1
5
1


8
2
0
3
2
6
8
1
1
7
8
5
3
1
8
6
9
2
1
5
4
1


8
2
0
3
2
6
8
1
1
7
8
5
3
1
8
6
9
2
1
5
4
1


8
2
0
3
2
6
8
1
1
7
8
5
3
1
8
6
9
2
1
5
4
1



3

Usted quiere irse de vacaciones y debe elegir una ciudad entre K posibles que le interesan. Como no dispone de mucho dinero, desea que el viaje de ida hacia la ciudad pueda realizarse con a lo sumo L litros de nafta.

b

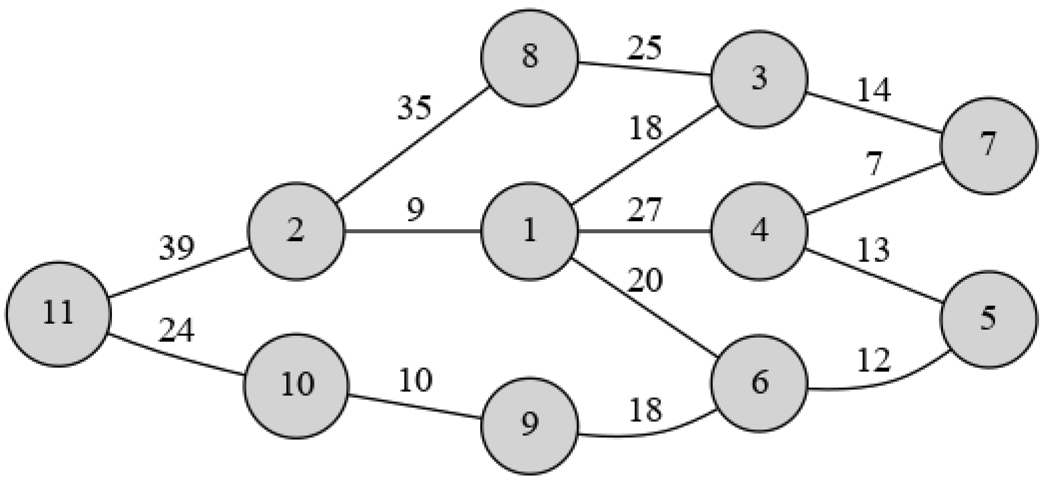
Dé un algoritmo que, dado un grafo representado por una matriz $E : array[1..n, 1..n]\ of\ Nat$, donde el elemento $E[i, j]$ indica el costo en litros de nafta necesario para ir desde la ciudad i hasta la ciudad j; un conjunto C de vértices entre 1 y n, representando las ciudades que quieren visitarse; un vértice v, representando la ciudad de origen del viaje; y un natural L, indicando la cantidad de litros de nafta total que puede gastar; devuelva un conjunto D de aquellos vértices de C que puede visitar con los L litros.

```
type Vertex = Nat

fun citiesToVisit(
        E: array[1..n, 1..n] of Nat,
        C: Set of Vertex,
        v: Vertex,
        L: Nat
) ret D: Set of Vertex
        var v_min, i_min: array[1..n] of Nat
        var C_copy: Set of Vertex
        var i: Vertex
        var l: Nat

        v_min := Dijkstra(E, v)
        C_copy := copy_set(C)
        D := empty_set()
```

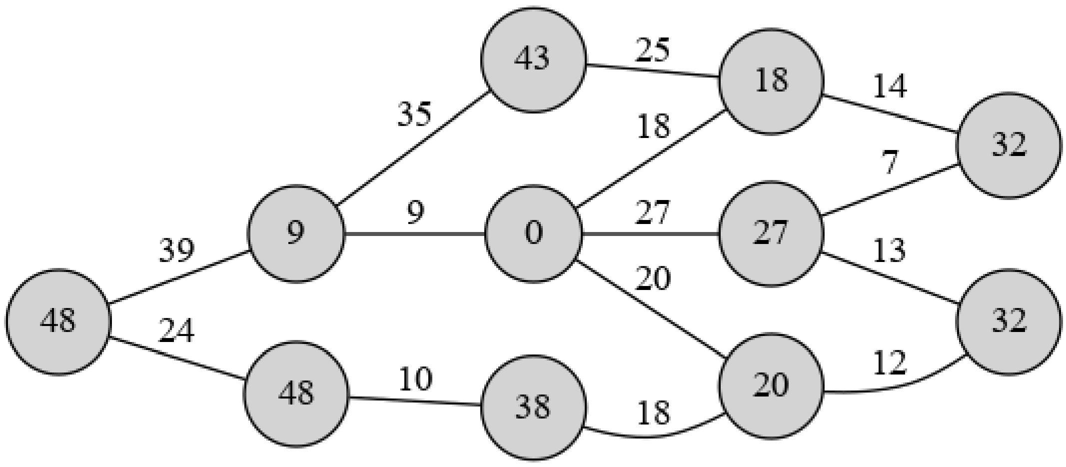
```
        while not is_empty_set(C_copy) do
                i := get(C_copy)
                elim(C_copy, i)

                l := v_min[i]

                if L >= l then
                        i_min := Dijkstra(E, i)
                        l := l + i_min[v]

                        if L >= l then
                                add(D, i)

        set_destroy(C_copy)
end fun


fun Dijkstra(
        L: array[1..n, 1..n] of Nat,
        v: Nat
) ret D: array[1..n] of Nat
        var c, c_aux: Nat
        var C, C_copy: Set of Nat

        C := empty_set()

        for i := 1 to n do
                add(C, i)

        elim(C, v)

        for j := 1 to n do
                D[j] := L[v, j]

        while not is_empty_set(C) do
                c := get(C)

                C_copy := copy_set(C)
                elim(C_copy, c)

                while not is_empty_set(C_copy) do
                        c_aux := get(C_copy)
                        elim(C_copy, c_aux)

                        if D[c_aux] < D[c] then
                                c := c_aux

                set_destroy(C_copy)
                elim(C, c)

                for j in C do
                        D[j] := min(D[j], D[c] + L[c, j])
end fun
```

b

Ejecute el algoritmo implementado en el inciso anterior para el grafo descripto en el siguiente gráfico, con vértices $1, 2, \ldots, 11$, tomando $C = \{11, 5, 10, 7, 8\}$ como las ciudades de interés, disponiendo de $L = 40$ litros de nafta. ¿Cuáles son los posibles destinos de acuerdo a su presupuesto?



v_min



i_min

Como es *no direccional*, $i\_min[v] = v\_min[i]$, $\forall i, v$

Resolución

Como $L = 40$, quedan descartados las ciudades 11, 10 y 8, y en cuanto a las ciudades 5 y 7, podría ir hasta esas ciudades, pero luego no tendría combustible suficiente para volver.