



Diseño de Algoritmos

Sesión 11

Profesores:

Tomás Lara Valdovinos – t.lara@uandresbello.edu

Jessica Meza-Jaque – je.meza@uandresbello.edu

OBJETIVOS DE LA SESIÓN

- Conocer los conceptos: camino, grafo completo, ciclo y recorrido
- Conocer el concepto de búsqueda en anchura como forma de resolución de problemas utilizando grafos
- Conocer el concepto de búsqueda en profundidad como forma de resolución de problemas utilizando grafos



CONTENIDOS DE LA SESIÓN



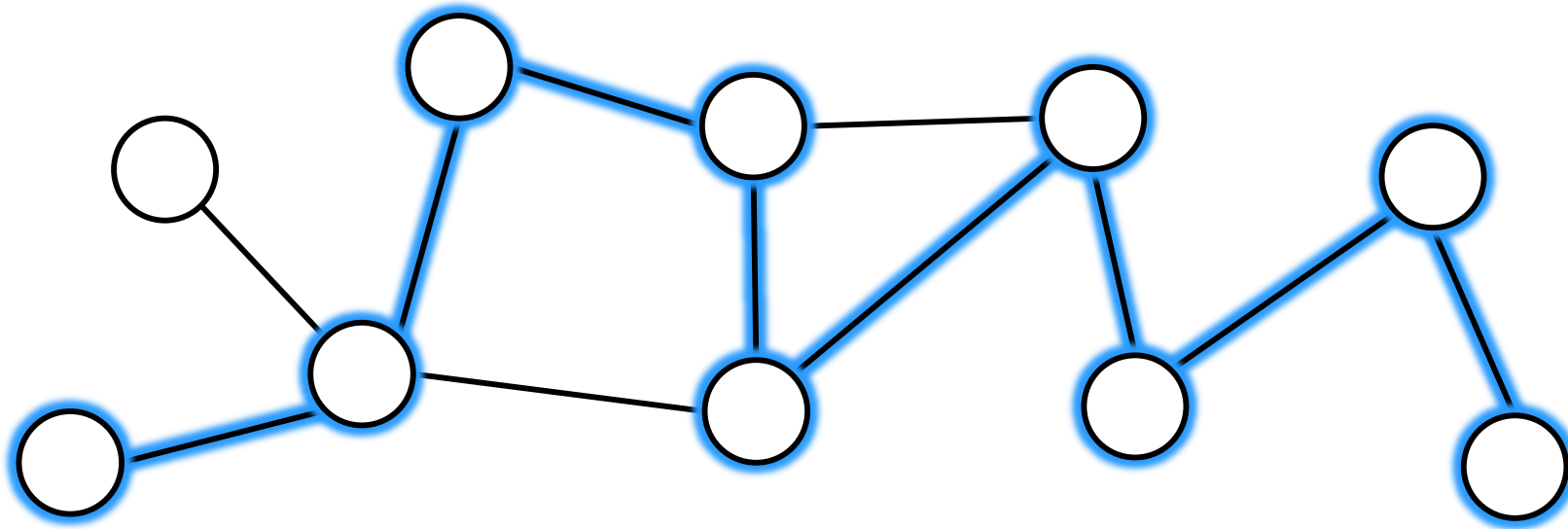
- Concepto camino, grafo completo, ciclo y recorrido
- Concepto búsqueda en anchura
- Concepto búsqueda en profundidad
- Ejemplos

Recordemos...

¿Qué es un grafo? ¿Qué características tienen?
¿Para qué los podemos utilizar?

Camino en grafos

- Llamaremos **camino** a una serie de nodos tal que exista una arista entre cada nodo y el siguiente.



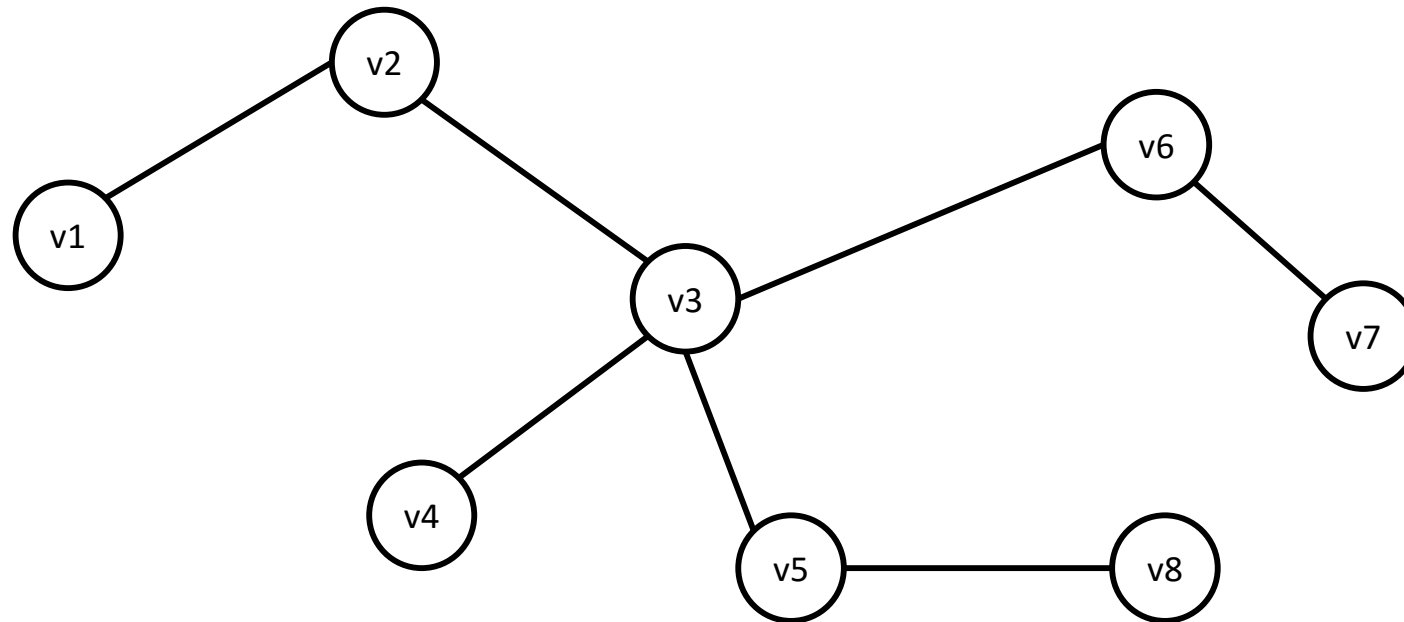
Ejemplo: Los nodos y aristas marcados son un ejemplo de camino

Camino en grafos

- Se aplica tanto a **grafos dirigidos y no-dirigidos**
- Se aplica a grafos **ponderados y no ponderados**
- También podemos llamarlas **Rutas**

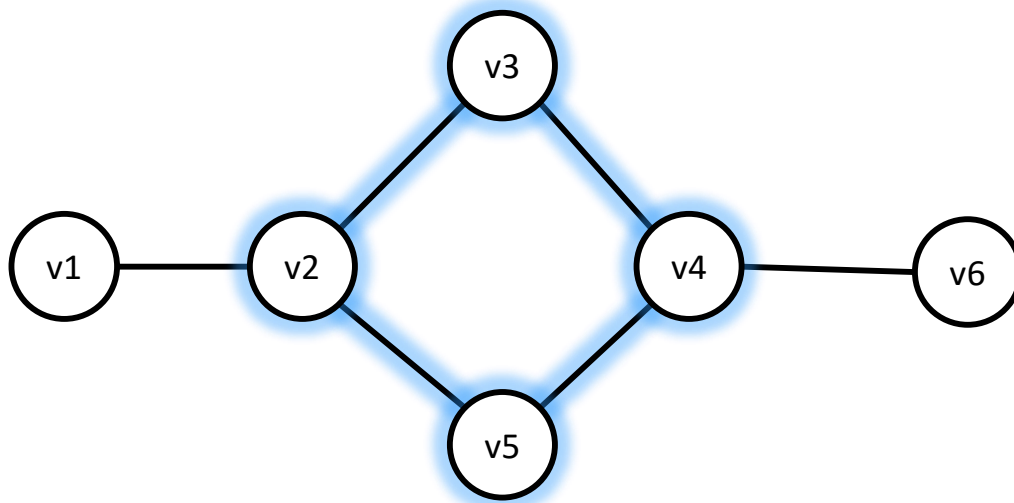
Grafos Completo

- Un grafo tiene la propiedad de ser completo, o completamente conectado, si para cualquier nodo en él, existe un camino hacia cualquier otro nodo del grafo.

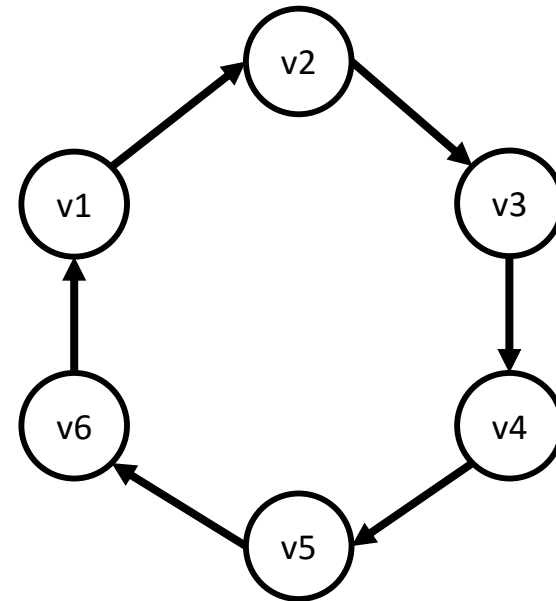


Ciclo

- Un **ciclo** es un camino el cual empieza y termina en el mismo nodo



Ciclo no dirigido



Ciclo dirigido

Recorrido de grafos

- Es la acción de **visitar** los nodos en un grafo tanto para:
 - obtener su propiedad,
 - buscar caminos contenidos o
 - buscar elementos.
- Existen diversos algoritmos para recorrer grafos, como:
 - Búsqueda en profundidad (Depth-First Search)
 - Búsqueda en anchura (Breadth-First Search)

Búsqueda en profundidad

- Abreviado DFS, del inglés Depth-First Search
- Permite recorrer un grafo o árbol de manera ordenada, pero no uniforme.
- Utiliza **backtracking** y es un **algoritmo recursivo**.

Algoritmo en pseudocódigo

```
DFS (G, s)
  foreach v in V do
    v.color = White

  DFS_visit(s)

  foreach v in V do
    if (v.color = White) then
      DFS_visit(v)
end
```

```
DFS_visit (u)
  u.color = Gray

  foreach neighbor v of u do
    if v.color = White then
      v.pred = u
      DFS_visit(v)

  u.color = Black
end
```

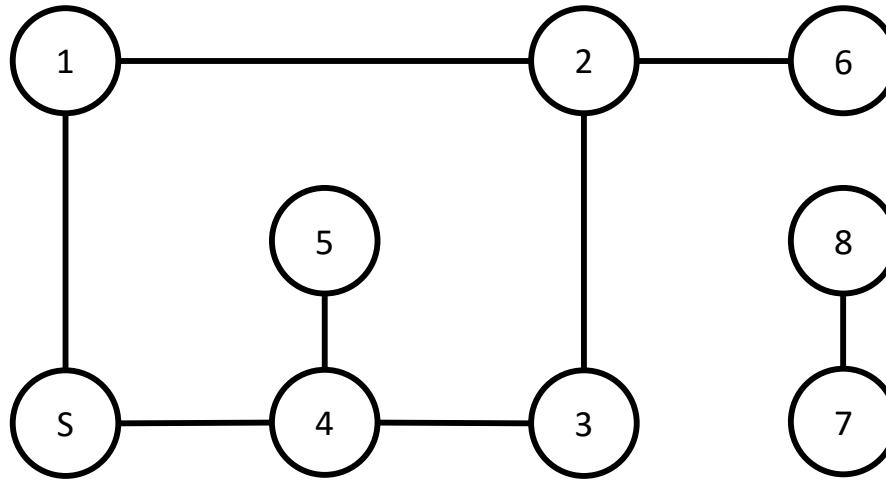
Explicación de los colores

- Blanco (White) : el nodo no ha sido visitado, al inicio del código todos los nodos son marcados con este color.
- Gris (Gray): El nodo ha sido visitado
- Negro (Black): El nodo y todos sus vecinos han sido visitados.

Estado final del grafo

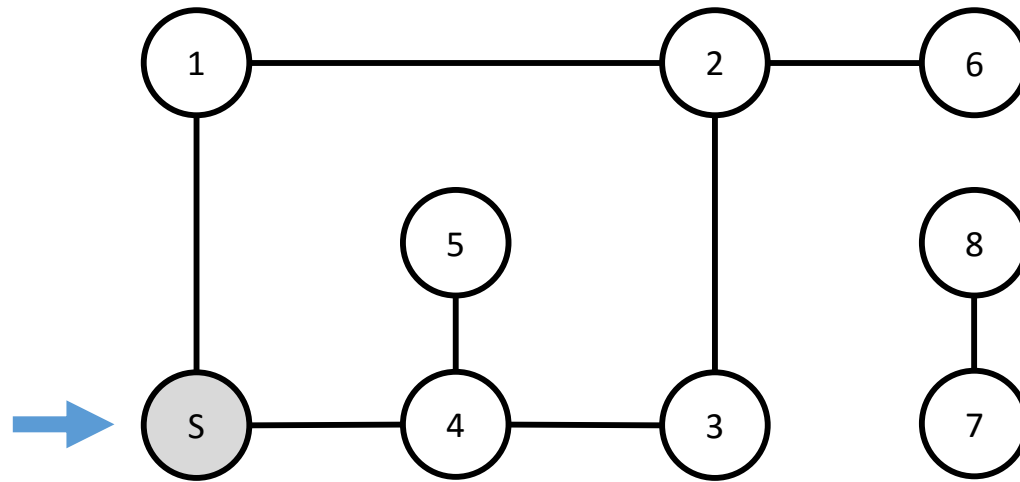
- Al finalizar la ejecución del algoritmo cada nodo quedará asociado al nodo al que está conectado directamente, indicando la ruta de llegada a éste.

Ejemplo de ejecución



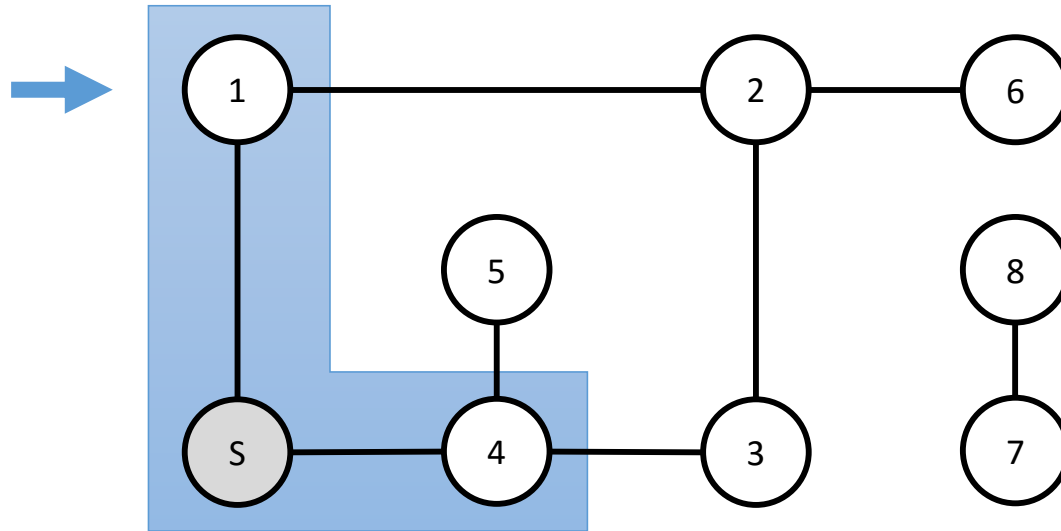
1.- Se marcan todos los nodos con el color blanco

Ejemplo de ejecución



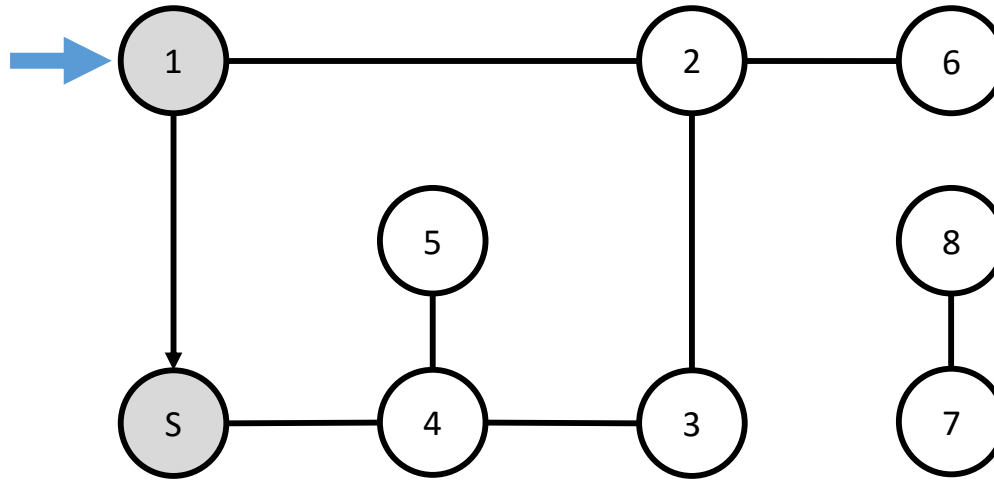
2.- Continuamos marcando en gris el nodo de inicio "S"

Ejemplo de ejecución



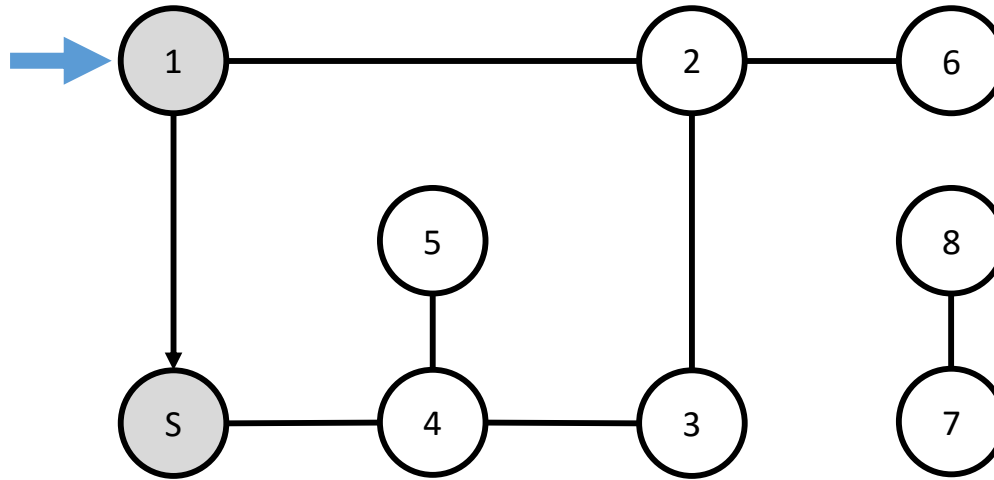
- 3.- Recorremos los vecinos blancos de S, partiendo con el nodo etiquetado 1
- 4.- Realizamos una llamada recursiva para el primer nodo vecino

Ejemplo de ejecución



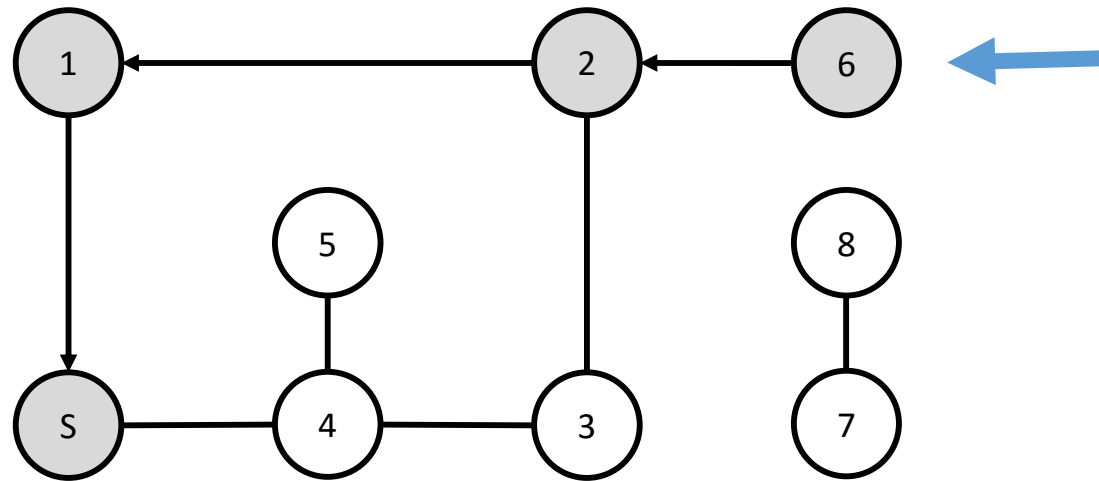
5.- Se marca en Gris el primer vecino, se indica su predecesor y se realiza una llamada recursiva a sus vecinos

Ejemplo de ejecución



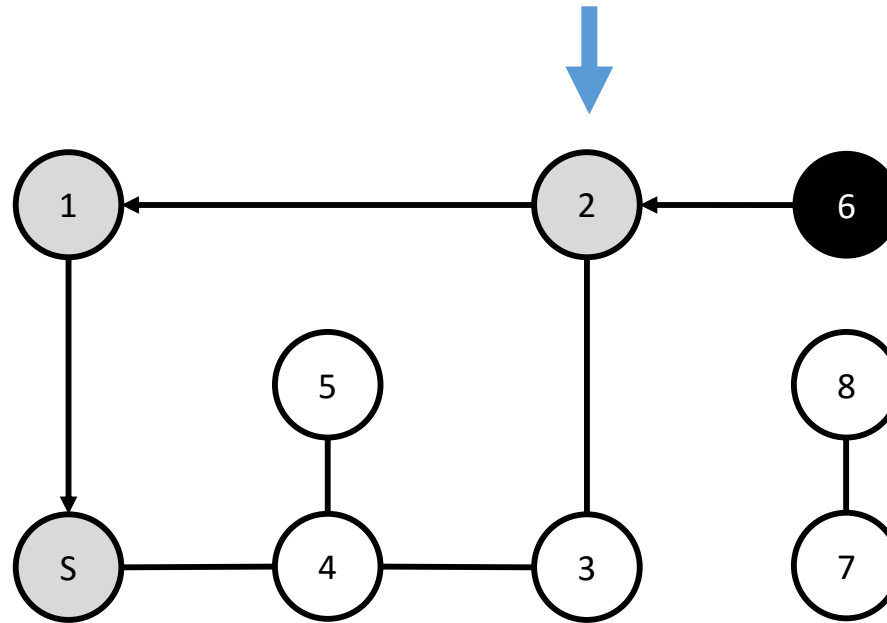
5.- Se marca en Gris el primer vecino y se realiza una llamada recursiva a sus vecinos blancos

Ejemplo de ejecución



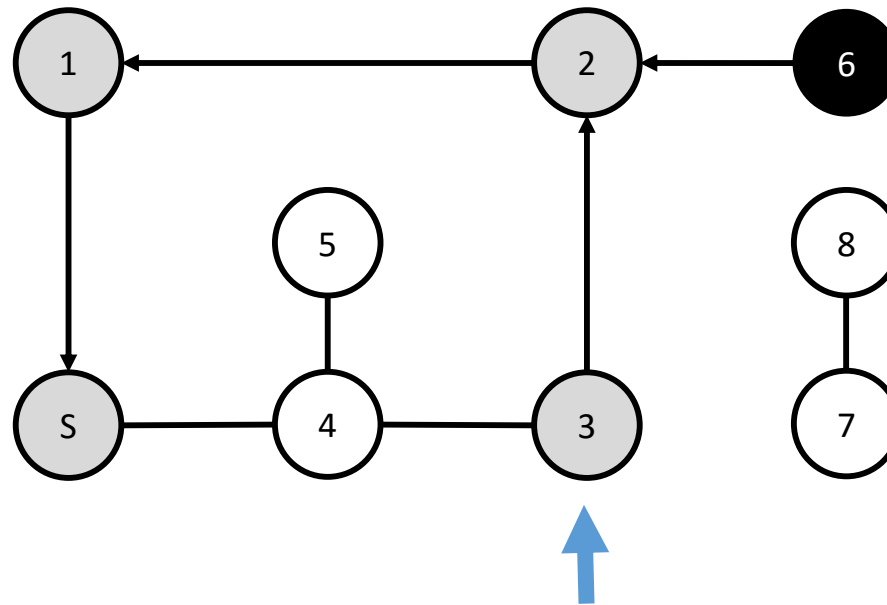
6.- Cuando llegamos a un nodo que no tiene vecinos marcados en blanco llegamos al caso base de la recursividad, por lo que marcamos el nodo en negro

Ejemplo de ejecución



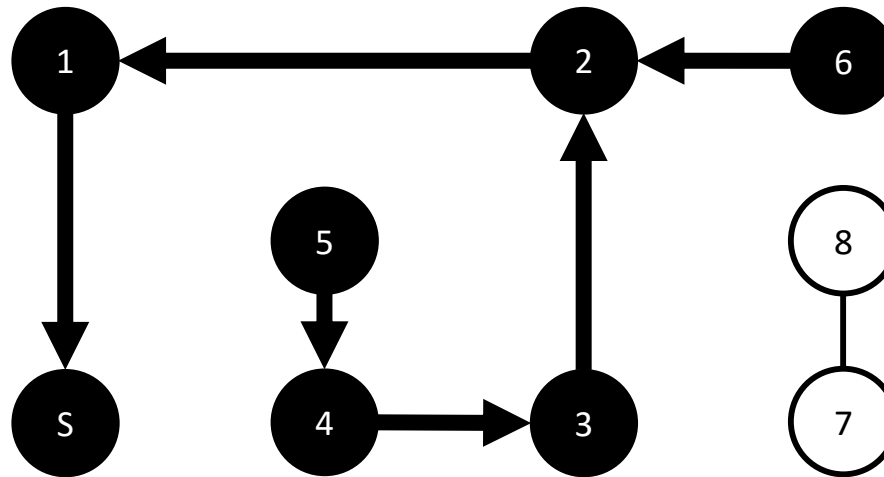
7.- Ahora regresamos al nodo anterior por la recursividad y seguimos visitando sus vecinos blancos restantes

Ejemplo de ejecución



7.- Ahora regresamos al nodo anterior por la recursividad y seguimos visitando sus vecinos blancos restantes

Ejemplo de ejecución



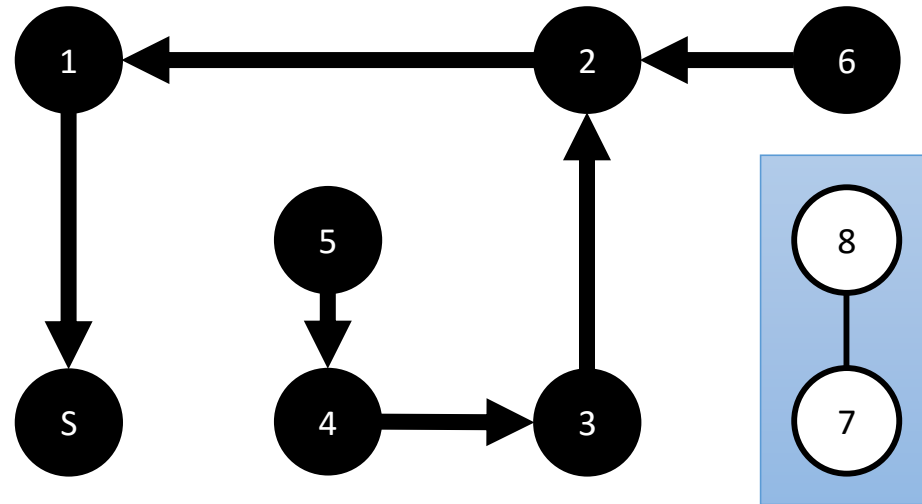
- 8.- De esa manera continúa hasta visitar todos los nodos y volver al nodo inicial "S" y seleccionar los predecesores
- 9.- Nótese que los **vecinos** que nunca fueron visitados (Grises o Negros) no forman parte del camino, ni tampoco los nodos que no están conectados a los vecinos lejanos

Ejemplo de ejecución

```
DFS (G, s)
  foreach v in V do
    v.color = White

  DFS_visit(s)

  foreach v in V do
    if (v.color = White) then
      DFS_visit(v)
    end
```



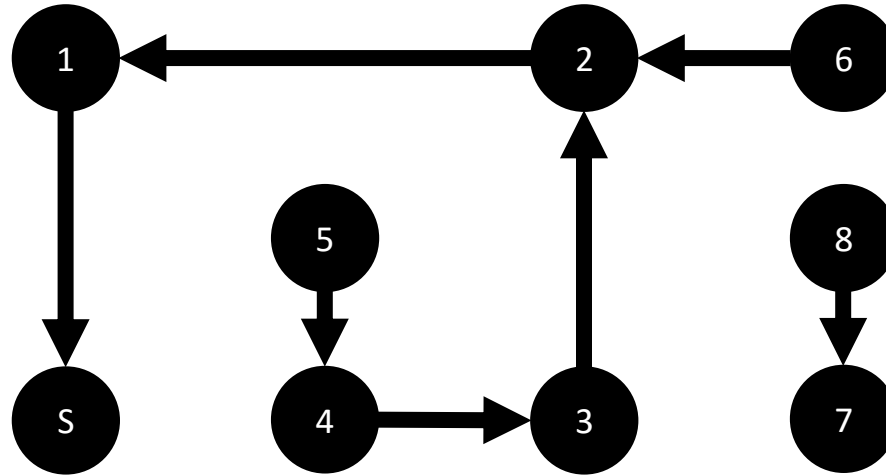
10.- El código marcado en rojo termina de recorrer los nodos que no fueron marcados, usando la misma llamada recursiva a los subgrafos

Ejemplo de ejecución

```
DFS (G, s)
  foreach v in V do
    v.color = White

  DFS_visit(s)

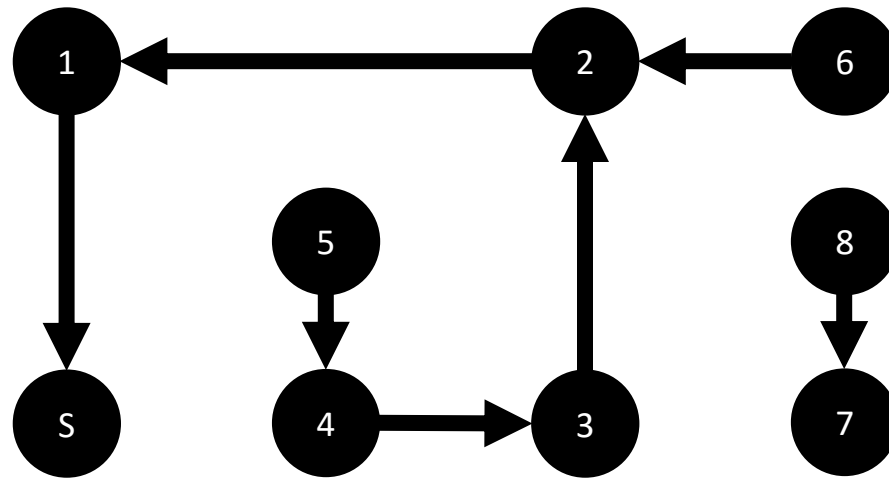
  foreach v in V do
    if(v.color = White) then
      DFS_visit(v)
    end
```



10.- El código marcado en rojo termina de recorrer los nodos que no fueron marcados, usando la misma llamada recursiva a los subgrafos

Búsqueda en profundidad

- El objetivo de este algoritmo es recorrer todos los nodos, no necesariamente encuentra la ruta más corta.



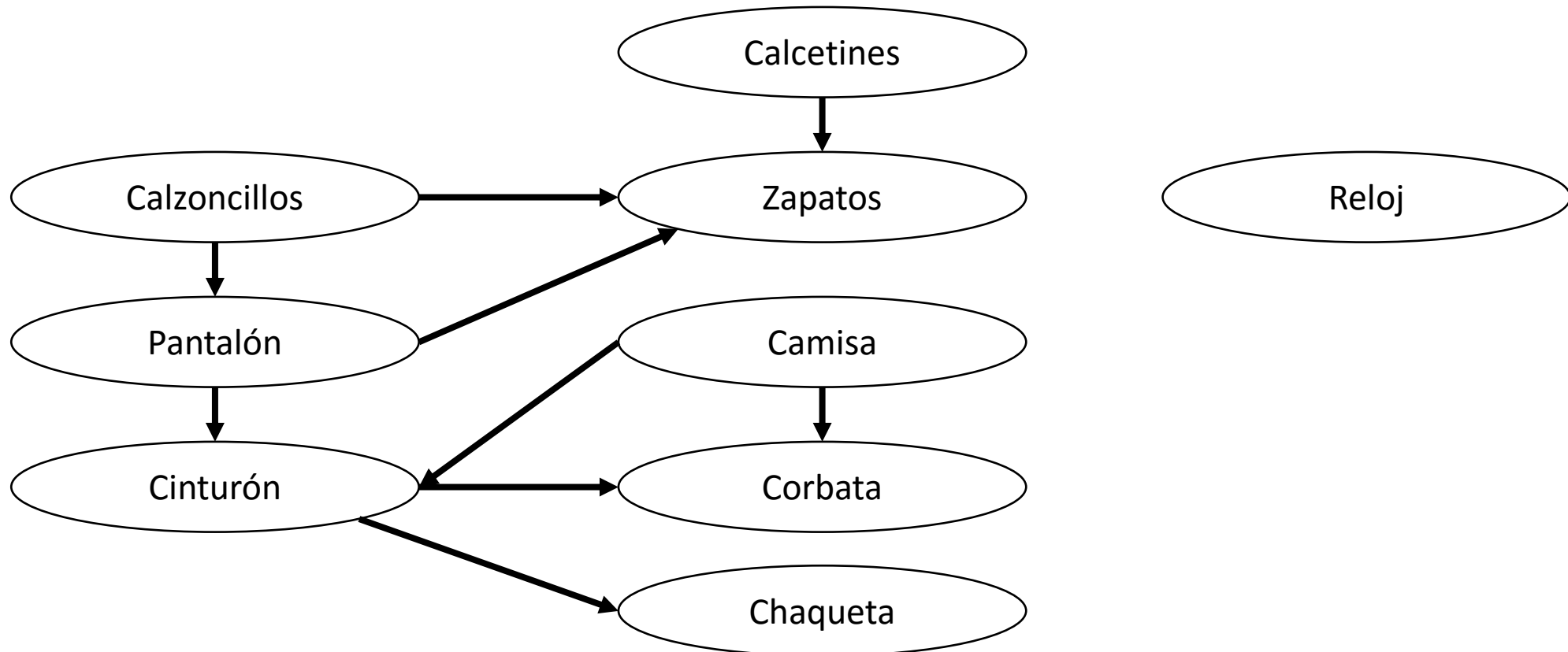
- Según las rutas resultantes, el nodo 5 llega a S a través de la ruta 5 -> 4 -> 3 -> 2 -> 1 -> S
- Y un ejemplo de ruta más corta habría sido 5 -> 4 -> S

DFS – Ejemplos de uso: Ordenación topológica

- Genera una lista en la cual todos los nodos permanecen unidos de la misma manera que el grafo original.
- Es utilizado para enlistar tareas en el orden en que podrían llevarse a cabo respetando dependencias.
- No es posible aplicarlo en grafos con ciclos.

Ejemplo: Vestirse

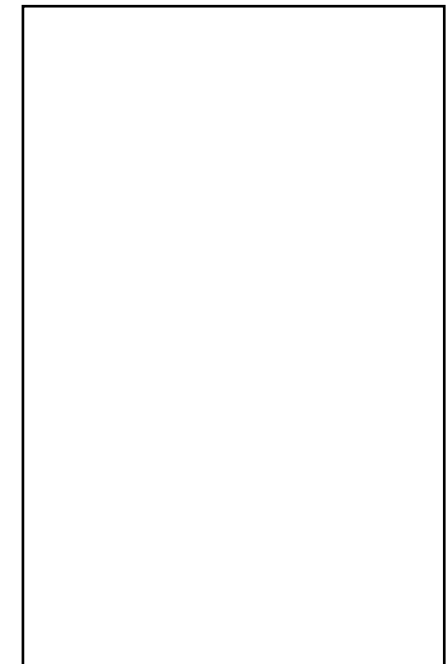
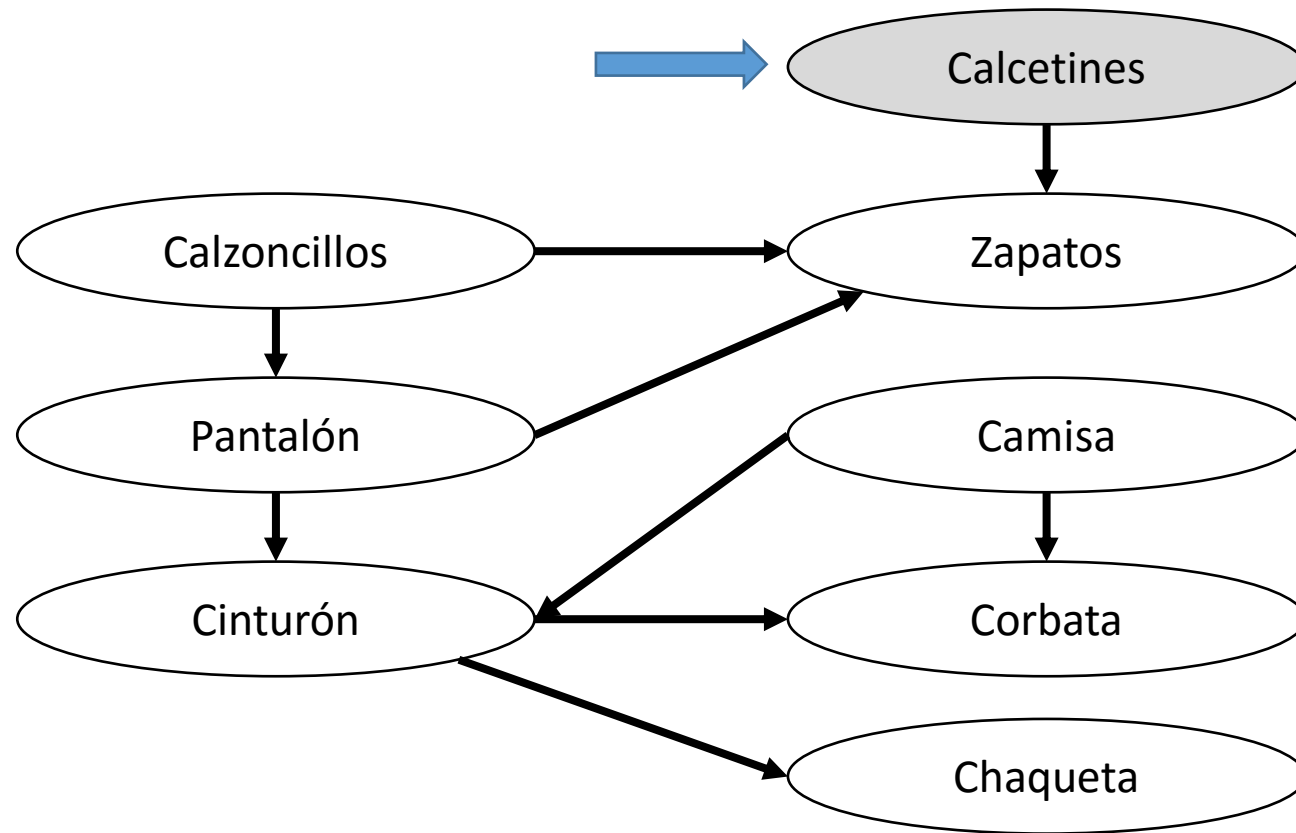
- El siguiente grafo muestra las dependencias de actividades para poder vestirse.



Resolución

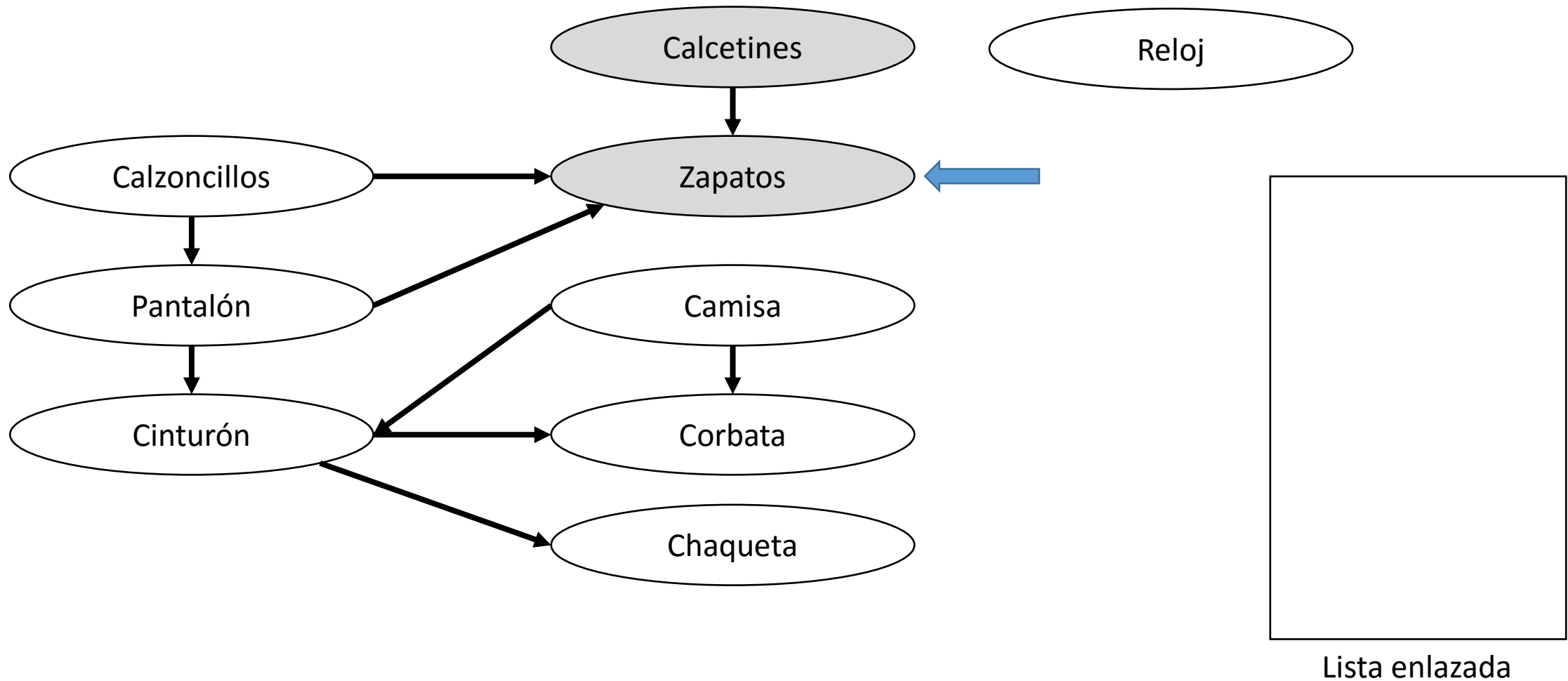
- Se buscan todos los nodos que no posean dependencias, en este caso:
 - Calcetines
 - Calzoncillos
 - Reloj
 - Camisa
- Luego se realiza una **búsqueda en profundidad** para cada uno de estos nodos como partida.
- Cuando se finaliza el recorrido de un nodo (Marca en Negro) este se almacena en una lista enlazada (LIFO)

Ejecución

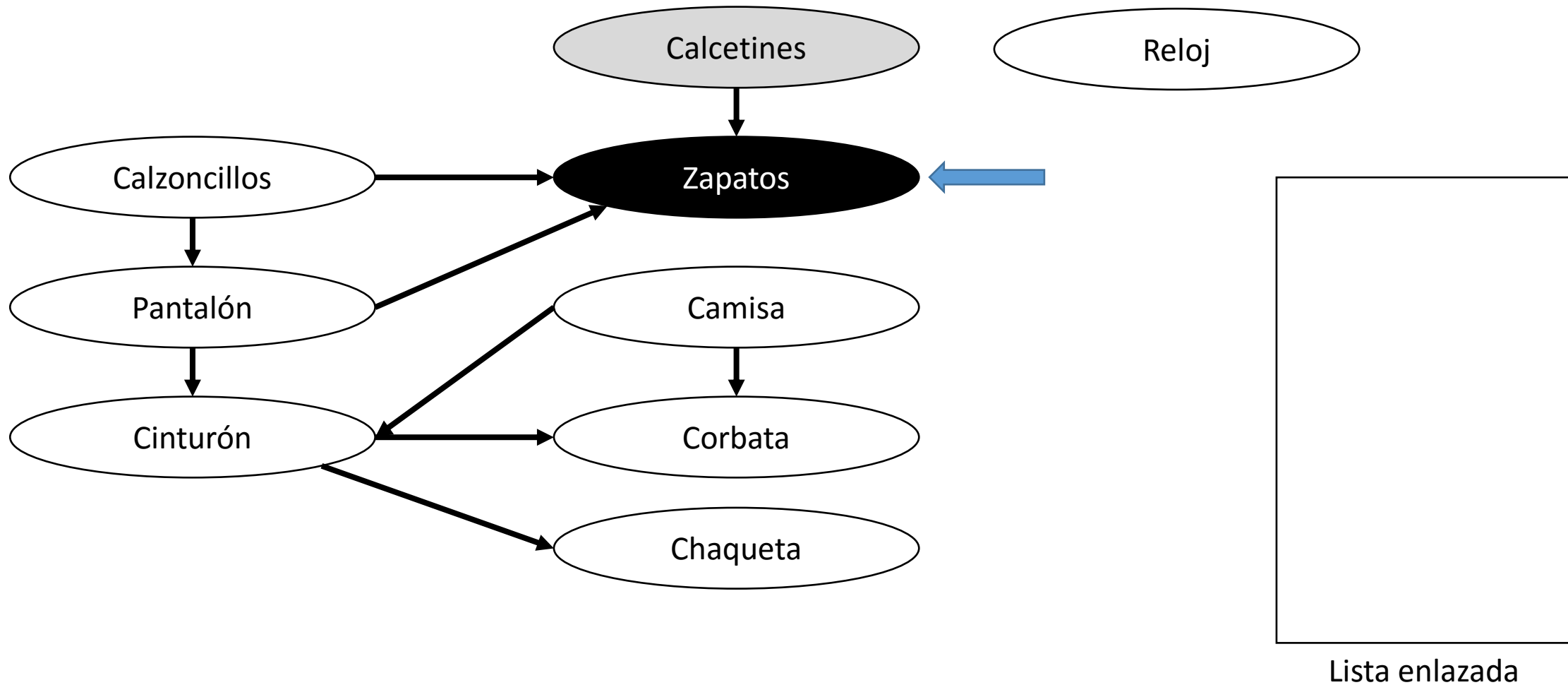


Lista enlazada

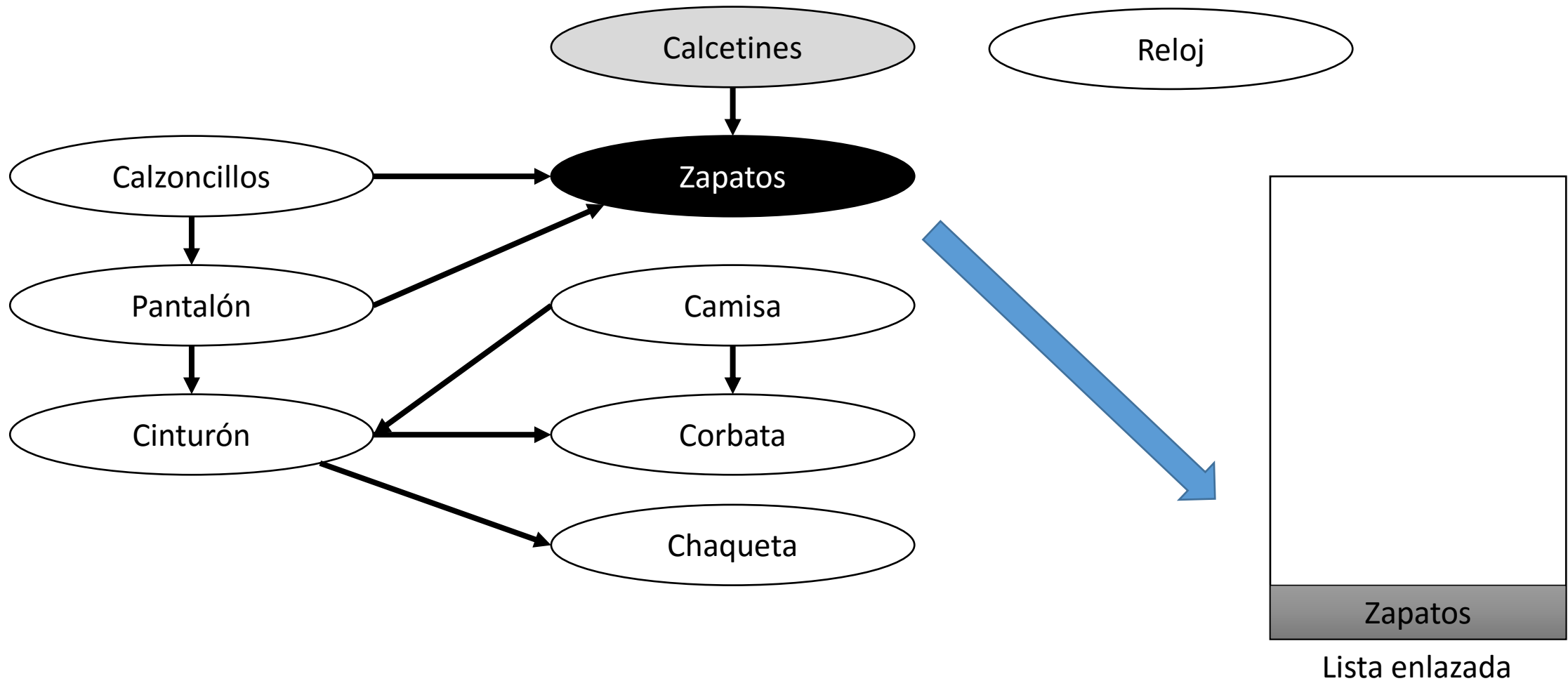
Ejecución



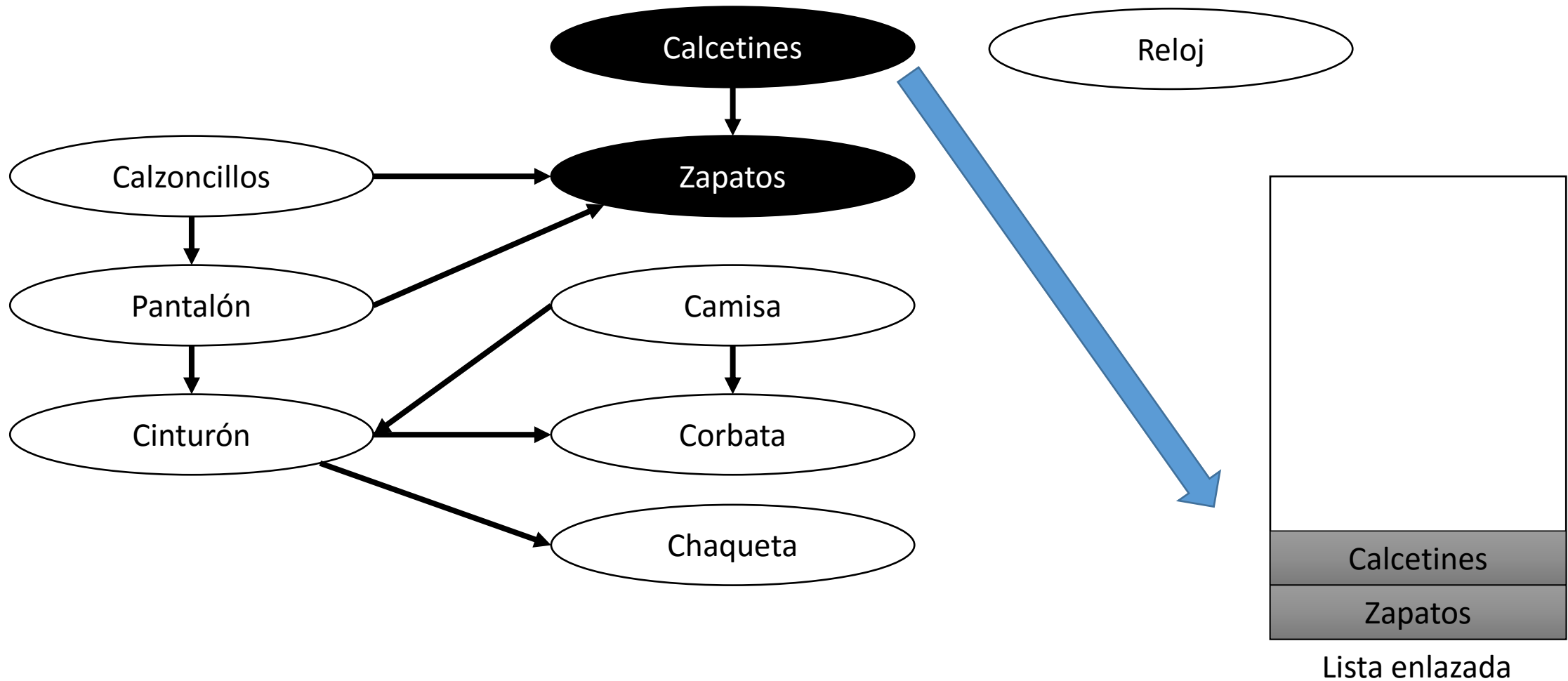
Ejecución



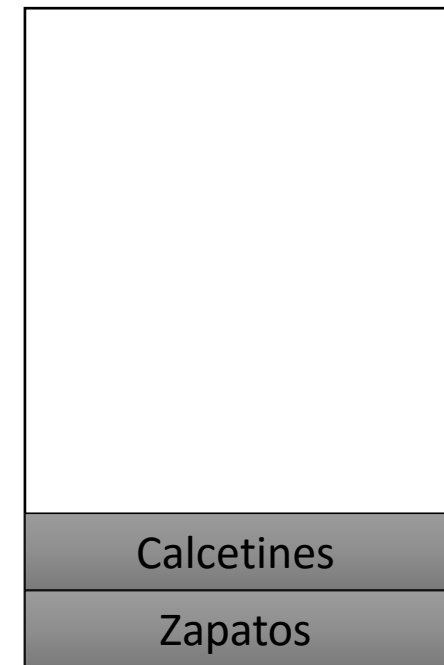
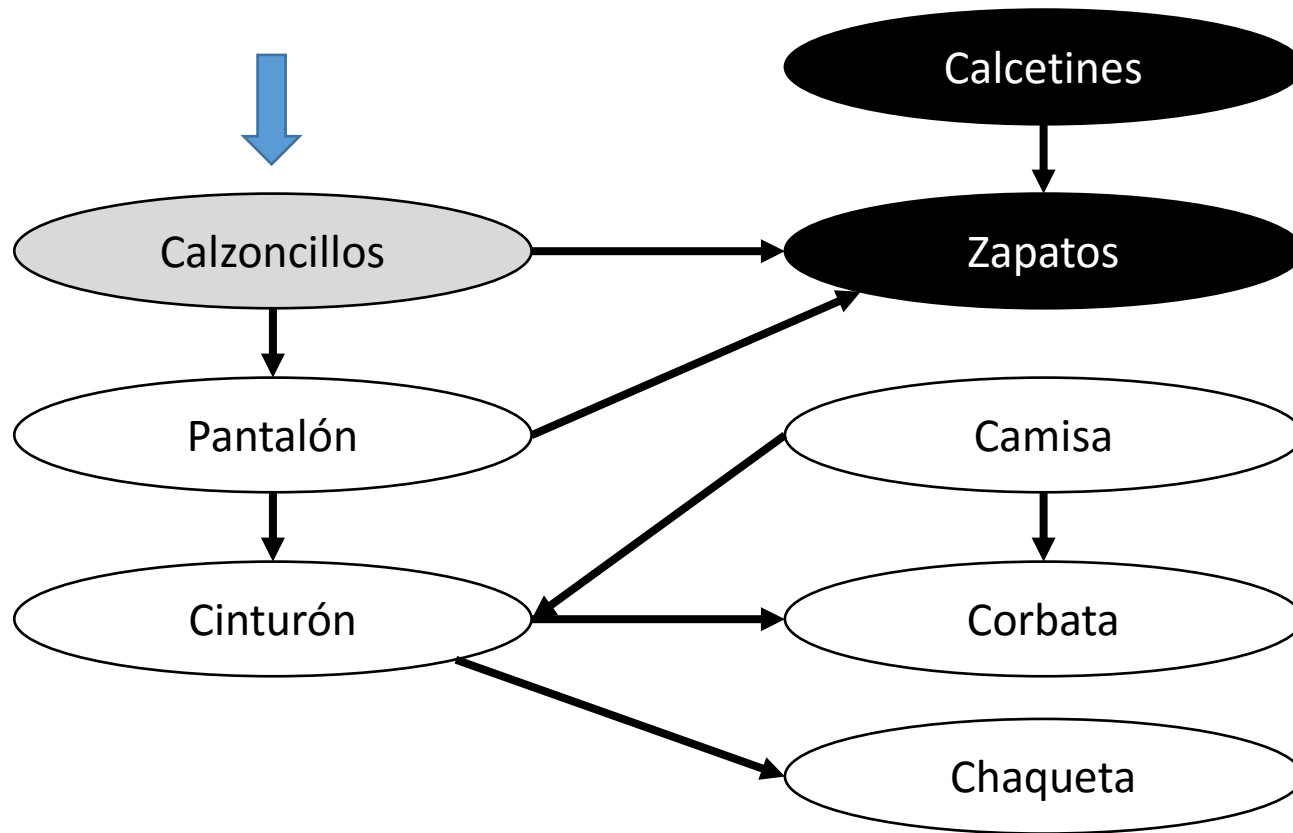
Ejecución



Ejecución

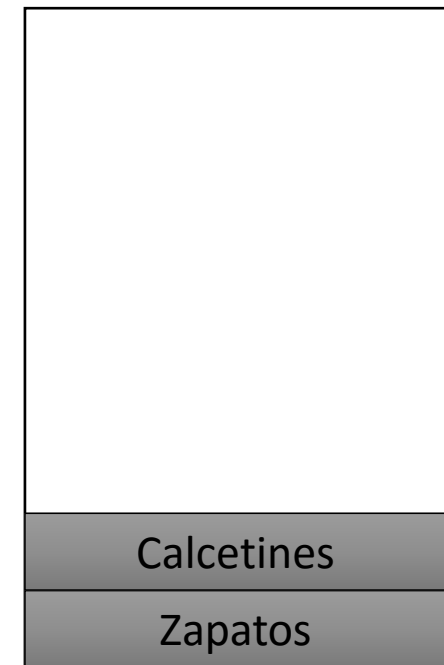
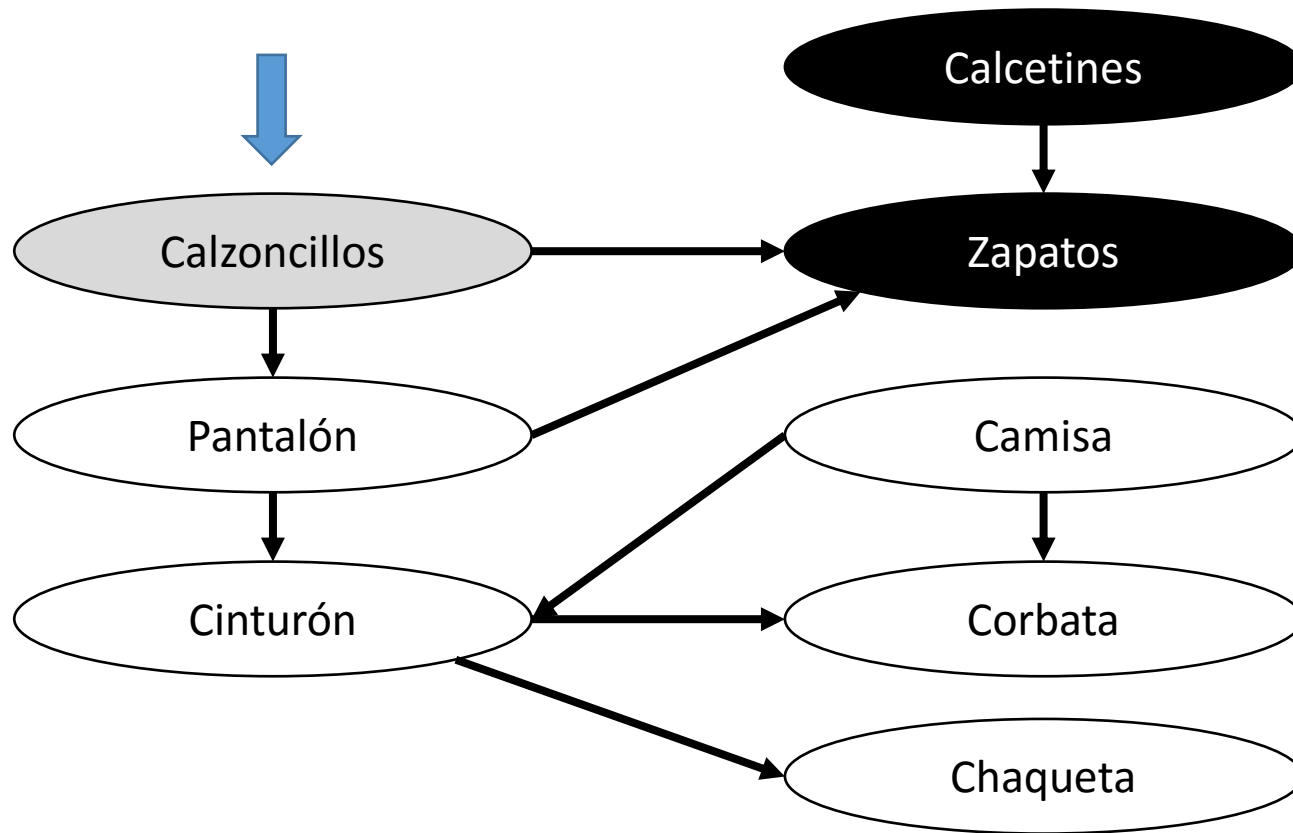


Ejecución



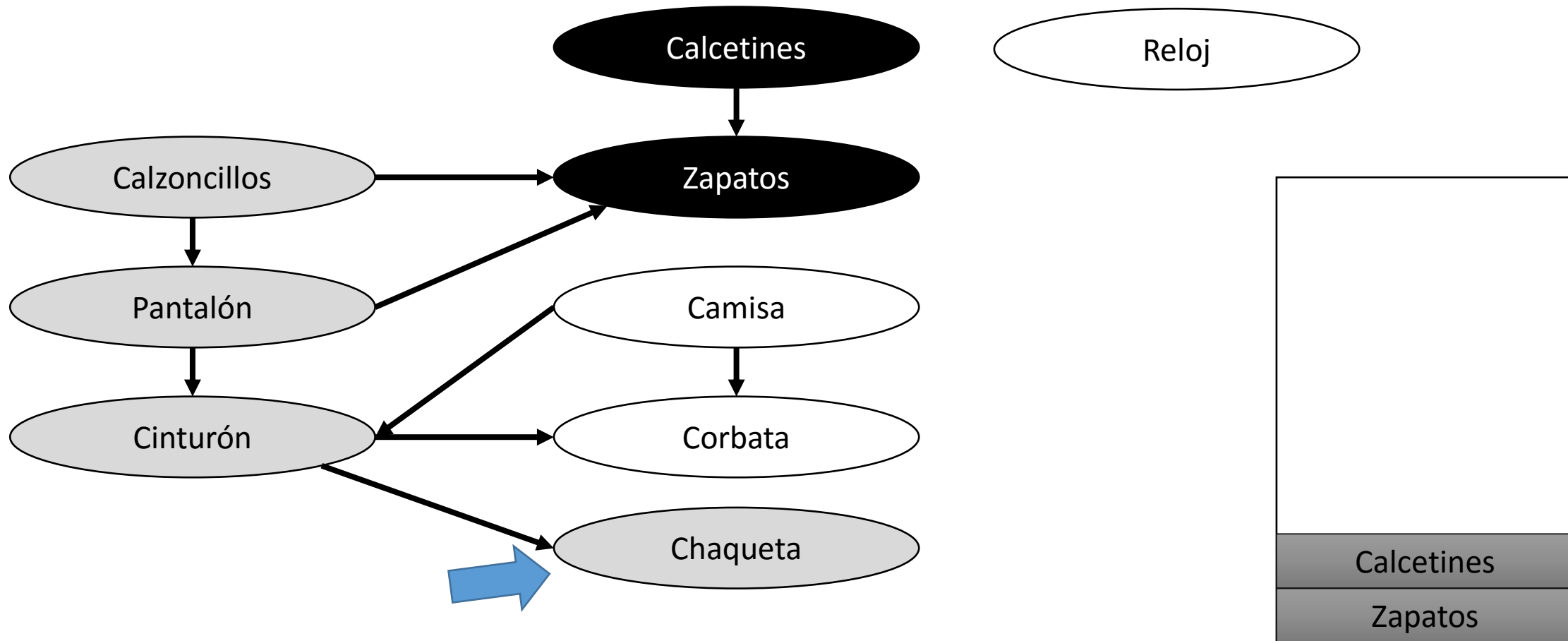
Lista enlazada

Ejecución



Lista enlazada

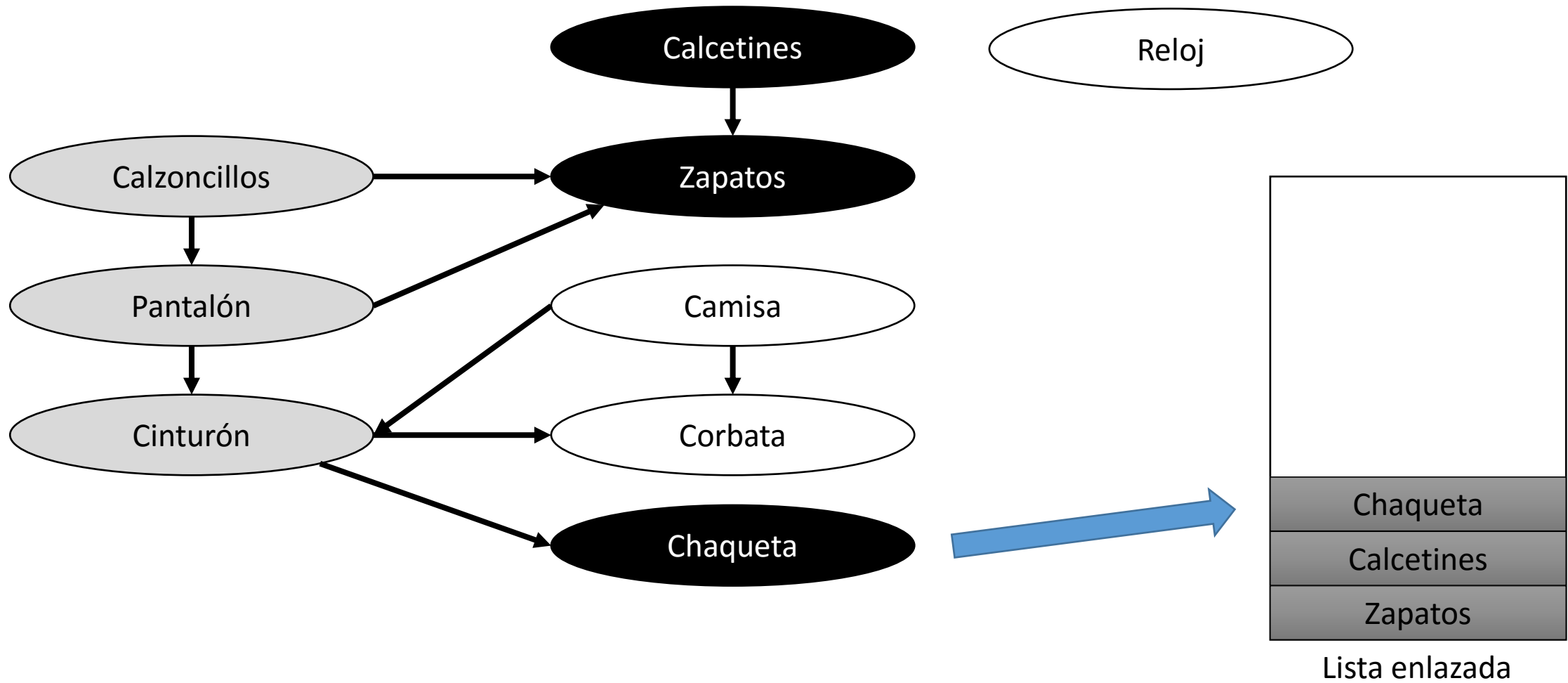
Ejecución



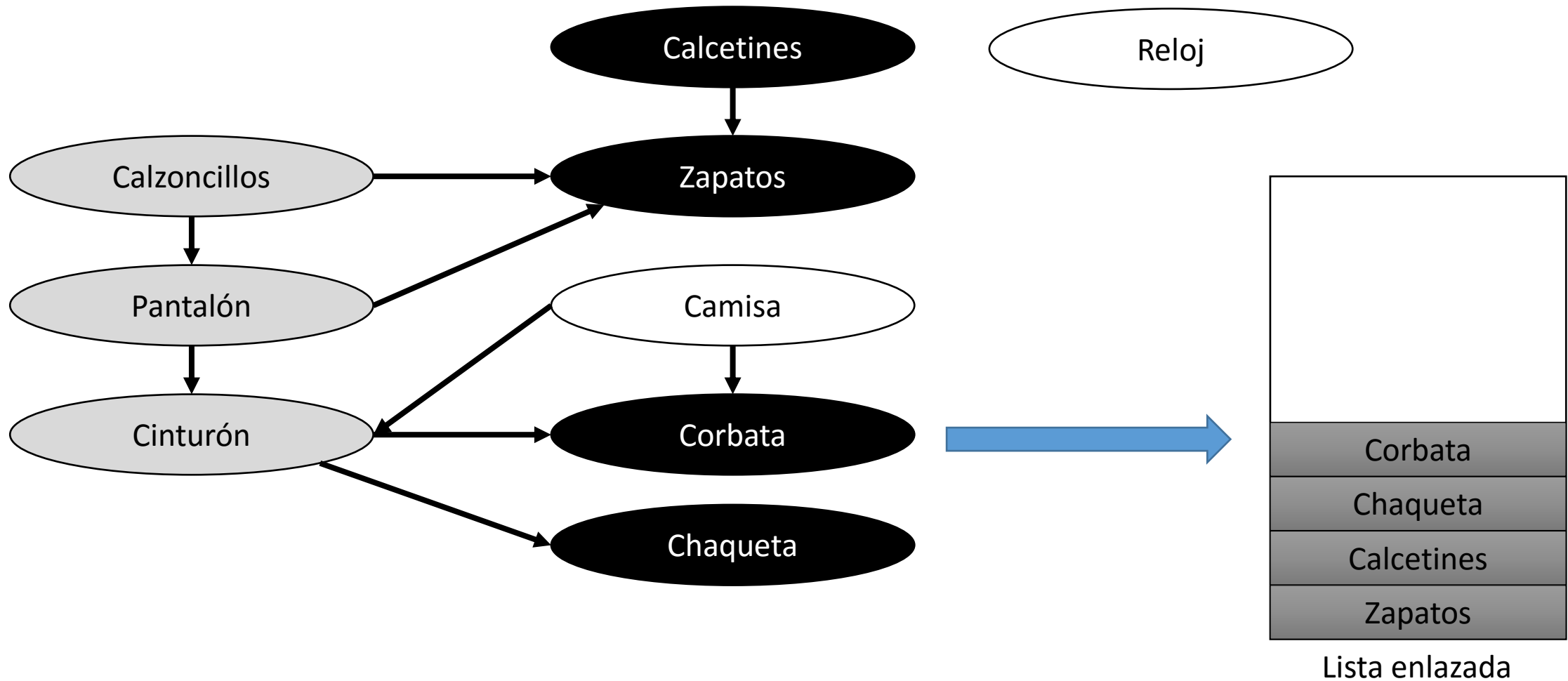
Tip: Zapatos no se vuelve a recorrer ya que está marcado en Negro

Lista enlazada

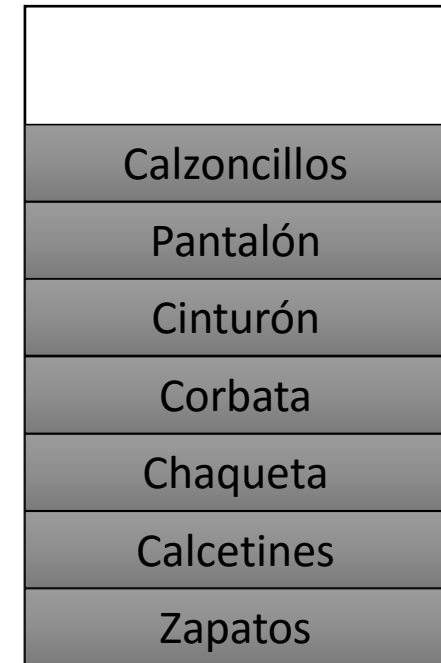
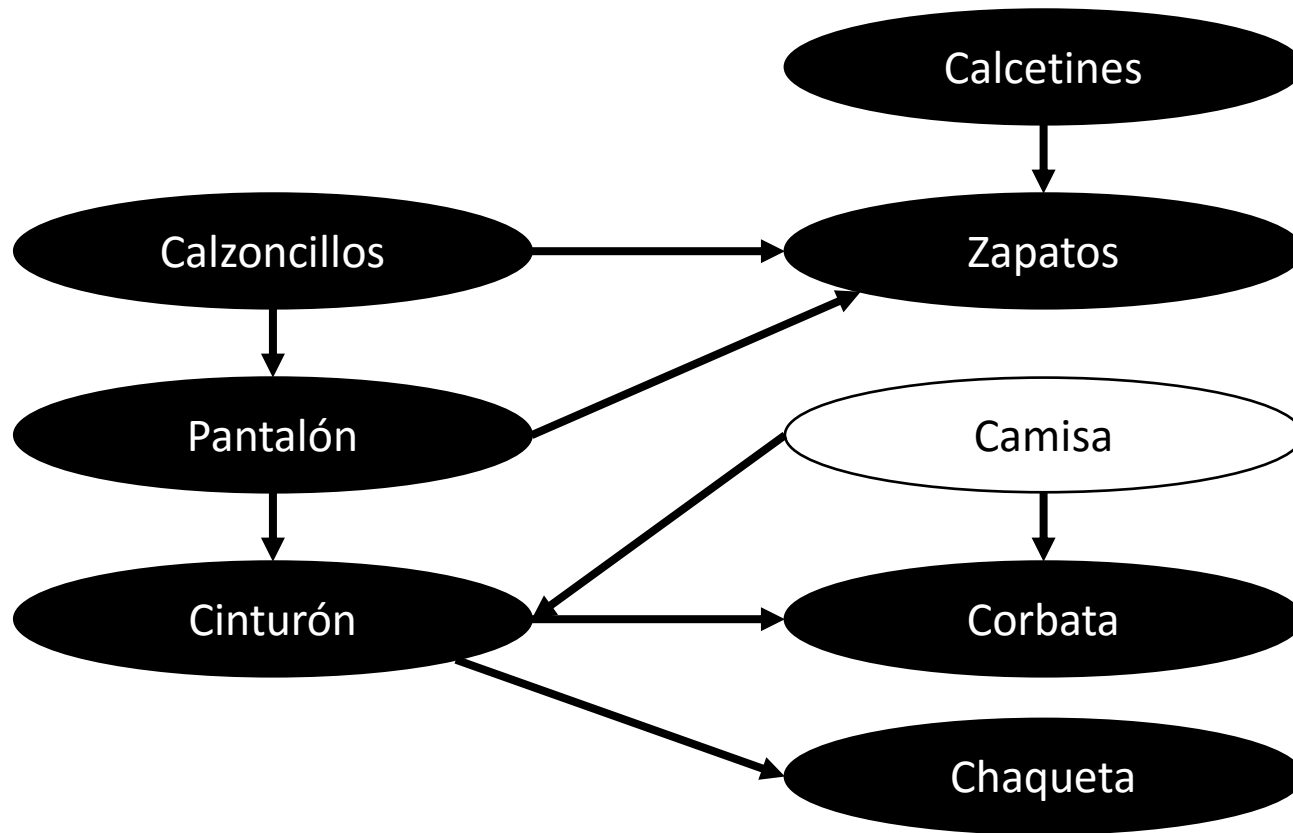
Ejecución



Ejecución

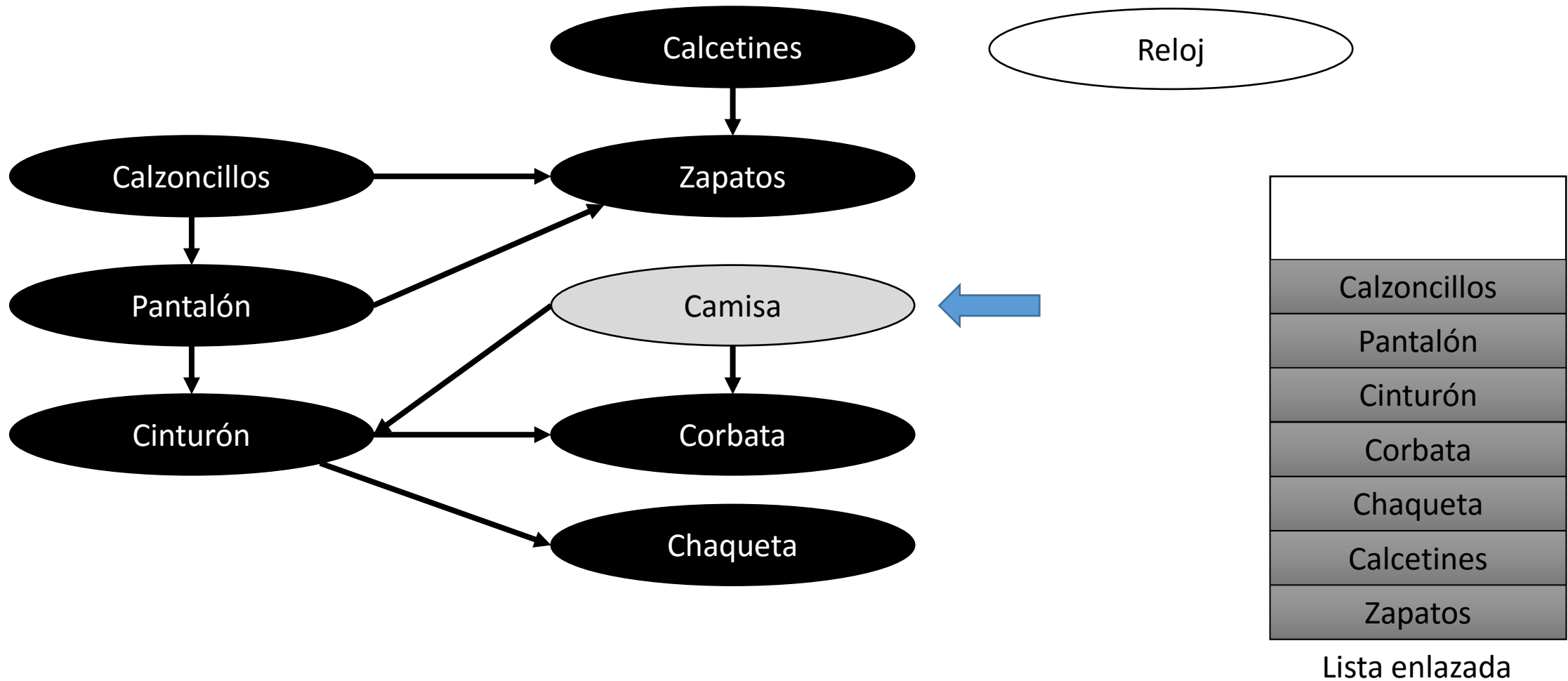


Ejecución

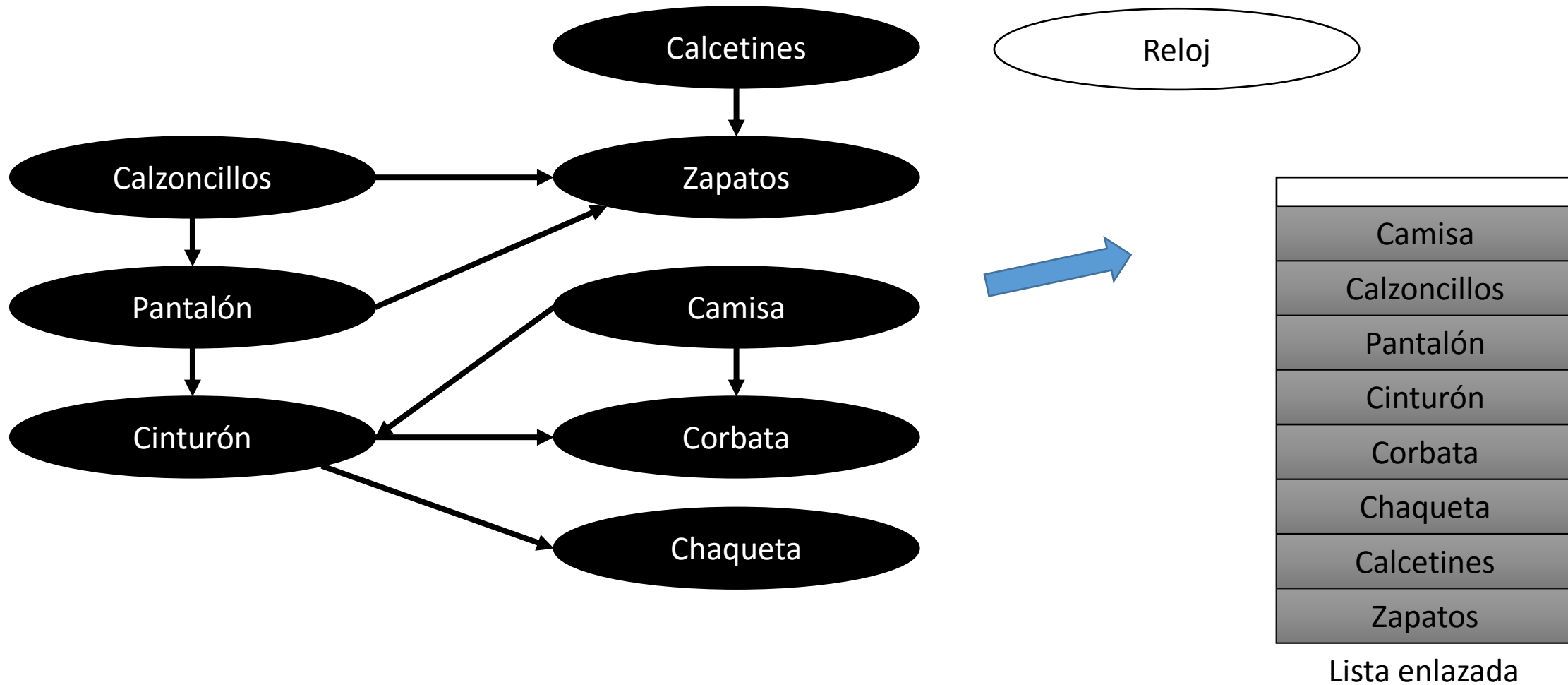


Lista enlazada

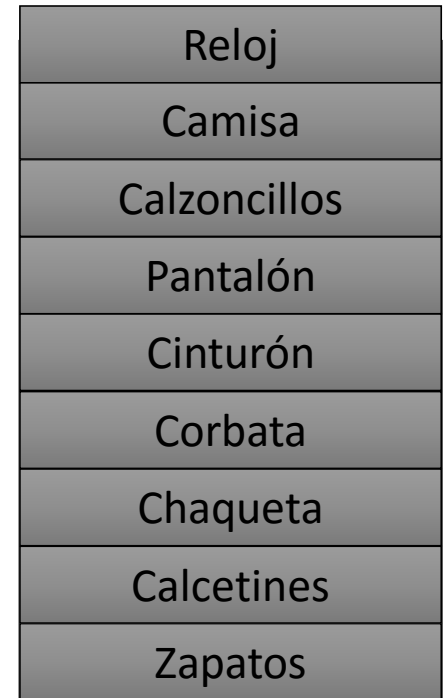
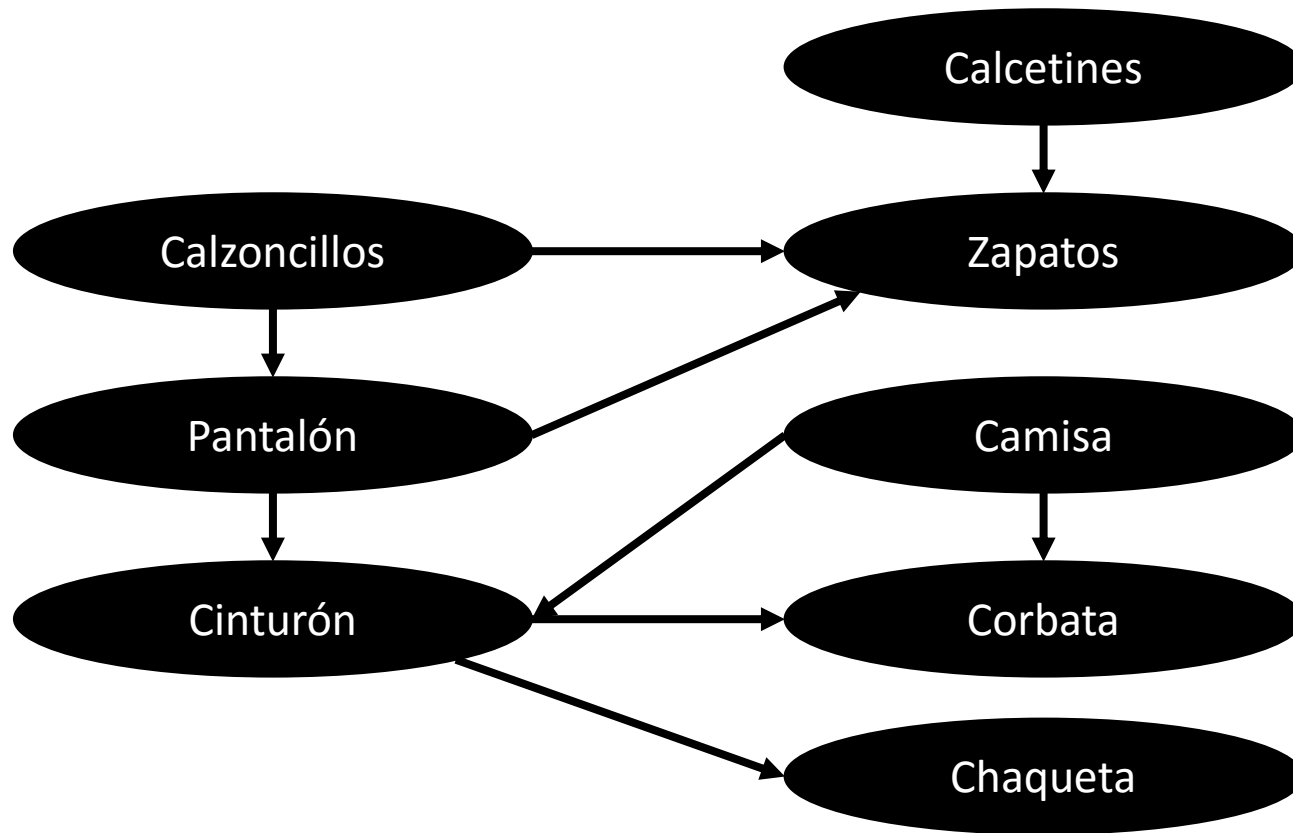
Ejecución



Ejecución

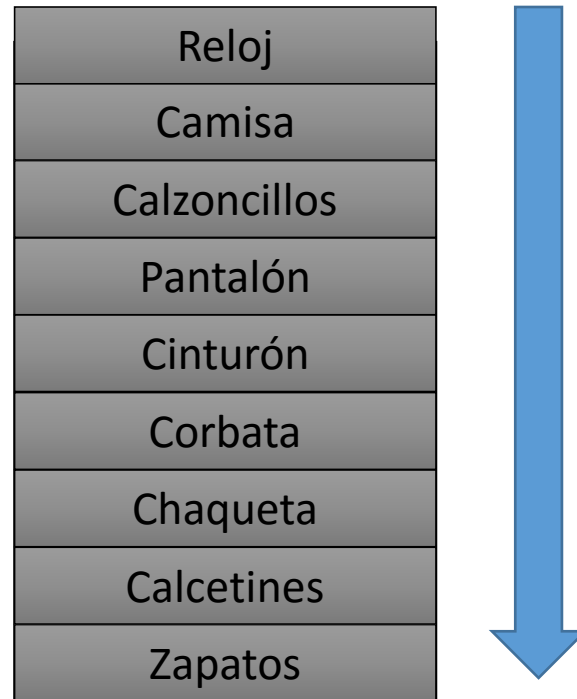


Ejecución



Lista enlazada

Lista de ordenamiento de tareas



Otros usos de DFS

- Identificar componentes fuertemente conectados
- Recorrido de laberintos
- Recorrido de árboles binarios

Búsqueda en anchura

- Abreviado BFS, del inglés Breadth-First Search
- Sistemáticamente visita todos los vecinos de un nodo antes de avanzar al siguiente nivel.
- Se realiza esto hasta que no existen más vértices alcanzables por S
- Este algoritmo no visita vértices en el grafo que no son alcanzables por S

Algoritmo en Pseudocódigo

BFS (G, s)

foreach v in V **do**

 v.pred = NULL

 v.dist = ∞

 v.color = White

s.color = Gray

s.dist = 0

Q = empty Queue

enqueue(Q, s)

while Q is not empty **do**

 u = head(Q)

foreach neighbor v of u **do**

if v.color = White **then**

 v.dist = u.dist + 1

 v.pred = u

 v.color = Gray

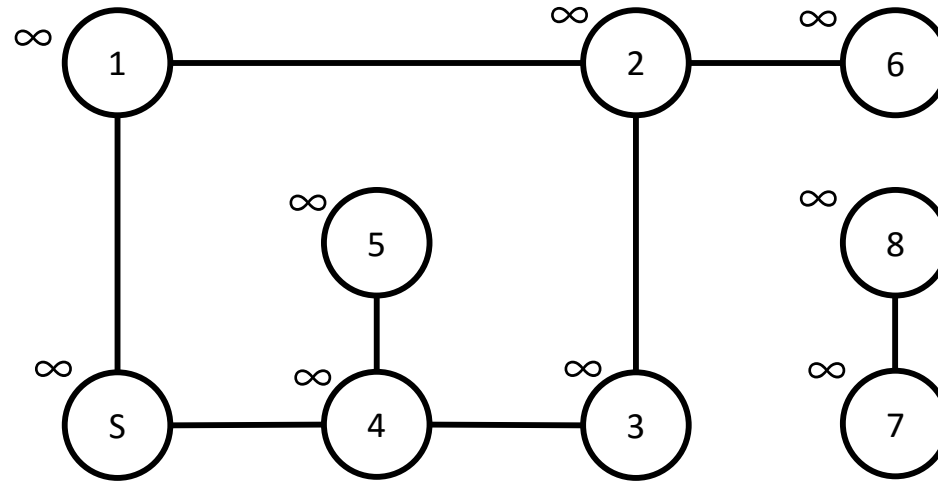
 enqueue(Q, v)

 dequeue(Q)

 u.color = Black

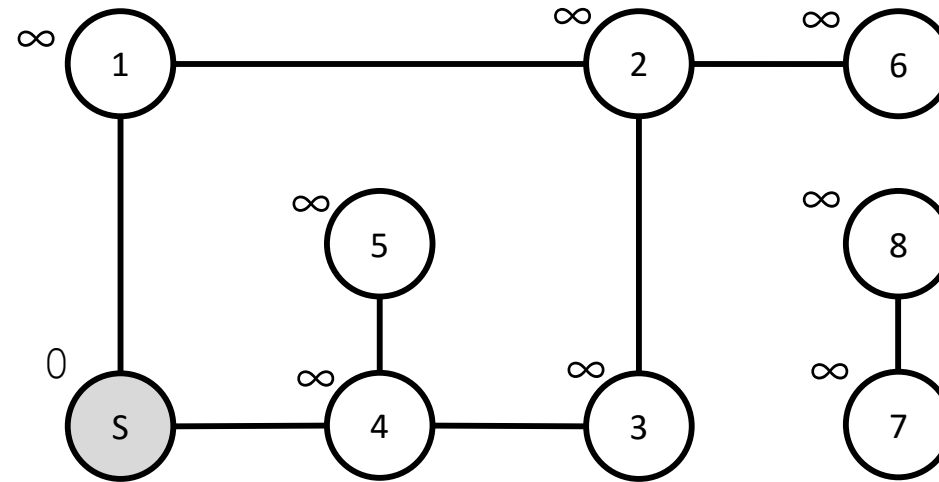
end

Ejemplo de ejecución



- 1.- Se marcan todos los nodos con el color blanco y su distancia contra el inicio en infinito

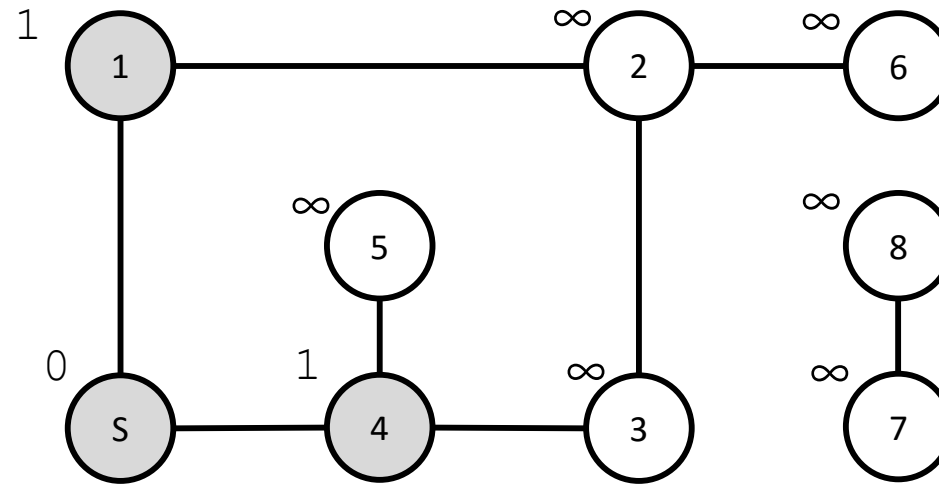
Ejemplo de ejecución



COLA:
S

2.- Se marca en gris el nodo de inicio, se indica que la distancia a él mismo es 0 y se ingresa en una cola

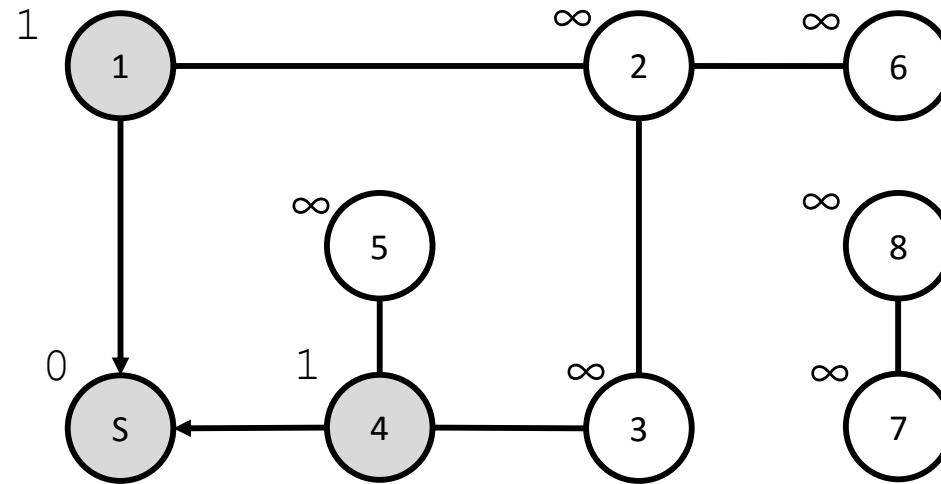
Ejemplo de ejecución



COLA:
S

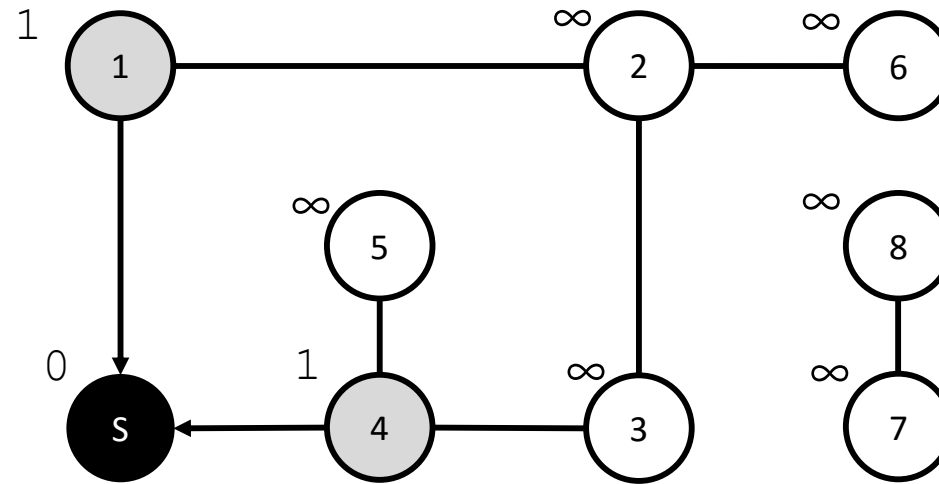
- 3.- Se empieza por el primer elemento de la cola a visitar (en este caso S es el único elemento en la cola), se recorren cada vecino blanco y se indica que la distancia al origen "S" es uno más que la distancia actual del nodo padre

Ejemplo de ejecución

COLA:
S - 1 - 4

- 4.- Se encola cada vecino en el orden en que fueron visitados
y se indica el nodo precedente igual al nodo padre actual (en este caso S)

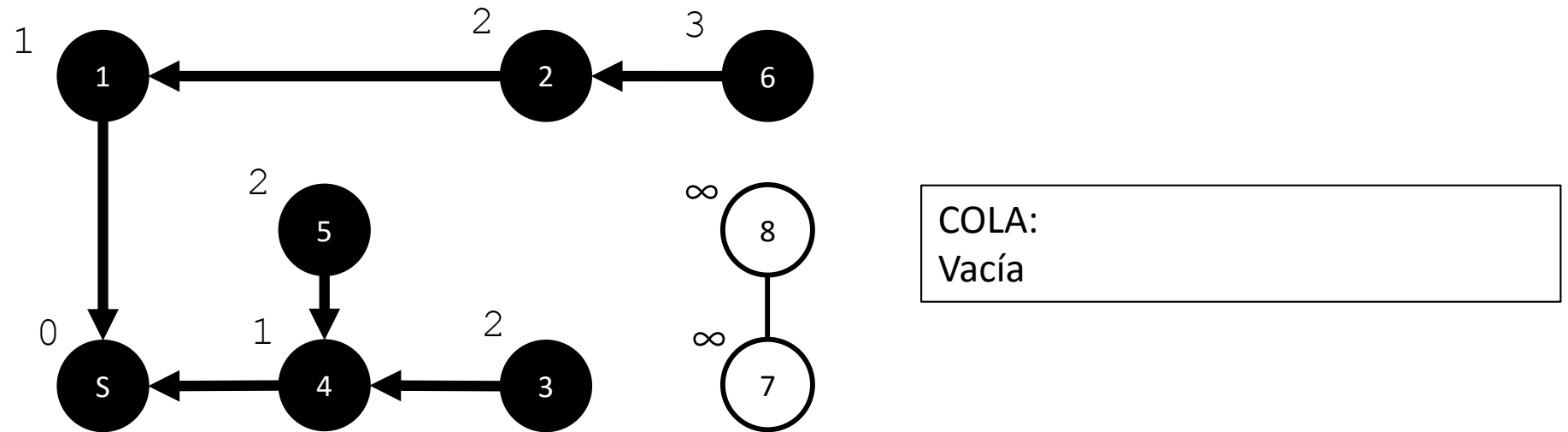
Ejemplo de ejecución



COLA:
1 – 4

- 4.- El nodo actual (En este caso S) se marca en negro y se quita de la cola
- 5.- Se procede con el siguiente nodo en la cola

Ejemplo de ejecución



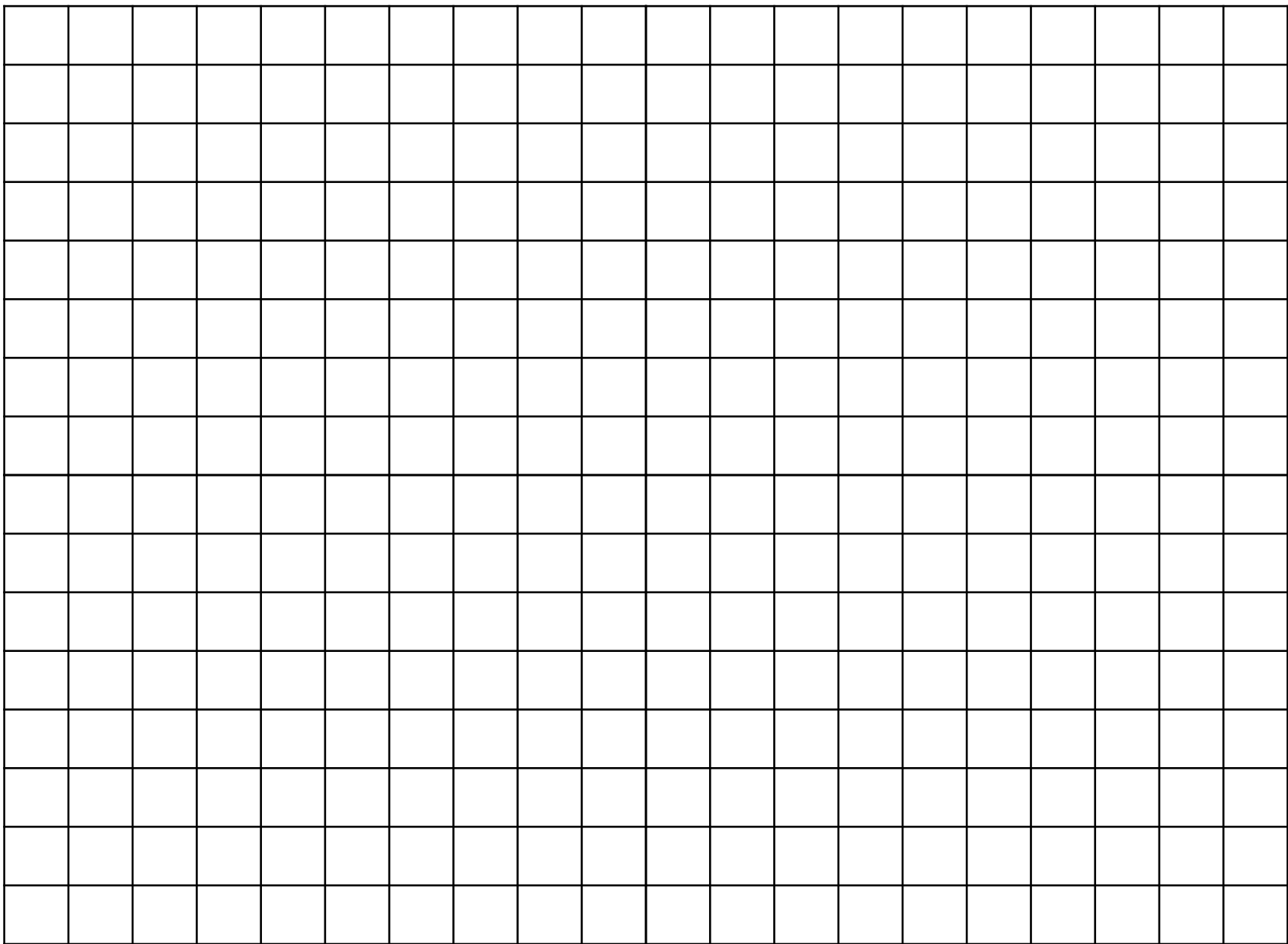
- 6.- Se siguen encolando los nodos y recorriendo sus vecinos hasta que la cola está vacía y todos los nodos alcanzables por S han sido marcados en negro
- 7.- Los vecinos no recorridos son quitados de las rutas posibles

Búsqueda en anchura

- Este algoritmo entrega las rutas más cortas en grafos no ponderados (asegura la cantidad mínima de saltos, paso por aristas).
- En grafos ponderados ignora los costos por lo que posiblemente la ruta no sea la más corta.
- Los nodos no alcanzables por S quedan marcados en blanco y con una **distancia infinita**.

Aplicaciones

- Obtención de rutas más cortas en grafos no-ponderados
- Recorrido de árboles por niveles
- Experimento de Milgram (Experimento del mundo pequeño)




CHECK - OBJETIVOS DE LA SESIÓN

- Conocer los conceptos: camino, grafo completo, ciclo y recorrido
- Conocer el concepto de búsqueda en anchura como forma de resolución de problemas utilizando grafos
- Conocer el concepto de búsqueda en profundidad como forma de resolución de problemas utilizando grafos

CHECK





Diseño de Algoritmos

Sesión 11

Profesores:

Tomás Lara Valdovinos – t.lara@uandresbello.edu

Jessica Meza-Jaque – je.meza@uandresbello.edu