GRaphos Ciritical Infrastructure

Ricardo R. Palma

2024-05-15

Grafos de Infraestructura Crítica

Las infraestructuras críticas son sistemas, redes y facilidades físicas y virtuales esenciales para el funcionamiento seguro, económico y social de una sociedad. Estas infraestructuras abarcan una amplia gama de sectores, como energía, transporte, agua, comunicaciones, finanzas, salud y gobierno, entre otros. Son "críticas" debido a su importancia estratégica y al impacto significativo que su interrupción o degradación puede tener en la seguridad nacional, la economía y el bienestar público.

Las infraestructuras críticas están cada vez más vinculadas con la Industria 4.0 debido a la integración de tecnologías digitales avanzadas en su funcionamiento. La Industria 4.0 se refiere a la adopción de tecnologías como la Internet de las cosas (IoT), la inteligencia artificial (IA), el análisis de datos, la robótica avanzada y la fabricación aditiva en los procesos industriales. Estas tecnologías están transformando la forma en que se gestionan y operan las infraestructuras críticas, mejorando la eficiencia, la seguridad y la resiliencia.

Por ejemplo, en el sector de la energía, se están implementando sistemas de gestión de redes inteligentes que utilizan sensores y dispositivos IoT para monitorear y controlar la distribución de energía de manera más eficiente y segura. En el transporte, la utilización de sistemas de transporte inteligentes y vehículos autónomos puede mejorar la movilidad y reducir los riesgos de accidentes.

Sin embargo, esta convergencia también plantea desafíos en términos de ciberseguridad y privacidad, ya que las infraestructuras críticas se vuelven más interconectadas y dependientes de sistemas informáticos y de comunicación. Por lo tanto, es fundamental garantizar que se implementen medidas sólidas de ciberseguridad para proteger estas infraestructuras contra amenazas digitales.

Sin embrago existedefinitivamente vínculos entre las infraestructuras críticas y los "Digital Twins" (Gemelos Digitales). Los Digital Twins son representaciones virtuales en tiempo real de objetos físicos, sistemas o procesos. Estos modelos digitales pueden ser utilizados para simular, predecir y optimizar el rendimiento de infraestructuras físicas, lo que los hace especialmente relevantes para las infraestructuras críticas.

Aquí hay algunas formas en que los Digital Twins pueden estar vinculados con las infraestructuras críticas:

1. **Monitorización y mantenimiento predictivo:** Los Digital Twins pueden ser utilizados para monitorear continuamente el estado de componentes críticos de infraestructuras, como puentes, redes eléctricas o plantas de tratamiento de aguas. Esto permite identificar problemas potenciales antes de que ocurran, lo que facilita la realización de mantenimiento predictivo y la prevención de fallos.
2. **Resiliencia y planificación de emergencias:** Mediante la simulación de diferentes escenarios de crisis, como desastres naturales o ciberataques, los Digital Twins pueden ayudar a las autoridades y operadores de infraestructuras críticas a planificar respuestas efectivas y a mejorar la resiliencia de las infraestructuras frente a eventos adversos.
3. **Optimización de la eficiencia operativa:** Al analizar datos en tiempo real y simular diferentes estrategias operativas, los Digital Twins pueden ayudar a optimizar el rendimiento y la eficiencia de las infraestructuras críticas, como la gestión de flujos de tráfico en ciudades o la distribución de energía en redes eléctricas.

4. **Entrenamiento y formación:** Los Digital Twins también pueden ser utilizados para entrenar a operadores y personal de emergencias en situaciones realistas pero controladas, lo que les permite adquirir experiencia y practicar protocolos de respuesta sin poner en riesgo la infraestructura real.

Por estas razones, los Digital Twins ofrecen herramientas poderosas para mejorar la gestión, seguridad y eficiencia de las infraestructuras críticas, lo que las convierte en una tecnología prometedora en este ámbito.

IC - y métodos de grafos

En este texto desarrollaremos una serie de herramientas para visualizar las infraestructuras críticas y entender la interdependencia utilizando una aproximación de lenguajes matriciales. Recurriremos a la biblioteca *igraph* desde R-Cran.

Se puedes representar una red de infraestructuras críticas utilizando grafos. Los grafos son estructuras matemáticas que consisten en nodos (también llamados vértices) que están conectados por bordes (también llamados aristas). Cada nodo en el grafo puede representar una infraestructura crítica, mientras que las aristas representan las conexiones entre estas infraestructuras.

Aquí tienes algunos pasos para representar una red de infraestructuras críticas utilizando grafos:

1. **Identifica las infraestructuras críticas:** Haz una lista de las infraestructuras críticas que deseas incluir en tu red. Estas pueden ser instalaciones físicas como plantas de energía, centros de comunicaciones, hospitales, etc.
2. **Define las conexiones:** Determina cómo están interconectadas estas infraestructuras críticas. Las conexiones pueden ser físicas, como carreteras, líneas eléctricas o tuberías, o pueden ser conexiones virtuales, como enlaces de comunicación.
3. **Crea un grafo:** Utiliza software de representación de grafos como Igraph, Graphviz, NetworkX en Python, Gephi, o cualquier otra herramienta de tu elección para crear un grafo. Agrega un nodo para cada infraestructura crítica y una arista para cada conexión o interdependencia entre ellas.
4. **Asigna atributos:** Puedes asignar atributos a los nodos y aristas para representar información adicional sobre las infraestructuras críticas y sus conexiones. Por ejemplo, puedes agregar información sobre la capacidad de una planta de energía o el ancho de banda de una conexión de red.
5. **Visualiza el grafo:** Utiliza las herramientas de visualización del software que estés utilizando para representar gráficamente la red de infraestructuras críticas. Esto te permitirá ver de manera clara y visual cómo están interconectadas estas infraestructuras y cómo se comunican entre sí.

Al representar una red de infraestructuras críticas como un grafo, puedes obtener una comprensión más clara de su estructura y relaciones, lo que puede ayudarte a tomar decisiones informadas sobre su gestión, seguridad y resiliencia.

Riesgo y Resiliencia de IC

La resiliencia de una infraestructura crítica se refiere a su capacidad para resistir, adaptarse y recuperarse de perturbaciones o eventos adversos, manteniendo sus funciones esenciales y minimizando el impacto en la sociedad. Medir la resiliencia de una infraestructura crítica implica evaluar diversos aspectos de su capacidad para hacer frente a diferentes tipos de amenazas y perturbaciones. Aquí hay algunos enfoques comunes para medir la resiliencia de una infraestructura crítica:

1. **Análisis de vulnerabilidad:** Identifica las vulnerabilidades de la infraestructura y evalúa cómo pueden ser explotadas por amenazas específicas. Esto implica analizar la susceptibilidad de la infraestructura a eventos como desastres naturales, ciberataques, fallas técnicas, etc.

2. **Capacidad de resistencia:** Evalúa la capacidad de la infraestructura para resistir y mantener sus funciones esenciales durante una perturbación. Esto puede incluir medidas de redundancia, capacidad de carga, sistemas de respaldo, procedimientos de emergencia, etc.

3. **Capacidad de adaptación:** Evalúa la capacidad de la infraestructura para adaptarse a situaciones cambiantes y tomar medidas correctivas durante una perturbación. Esto puede implicar la flexibilidad en los procesos de operación, capacidad para implementar soluciones temporales, capacidad para ajustar las operaciones en tiempo real, etc.

4. **Tiempo de recuperación:** Evalúa el tiempo necesario para que la infraestructura se recupere completamente después de una perturbación y vuelva a sus niveles normales de funcionamiento. Esto implica identificar los procesos de recuperación, los recursos necesarios y los plazos asociados con la restauración de las funciones esenciales.

5. **Impacto en la sociedad:** Evalúa el impacto que una perturbación en la infraestructura tendría en la sociedad, incluyendo aspectos como la seguridad pública, la economía, la salud y el bienestar de las personas. Esto puede implicar la evaluación de los servicios críticos que proporciona la infraestructura y cómo su interrupción afectaría a la comunidad.

Estas son solo algunas formas de medir la resiliencia de una infraestructura crítica. Dependiendo del contexto específico y de los riesgos asociados, es posible que se requieran enfoques adicionales o medidas específicas para evaluar y mejorar la resiliencia de una infraestructura crítica.

Médición del riesgo

Para medir el riesgo de infraestructuras críticas, se utilizan una variedad de métodos matemáticos y técnicas de análisis de riesgos. Algunos de los métodos más comunes incluyen:

1. **Análisis de Árbol de Fallas (FTA, por sus siglas en inglés):** El FTA es un método sistemático para identificar y evaluar las posibles causas de un evento no deseado. Se utiliza para analizar la cadena de eventos que pueden llevar a una falla en la infraestructura crítica y calcular la probabilidad de que ocurra el evento no deseado.

2. **Análisis de Modo de Falla y Efecto (FMEA, por sus siglas en inglés):** El FMEA es un enfoque estructurado para identificar y evaluar los posibles modos de falla de un sistema y los efectos de esas fallas en el funcionamiento del sistema. Se utiliza para evaluar el impacto de posibles fallas en la infraestructura crítica y priorizar acciones de mitigación.

3. **Simulación de Monte Carlo:** La simulación de Monte Carlo es una técnica estadística que se utiliza para modelar la incertidumbre en sistemas complejos y calcular la probabilidad de diferentes resultados. Se puede utilizar para evaluar el riesgo de eventos adversos en la infraestructura crítica y estimar la probabilidad de pérdidas económicas o de otro tipo.

4. **Análisis de Sensibilidad:** El análisis de sensibilidad se utiliza para evaluar cómo varían los resultados de un modelo en respuesta a cambios en sus entradas o parámetros. Se puede utilizar para identificar las variables más influyentes en el riesgo de la infraestructura crítica y priorizar acciones de mitigación.

5. **Modelos de Confiabilidad:** Los modelos de confiabilidad se utilizan para evaluar la confiabilidad y disponibilidad de sistemas complejos. Se pueden utilizar para calcular la probabilidad de que la infraestructura crítica falle en un período de tiempo dado y estimar el riesgo asociado con esa falla.

Estos son solo algunos de los métodos matemáticos comúnmente utilizados para medir el riesgo de infraestructuras críticas. Dependiendo del contexto específico y de los requisitos de análisis, pueden utilizarse otros enfoques y técnicas complementarias.

Todos estos métodos afectan el factor de peso del grafo y pueden visualizarse tal como hemos visto.

Para representar matricialmente un grafo de riesgo de infraestructura crítica, podemos utilizar una matriz de adyacencia ponderada. Esta matriz representará las conexiones entre las infraestructuras críticas, así como los niveles de riesgo asociados con esas conexiones. Aquí hay un enfoque general para construir esta matriz:

1. **Identificar nodos:** Enumera todas las infraestructuras críticas como nodos en el grafo y asigna un índice único a cada una.
2. **Crear la matriz de adyacencia:** Crea una matriz cuadrada $n \times n$, donde n es el número de nodos (infraestructuras críticas). Inicialmente, esta matriz estará llena de ceros.
3. **Asignar pesos:** Para cada conexión entre nodos, asigna un peso que represente el nivel de riesgo asociado con esa conexión. Si no hay conexión entre dos nodos, el peso se establece en cero. Si hay una conexión, el peso puede ser una medida del riesgo, como la probabilidad de falla o el impacto de una falla en la otra infraestructura.
4. **Llenar la matriz:** Para cada conexión entre nodos, actualiza el valor correspondiente en la matriz de adyacencia con el peso asignado en el paso anterior.
5. **Representación matricial:** Una vez que la matriz de adyacencia esté completa, tendrás una representación matricial del grafo de riesgo de infraestructura crítica. Cada fila y columna de la matriz corresponde a una infraestructura crítica, y los valores en la matriz representan los niveles de riesgo asociados con las conexiones entre esas infraestructuras.

Por ejemplo, si tienes tres infraestructuras críticas (A, B, C) y sabes que la conexión entre A y B tiene un alto riesgo, la matriz de adyacencia podría verse así:

```
     A    B    C
A    0   0.8   0
B   0.8   0    0
C    0    0    0
```

En esta matriz, el valor 0.8 indica un alto riesgo en la conexión entre A y B, mientras que los valores 0 representan la falta de conexión o riesgo entre las otras infraestructuras.

Esta representación matricial te permite realizar análisis numéricos y cálculos relacionados con el riesgo de la infraestructura crítica utilizando técnicas matriciales estándar.

Técnicas MST (Máximun Expand Tree)

El Maximum Spanning Tree (MST), o árbol de expansión máximo, es un subgrafo de un grafo ponderado no dirigido que conecta todos los vértices sin formar ciclos y tiene el mayor peso total posible. En tu caso, dado que estás tratando con una matriz de adyacencia ponderada que representa un grafo de riesgo de infraestructura crítica, el Maximum Spanning Tree representaría la conexión más crítica entre las infraestructuras.

Para encontrar el Maximum Spanning Tree de esta matriz, puedes utilizar algoritmos como el algoritmo de Kruskal o el algoritmo de Prim. Estos algoritmos te permiten encontrar eficientemente el árbol de expansión máximo en un grafo ponderado.

Dado que ya tienes la matriz de adyacencia ponderada, puedes aplicar uno de estos algoritmos para encontrar el Maximum Spanning Tree y determinar la conexión más crítica entre las infraestructuras. Este árbol te proporcionará una representación de la red de infraestructura crítica que maximiza la resiliencia al conectar las infraestructuras críticas de manera óptima, minimizando el riesgo total de la red.

En R, puedes utilizar la librería "igraph" para trabajar con grafos y calcular el Maximum Spanning Tree (MST). Esta librería proporciona varias funciones para trabajar con grafos, incluida la función para calcular el MST.

Aquí hay un ejemplo de cómo calcular el MST utilizando la función `mst` de la librería "igraph" en R:

```r
# Instalar e importar la librería
## Si es necesario ejecuta install.packages("igraph")
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

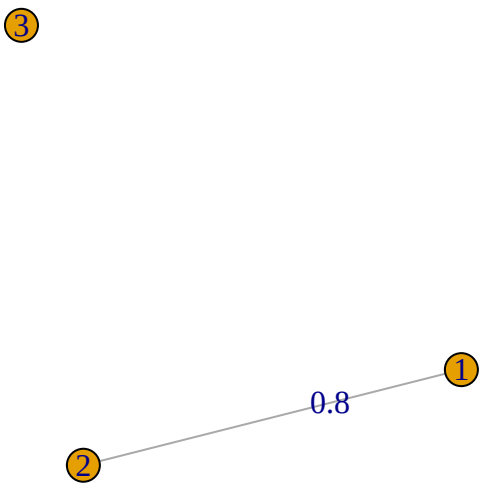
```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```r
# Crear un grafo con la matriz de adyacencia ponderada
matriz <- matrix(c(0, 0.8, 0,
                   0.2, 0, 0,
                   0, 0, 0), nrow = 3, byrow = TRUE)

# Convertir la matriz en un grafo
grafo <- graph.adjacency(matriz, mode = "undirected", weighted = TRUE)
```

```
## Warning: `graph.adjacency()` was deprecated in igraph 2.0.0.
## i Please use `graph_from_adjacency_matrix()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
# Calcular el Maximum Spanning Tree
mst <- mst(grafo)

# Mostrar el resultado
plot(mst, edge.label = E(mst)$weight)
```



En este código, primero instalamos y cargamos la librería "igraph". Luego creamos un grafo a partir de la matriz de adyacencia ponderada utilizando la función `graph.adjacency`. Después, utilizamos la función `mst` para calcular el MST del grafo. Finalmente, visualizamos el MST utilizando la función `plot`.

Este es solo un ejemplo básico de cómo calcular el MST en R utilizando la librería "igraph". Puedes ajustar este código según tus necesidades específicas y la estructura de tu matriz de adyacencia ponderada.

Maximun and Minimun Span Tree

El Maximum Spanning Tree (MST) y el Minimum Spanning Tree (MST) son conceptos relacionados pero con objetivos opuestos en términos de ponderación en un grafo. Aquí tienes las diferencias clave entre ambos:

1. **Objetivo:**
   - **MST:** El Minimum Spanning Tree busca encontrar un árbol que conecte todos los vértices del grafo con el peso total mínimo posible.
   - **MST:** El Maximum Spanning Tree busca encontrar un árbol que conecte todos los vértices del grafo con el peso total máximo posible.
2. **Ponderación:**
   - **MST:** En el Minimum Spanning Tree, los pesos de las aristas se minimizan.
   - **MST:** En el Maximum Spanning Tree, los pesos de las aristas se maximizan.
3. **Utilidad:**
   - **MST:** El Minimum Spanning Tree es útil en situaciones donde deseas encontrar la conexión más eficiente y económica entre todos los vértices de un grafo, como en redes de transporte o distribución.
   - **MST:** El Maximum Spanning Tree puede ser útil en situaciones donde deseas encontrar la conexión más crítica o robusta entre todos los vértices de un grafo, como en el caso de redes de infraestructuras críticas.

Ambos conceptos son fundamentales en teoría de grafos y se utilizan en una variedad de aplicaciones en ciencia de la computación, ingeniería y ciencias sociales para modelar y analizar relaciones entre objetos o entidades. La elección entre un MST y un MST depende del contexto específico y del objetivo del análisis.

Otros métodos y algoritmos para IC

Además del cálculo del Minimum Spanning Tree (MST) o del Maximum Spanning Tree (MST), hay una serie de algoritmos y técnicas de análisis de grafos que pueden ser útiles en el análisis de redes de infraestructura crítica. Aquí hay algunos ejemplos:

1. **Algoritmos de flujo máximo:** Estos algoritmos, como el algoritmo de Ford-Fulkerson o el algoritmo de Edmonds-Karp, se utilizan para calcular el flujo máximo en una red, lo que puede ser útil para determinar la capacidad máxima de una red de infraestructura crítica y para identificar cuellos de botella.
2. **Algoritmos de camino más corto:** Algoritmos como Dijkstra o Bellman-Ford se utilizan para encontrar los caminos más cortos entre dos nodos en un grafo ponderado. Esto puede ser útil en la planificación de rutas eficientes para la distribución de recursos o para la respuesta a emergencias en una red de infraestructura crítica.
3. **Algoritmos de clustering:** Estos algoritmos, como el algoritmo de Girvan-Newman o el algoritmo de Louvain, se utilizan para identificar comunidades o grupos de nodos densamente conectados en una red. Esto puede ayudar a identificar subgrupos funcionales dentro de una red de infraestructura crítica y a entender mejor su estructura y dinámica.

4. **Análisis de centralidad:** Métricas de centralidad como la centralidad de intermediación, la centralidad de cercanía y la centralidad de grado se utilizan para identificar nodos importantes en una red. Estos nodos pueden ser críticos para el funcionamiento y la resiliencia de la red de infraestructura crítica.

5. **Análisis de comunidades:** Estas técnicas se utilizan para identificar comunidades o grupos de nodos altamente interconectados en una red. Esto puede ser útil para comprender la estructura y la dinámica de una red de infraestructura crítica y para identificar posibles puntos de vulnerabilidad o áreas de mayor riesgo.

Estos son solo algunos ejemplos de algoritmos y técnicas de análisis de grafos que pueden ser útiles en el análisis de redes de infraestructura crítica. La elección de algoritmos dependerá del objetivo específico del análisis y de las características particulares de la red en cuestión.

Caso de estudio

En nuestro ejemplo utilizaremos cinco infraestructuras, a saber:

1. **A** Energía Eléctrica
2. **B** Combustibles Fósiles
3. **C** Comunicaciones
4. **D** Transporte
5. **E** Vías de comunicación (caminos - ríos)
6. **F** Alimentos y Agua

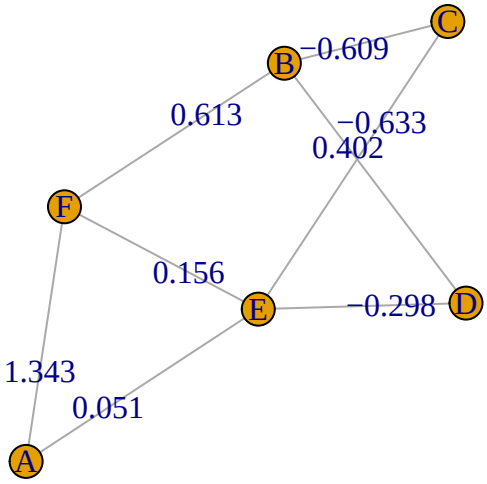
Construiremos una matriz de adyacencia (como las que se utilizan en investigación operativa). Esta matriz

```
m <- read.table(row.names=1, header=TRUE, text=
"           A          B          C          D           E         F
A 0.00000000  0.0000000  0.0000000  0.0000000  0.05119703 1.3431599
B 0.00000000  0.0000000 -0.6088082  0.4016954  0.00000000 0.6132168
C 0.00000000 -0.6088082  0.0000000  0.0000000 -0.63295415 0.0000000
D 0.00000000  0.4016954  0.0000000  0.0000000 -0.29831267 0.0000000
E 0.05119703  0.0000000 -0.6329541 -0.2983127  0.00000000 0.1562458
F 1.34315990  0.6132168  0.0000000  0.0000000  0.15624584 0.0000000")
m <- as.matrix(m)
```

```
library(igraph)
ig <- graph.adjacency(m, mode="undirected", weighted=TRUE)
```

```
plot(ig, edge.label=round(E(ig)$weight, 3))
```

```
## Warning in v(graph): Non-positive edge weight found, ignoring all weights
## during graph layout.
```

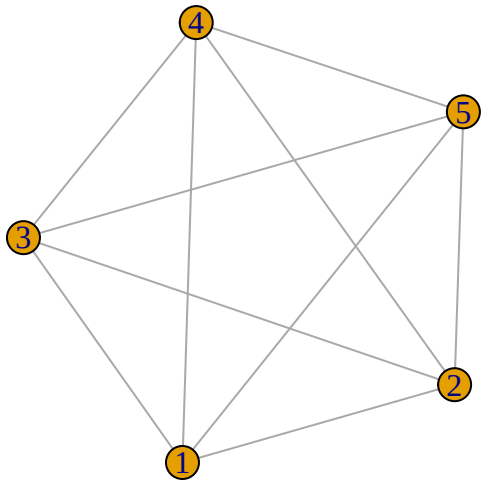



```
# reproducible example:
set.seed(12345)
a <- matrix(runif(5*5, min=-10, max=10), ncol=5)
diag(a) <- 0 # remove loops.
a
```

```
##            [,1]       [,2]      [,3]       [,4]       [,5]
## [1,]  0.0000000 -6.6725643 -9.309291 -0.7501069 -0.9254385
## [2,]  7.5154639  0.0000000 -6.952530 -2.2371204 -3.4649518
## [3,]  5.2196466  0.1844867  0.000000 -1.9502972  9.3083065
## [4,]  7.7224913  4.5541051 -9.977268  0.0000000  4.1496375
## [5,] -0.8703808  9.7947388 -2.175933  9.0331751  0.0000000
```

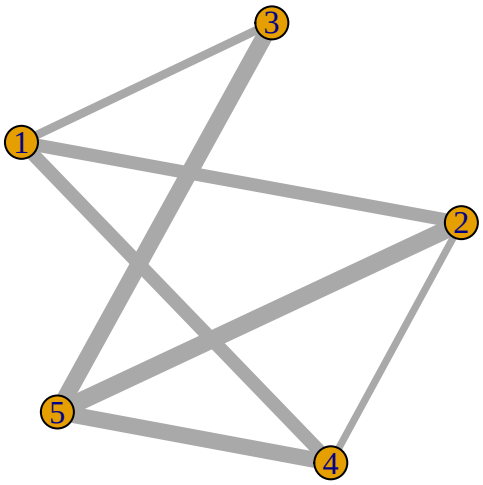
```
# create igraph object.
g <- graph.adjacency(a, mode="undirected",weighted=TRUE)
plot(g)
```

```
## Warning in v(graph): Non-positive edge weight found, ignoring all weights
## during graph layout.
```



```r
# assign edge's width as a function of weights.
E(g)$width <- E(g)$weight + min(E(g)$weight) + 1 # offset=1
plot(g)
```

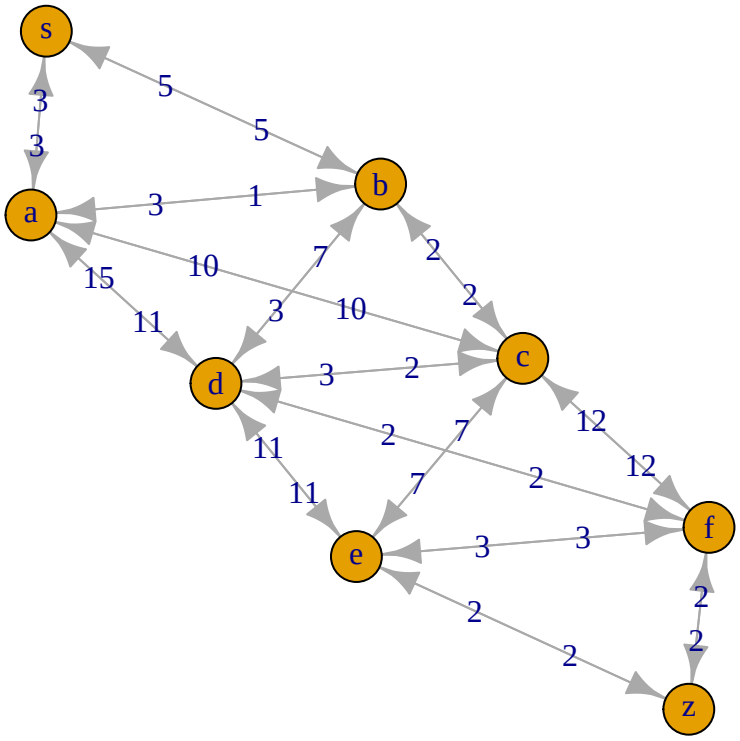
```
## Warning in v(graph): Non-positive edge weight found, ignoring all weights
## during graph layout.
```



Ejemplo de algoritmo de Dijktra en Grafo de R

```r
library(igraph)
## Attaching package: 'igraph'
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
## The following object is masked from 'package:base':
##
##     union
graph <- list(s = c("a", "b"),
              a = c("s", "b", "c", "d"),
              b = c("s", "a", "c", "d"),
              c = c("a", "b", "d", "e", "f"),
              d = c("a", "b", "c", "e", "f"),
              e = c("c", "d", "f", "z"),
              f = c("c", "d", "e", "z"),
              z = c("e", "f"))

weights <- list(s = c(3, 5),
                a = c(3, 1, 10, 11),
                b = c(5, 3, 2, 3),
                c = c(10, 2, 3, 7, 12),
                d = c(15, 7, 2, 11, 2),
                e = c(7, 11, 3, 2),
                f = c(12, 2, 3, 2),
                z = c(2, 2))
```

```r
# create edgelist with weights
G <- data.frame(stack(graph), weights = stack(weights)[[1]])

set.seed(500)
el <- as.matrix(stack(graph))
g <- graph_from_edgelist(el)

oldpar <- par(mar = c(1, 1, 1, 1))
plot(g, edge.label = stack(weights)[[1]])
```




```r
par(oldpar)
```

crearemos una funcunción para Cálculo de largo del camino

```r
path_length <- function(path) {
  # if path is NULL return infinite length
  if (is.null(path)) return(Inf)

  # get all consecutive nodes
  pairs <- cbind(values = path[-length(path)], ind = path[-1])
  # join with G and sum over weights
  sum(merge(pairs, G)[ , "weights"])
}
```

Y ahora, el meollo del asunto, el algoritmo de Dijkstra: la idea general del algoritmo es muy simple y elegante: comience en el nodo inicial y llame al algoritmo recursivamente para todos los nodos vinculados desde allí como nuevos nodos iniciales y así construya su paso de ruta. Por paso. Mantenga solo el camino más corto y deténgase cuando llegue al nodo final (caso base de la recursividad). En caso de que llegue a un callejón sin salida en el medio, asigne infinito como longitud (mediante la función path_length anterior).

Agregué mucha documentación al código para que sea posible entender cómo funciona:

```
find_shortest_path <- function(graph, start, end, path = c()) {
  # if there are no nodes linked from current node (= dead end) return NULL
  if (is.null(graph[[start]])) return(NULL)
  # add next node to path so far
  path <- c(path, start)

  # base case of recursion: if end is reached return path
  if (start == end) return(path)

  # initialize shortest path as NULL
  shortest <- NULL
  # loop through all nodes linked from the current node (given in start)
  for (node in graph[[start]]) {
    # proceed only if linked node is not already in path
    if (!(node %in% path)) {
      # recursively call function for finding shortest path with node as start and assign it to newpath
      newpath <- find_shortest_path(graph, node, end, path)
      # if newpath is shorter than shortest so far assign newpath to shortest
      if (path_length(newpath) < path_length(shortest))
        shortest <- newpath
    }
  }
  # return shortest path
  shortest
}
```

Probaremos el algoritmo

```
find_shortest_path(graph, "s", "z") # via b
```

```
## [1] "s" "b" "c" "d" "f" "z"
```

```
## [1] "s" "b" "c" "d" "f" "z"

find_shortest_path(graph, "z", "s") # back via a
```

```
## [1] "z" "f" "d" "b" "a" "s"
```

```
## [1] "z" "f" "d" "b" "a" "s"
```

Tenga en cuenta que las dos rutas en realidad son diferentes debido a los diferentes pesos en ambas direcciones (por ejemplo, piense en algún trabajo de construcción en una dirección pero no en la otra).

Análisis de Centralidad

En R, también puedes realizar un análisis de centralidad en una red de infraestructura crítica utilizando la librería "igraph". Esta librería proporciona una amplia gama de funciones para analizar grafos, incluida la capacidad de calcular diferentes medidas de centralidad. Aquí te muestro cómo puedes hacerlo:

Instalar y cargar la librería igraph:

Crear un grafo: Puedes crear un grafo representando tu red de infraestructura crítica. Puedes utilizar la función graph.data.frame() para crear un grafo a partir de un marco de datos que contenga información sobre las conexiones entre las infraestructuras.

```
# Ejemplo de creación de un grafo
grafo <- graph.data.frame(data.frame(from = c("A", "B", "C", "A"),
                                     to = c("B", "C", "D", "D")))
```

```
## Warning: `graph.data.frame()` was deprecated in igraph 2.0.0.
## i Please use `graph_from_data_frame()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

Calcular medidas de centralidad:

Utiliza las funciones de la librería "igraph" para calcular medidas de centralidad en el grafo que has creado. Por ejemplo, puedes calcular la centralidad de grado, la centralidad de intermediación (betweenness centrality), la centralidad de cercanía (closeness centrality), etc.

```
# Ejemplo de cálculo de centralidad de grado
grado <- degree(grafo)
grado
```

```
## A B C D
## 2 2 2 2
```

```
# Ejemplo de cálculo de centralidad de intermediación
intermediacion <- betweenness(grafo)
intermediacion
```

```
## A B C D
## 0 1 1 0
```

```
# Ejemplo de cálculo de centralidad de cercanía
cercania <- closeness(grafo)
cercania
```

```
##         A         B         C         D
## 0.2500000 0.3333333 1.0000000       NaN
```

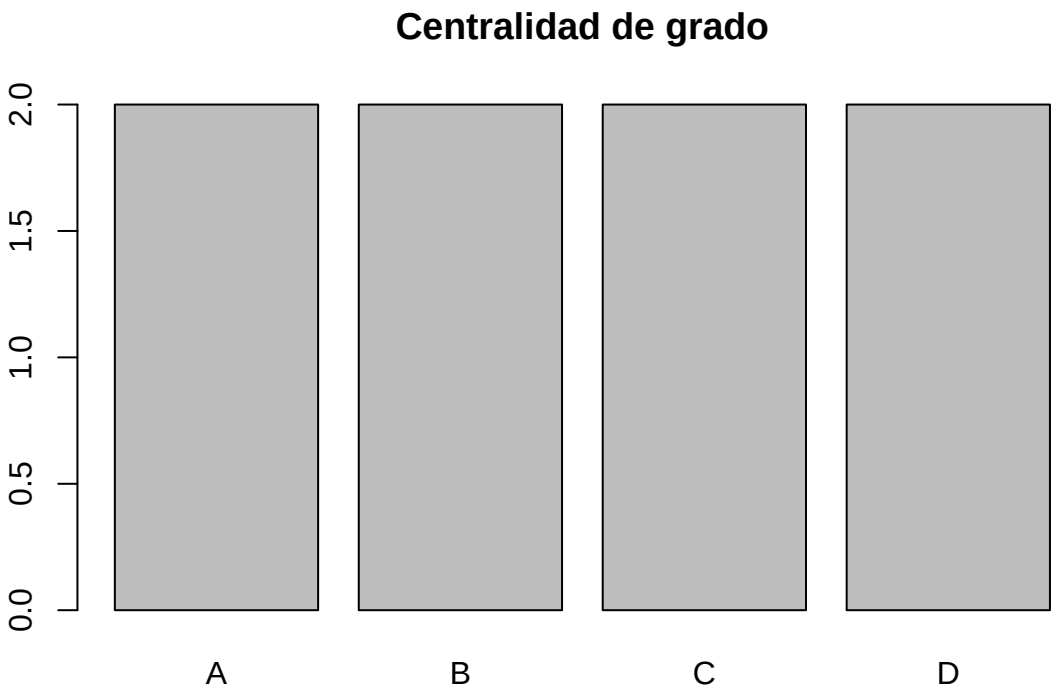
Ejemplo de cálculo de centralidad de intermediación

```
# Ejemplo de cálculo de centralidad de grado
grado <- degree(grafo)

# Ejemplo de cálculo de centralidad de intermediación
intermediacion <- betweenness(grafo)

# Ejemplo de cálculo de centralidad de cercanía
cercania <- closeness(grafo)
```

Visualizar los resultados:

Puedes visualizar los resultados del análisis de centralidad utilizando gráficos u otras representaciones visuales. Por ejemplo, puedes crear un gráfico de barras para mostrar la centralidad de grado de cada nodo en el grafo.

```
# Ejemplo de visualización de la centralidad de grado
barplot(grado, names.arg = V(grafo)$name, main = "Centralidad de grado")
```



Esto es solo un ejemplo básico de cómo puedes realizar un análisis de centralidad en una red de infraestructura crítica utilizando R y la librería "igraph". Puedes ajustar y personalizar este código según tus necesidades específicas y la estructura de tu red.