Tema 4: Teoría de Grafos

Mariam Cobalea

Universidad de Málaga
Dpto. de Matemática Aplicada

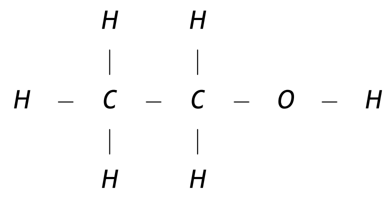
Tema 4: Teoría de Grafos

Tema 4: Teoría de Grafos

4.7 Árboles

- Los árboles son grafos conexos sin ciclos.
- Cayley los usó por primera vez en 1857 para representar ciertos tipos de componentes químicos.

Ejemplo



4.7 Árboles

Introducción

En Ciencias de la Computación los árboles nos sirven para:

- **almacenar** y **recuperar** la información.
- construir **algoritmos** eficientes para **localizar** items en una lista.
- construir códigos eficientes para **almacenar / transmitir** datos.
- modelar procedimientos que se realizan usando una secuencia de **decisiones**. Por tanto, son útiles en los algoritmos de clasificación.

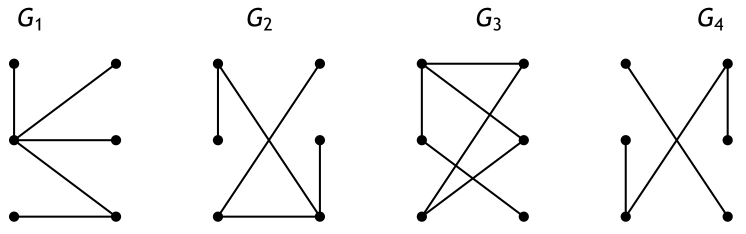
4.7 Árboles

Conceptos fundamentales

Definición

*Un **árbol** es un grafo conexo sin ciclos.*

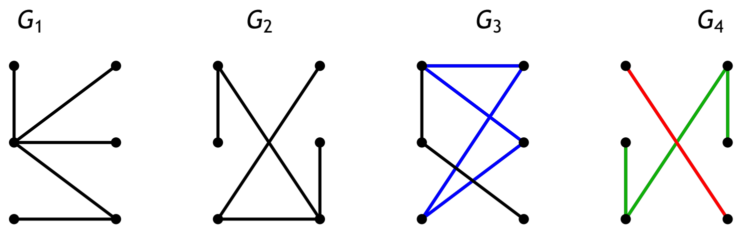
Ejemplo Indica qué grafos de la siguiente figura son árboles.



4.7 Árboles

Conceptos fundamentales

Ejemplo Indica qué grafos de la siguiente figura son árboles.



- Los grafos G_1 y G_2 son árboles, pero G_3 y G_4 no lo son.
- El grafo G_3 tiene un ciclo y el grafo G_4 no es conexo.

4.7 Árboles

Conceptos fundamentales

Definición

*Se llama **bosque** a un grafo acíclico cuyas componentes conexas son árboles.*

Ejemplo El grafo G_4 es un bosque.

✓ Puesto que un árbol es un grafo acíclico, **no** puede contener bucles, **ni** aristas múltiples.

✓ Por tanto, **un árbol es**, necesariamente, **un grafo simple**.

4.7 Árboles

Propiedades de los Árboles

Teorema

*Un grafo no dirigido $G = (V, E)$ es un árbol si, y sólo si, hay un **único** camino entre cada par de vértices.*

Teorema

Sea $G = (V, E)$ un grafo simple. Son equivalentes:

1. *G es un **árbol**.*
2. *G es **conexo** y $|E| = |V| - 1$.*
3. *G **no tiene ciclos** y $|E| = |V| - 1$.*
4. *En G hay **exactamente** un camino entre cada par de vértices.*
5. *G es **conexo**, pero si eliminamos una arista cualquiera, el grafo resultante **no** es conexo.*

4.7 Árboles
Árboles generadores

- Los árboles tienen propiedades de minimalidad, en el sentido de que **son los grafos conexos con el menor número posible de aristas**.
- Modelan, así, redes de comunicaciones con un **mínimo** número de conexiones entre nodos.

Definición

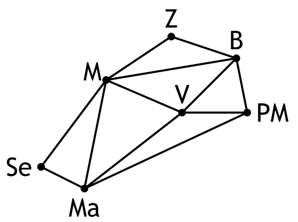
Un **subgrafo generador** *de un grafo simple* $G = (V, E)$ *es cualquier subgrafo de* G *que incluye todos los vértices de* G.

Definición

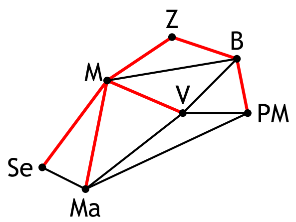
Un **árbol generador** (spanning tree) *de un grafo simple* $G = (V, E)$ *es un subgrafo generador de* G *que es un árbol.*

4.7 Árboles
Árboles generadores

Ejemplo Un árbol generador del grafo G es el árbol T



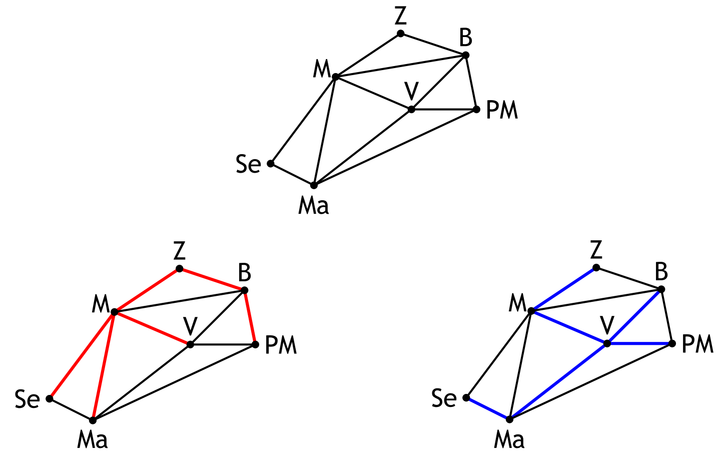
Grafo G



Árbol generador T

4.7 Árboles
Árboles generadores

✓ Un grafo simple puede tener más de un árbol generador



4.7 Árboles
Árboles generadores

Si un grafo simple tiene un árbol generador, entonces es conexo claramente.
El recíproco también es cierto. Así tenemos

Teorema

Un grafo simple G *es conexo si y sólo si tiene un árbol generador* T.

Demostración: $(\Longleftarrow)$ Evidente, por ser T un árbol.
$(\Longrightarrow)$ Sea G un grafo conexo. Si G no es un árbol, contiene un ciclo.
Al quitar uno de las aristas de ese ciclo, nos queda un subgrafo con una arista menos pero que todavía contiene todos los vértices de G y es conexo.
Si este subgrafo no es un árbol, tendrá un ciclo.Se procede igual que antes y se repite el proceso anterior hasta que no queden ciclos.
El resultado es un árbol, ya que el grafo permanece conectado mientras las aristas se eliminan. Este árbol es un árbol generador.

4.7 Árboles
Árboles con raíz

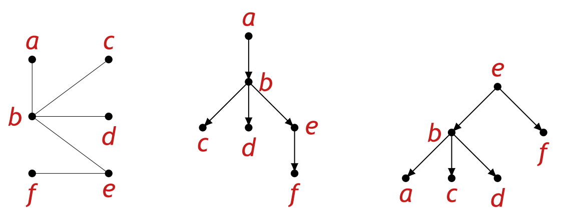
- En muchas aplicaciones se necesita destacar un vértice particular.
- A este vértice se le llama ***raíz***.
- Una vez especificada la raíz, podemos asignar una dirección a cada arista del árbol de la siguiente manera: puesto que hay un único camino entre la raíz y cada uno de los restantes vértices, la dirección de cada arista es la que se aleja de la raíz.
- De esta manera, un árbol junto a su raíz produce un grafo dirigido llamado ***árbol con raíz***.

4.7 Árboles
Árboles con raíz

Definición

*Un **árbol con raíz** es un árbol en el que uno de sus vértices se ha designado como la raíz y todas las aristas están orientadas de modo que se alejan de la raíz.*

Ejemplo Del árbol T podemos obtener distintos árboles con raíz:



Árbol T — Árbol con raíz a — Árbol con raíz e

4.7 Árboles
Árboles con raíz

- ✓ Para dibujar un árbol con raíz se dibuja la raíz en la parte superior del grafo.
- ✓ Las flechas indicando las direcciones de los arcos se pueden omitir, ya que la elección de la raíz determina las orientaciones de los arcos.

4.7 Árboles
Árboles con raíz

Definición

Sea T un árbol con raíz y sea v un vértice distinto a la raíz.

- *El **padre** de v es el único vértice u tal que hay una arista dirigida desde u hasta v.*
- *Cuando u es el padre de v, se dice que v es el **hijo** de u.*
- *Los vértices con el mismo padre se llaman **hermanos**.*
- *Los **antecesores** de un vértice v son todos los vértices que aparecen en el camino desde la raíz hasta v.*
- *Los **descendientes** de un vértice v son aquellos vértices para los que v es un antecesor.*

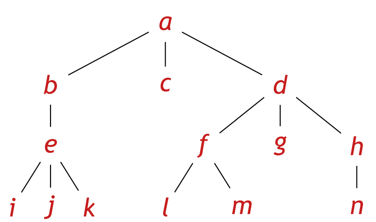
Definición

Sea T un árbol con raíz y sea v un vértice distinto a la raíz.

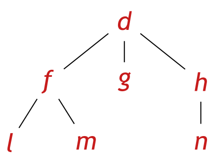
- *Una **hoja** es un vértice que no tiene hijos.*
- *Los **vértices internos** son los vértices que tienen hijos.*
- *Si x es un vértice de un árbol con raíz T, el **subárbol** con raíz en x es el subgrafo de T que contiene al vértice v, a todos sus descendientes y a todas las aristas incidentes en dichos descendientes. Se denota T_x*

☛ La raíz se considera vértice interno, a menos que sea el único vértice del grafo. En ese caso, se considerará una hoja.

Ejemplos



Árbol T con raíz en a



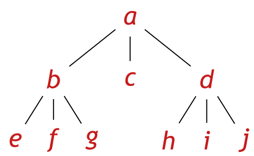
Subárbol con raíz en d

- Los árboles con raíz tales que todos sus vértices internos tienen el mismo número de hijos se utilizan en múltiples aplicaciones.
- Estos árboles se usan para estudiar problemas relativos a la búsqueda, la ordenación y la codificación.

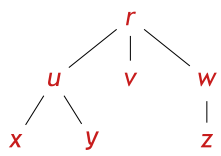
Definición

*Un árbol con raíz se llama **árbol m-ario** si todos los vértices internos tienen, a lo sumo, m hijos.*
*El árbol se llama **m-ario completo** si cada vértice interno tiene exactamente m hijos.*
*Un árbol m-ario con $m = 2$ se llama **árbol binario**.*

Ejemplos



Árbol ternario completo



Árbol ternario (no completo)

4.7 Árboles
Árboles con raíz

¿Cómo se relacionan los números de vértices y de aristas?

Teorema

Un árbol de n *vértices tiene* $n-1$ *aristas.*

Teorema (Propiedades de los árboles con raíz)

Un árbol m *-ario completo con* i *vértices internos tiene* $n = i \cdot m + 1$ *vértices.*

Sea T un árbol m -ario completo y sean n el número de vértices, i el número de vértices internos y l el número de hojas del árbol.

☛ Una vez que algunos de los enteros n, i ó l es conocido, las otras dos cantidades quedan determinadas.

4.7 Árboles
Árboles con raíz

Ejercicio 1

1. Un árbol ternario completo tiene 34 vértices internos. ¿Cuántas aristas tiene? ¿Cuántas hojas?
2. ¿Cuántos vértices internos tiene un árbol 5-ario completo con 817 hojas?

4.7 Árboles
Árboles con raíz

Ejercicio 2

Un aula tiene 25 ordenadores que deben conectarse a un enchufe de pared con cuatro salidas. Se hacen las conexiones mediante cables de extensión con cuatro salidas cada uno.
¿Cuál es el número mínimo de cables que se necesitan para poder utilizar todos los ordenadores?

4.7 Árboles
Árboles con raíz

Ejercicio 3

En una compañía donde trabajan 125 ejecutivos se instala un nuevo sistema de comunicación telefónica. Lo inaugura la presidenta, quien llama a sus cuatro vicepresidentes. A continuación, cada vicepresidente llama a otros cuatro ejecutivos; éstos, a su vez, a otros cuatro y así sucesivamente.

1. ¿Cuántas llamadas son necesarias para comunicar con los 125 ejecutivos?
2. ¿Cuántos ejecutivos hacen llamadas?

4.7 Árboles
Árboles con raíz

Definición

- *El **nivel** de un vértice v en un árbol con raíz es la longitud del único camino desde la raíz hasta dicho vértice. (El nivel de la raíz es cero)*
- *La **altura** de un árbol con raíz es el máximo de los niveles de sus vértices.*
- *Un árbol con raíz m -ario de altura h está **equilibrado** o **balanceado** si todas sus hojas están en los niveles h o $h-1$.*

4.7 Árboles
Árboles con raíz

Teorema

Un árbol m -ario de altura h tiene, a lo sumo, m^h hojas.

Corolario

Si un árbol m -ario de altura h tiene l hojas, entonces $h \geq \lceil \log_m l \rceil$.
Si el árbol m -ario es completo y equilibrado, entonces $h = \lceil \log_m l \rceil$.

(Se recuerda que $\lceil x \rceil$ es el menor entero mayor o igual que x).

Ejercicio

¿Cuál es el número máximo de vértices internos que puede tener un árbol cuaternario completo de altura 8?

4.7 Árboles
Árboles con raíz ordenados

Definición

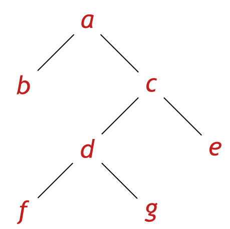
*Un **árbol con raíz ordenado** es un árbol con raíz en el que los hijos de cada vértice interno están ordenados.*

✓ Los árboles ordenados con raíz se dibujan de modo que los hijos de cada vértice interno se colocan ordenados de izquierda a derecha.

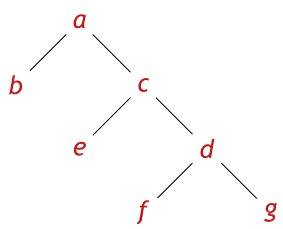
En un árbol binario completo ordenado , el primer hijo de cada vértice interno se llama ***hijo izquierdo*** y el segundo se llama ***hijo derecho***.
El árbol con raíz en el hijo izquierdo se llama ***subárbol izquierdo*** y el árbol con raíz en el hijo derecho se llama ***subárbol derecho***.

4.7 Árboles
Árboles con raíz ordenados

Ejemplos



Árbol T_1



Árbol T_2

Los árboles T_1 y T_2 son iguales como árboles no ordenados.
Sin embargo, como árboles ordenados no son iguales, ya que en T_1 el vértice d es el hijo izquierdo de c y en T_2 el vértice d es el hijo derecho de c.

4.7 Árboles

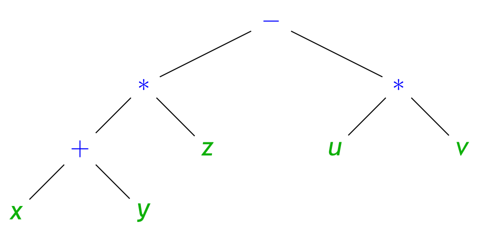
Árboles con raíz ordenados

- Los árboles con raíz ordenados se usan para representar
  - expresiones algebraicas,
  - enunciados,
  - expresiones en un lenguaje, etc
- Las hojas del árbol se etiquetan con variables, constantes, V, F, ... y los vértices interiores se etiquetan con operadores o conectivos.
- Estos árboles deben ser ordenados cuando las operaciones **no** son conmutativas.

4.7 Árboles

Árboles con raíz ordenados

Ejemplo La expresión algebraica $\Big(((x+y)*z)-(u*v)\Big)$ se puede representar mediante un árbol ordenado



4.7 Árboles

Árboles con raíz ordenados

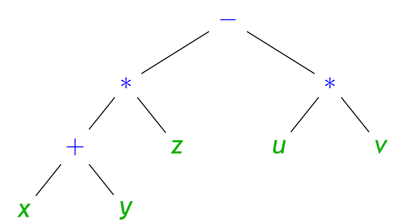
¿Cómo se evalúan expresiones representadas por a. r. o. ?

- El árbol se evalúa empezando desde abajo y asignando valores a cada vértice interior, ya que las expresiones de los paréntesis más profundos se evalúan primero (se ejecutan de dentro hacia fuera).
- A un vértice etiquetado con un operador se le asigna el valor que resulta de realizar la operación sobre los valores de sus hijos.
- Este proceso se puede considerar como un 'plegado' del árbol hacia arriba.
- Se ilustra por una secuencia de árboles.
- Cada árbol de la secuencia se obtiene de su predecesor 'plegando' un subárbol consistente en un vértice y las hojas que son sus hijos.

4.7 Árboles

Árboles con raíz ordenados

Ejemplo $\Big(((x+y)*z)-(u*v)\Big)$

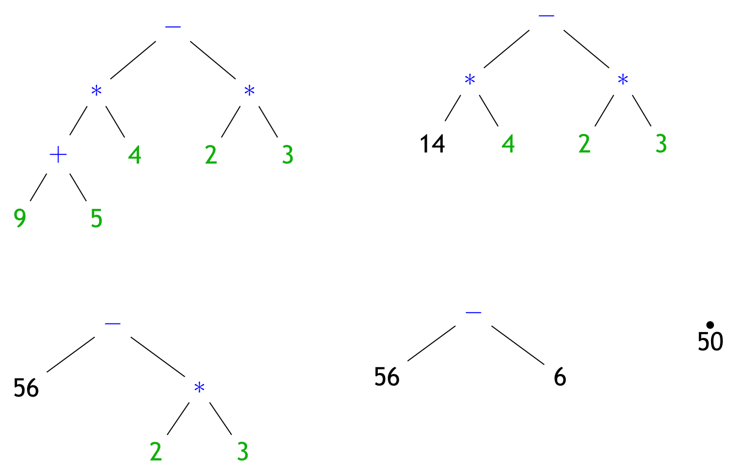


Para evaluarlo asignamos valores a las variables

$x = 9$ $\quad y = 5$ $\quad z = 4$ $\quad u = 2$ $\quad v = 3$

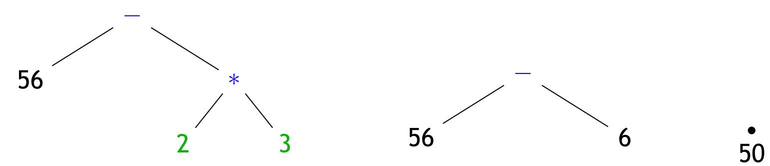
4.7 Árboles

Árboles con raíz ordenados



4.7 Árboles

Árboles con raíz ordenados



- ✓ Este procedimiento se llama **evaluación 'botton-up' de la representación de árbol de la expresión.**
- ✓ Se usa para el análisis sintáctico de una gramática.

4.7 Árboles

Recorridos en Árboles: Introducción

- ✓ Los **á**rboles con **r**aíz **o**rdenados se usan para almacenar información.
- ✓ Se necesitan procedimientos para visitar cada vértice de un **á**rbol con **r**aíz **o**rdenado para acceder a los datos.
  Recorridos: Procedimientos para visitar los vértices del **á.r.o**.
- ✓ Describiremos varios **algoritmos** importantes para visitar todos los vértices de un **á**rbol con **r**aíz **o**rdenado.
  1. Recorrido en ***orden previo*** (Preorder)
  2. Recorrido en ***orden simétrico*** (Inorder) (para árboles binarios)
  3. Recorrido en ***orden posterior*** (Postorder)

4.7 Árboles

Recorridos en Árboles: Orden Previo

1. ***Preorder*** Sea T un árbol con raíz r ordenado.

- Si T solo consta de la raíz, entonces r es el *recorrido en orden previo* de T.
- En otro caso, supongamos que $v_1, v_2, ..., v_k$ son los hijos de r de izquierda a derecha en T y $T_1, T_2, ..., T_k$ los subárboles correspondientes.
- El recorrido en orden previo empieza por visitar la raíz r, continúa recorriendo T_1 en orden previo, después T_2 en orden previo y así sucesivamente hasta que T_k es recorrido en orden previo.

4.7 Árboles

Recorridos en Árboles: Orden Previo

Preorder

- ***Paso 1***: Visitar la raíz r.
- ***Paso 2***: Recorrer T_1 en orden previo.
- ***Paso 3***: Recorrer T_2 en orden previo.

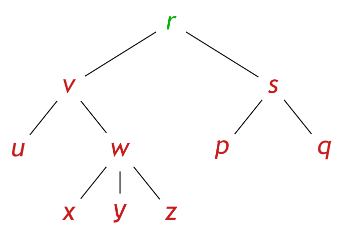
  $\vdots$
- ***Paso k+1***: Recorrer T_k en orden previo.

✓ El recorrido en orden previo de un árbol con raíz ordenado nos da la misma ordenación de los vértices que la ordenación obtenida usando el **sistema universal de direcciones.**

4.7 Árboles

Recorridos en Árboles: Orden Previo

Ejemplo



$$\begin{aligned} T &= \mathbf{r} - \mathbf{T}_v - \mathbf{T}_s \\ &= \mathbf{r} - v - \mathbf{T}_u - \mathbf{T}_w - s - \mathbf{T}_p - \mathbf{T}_q \\ &= r - v - u - w - \mathbf{T}_x - \mathbf{T}_y - \mathbf{T}_z - s - p - q \\ &= r - v - u - w - x - y - z - s - p - q \end{aligned}$$

4.7 Árboles

Recorridos en Árboles: Orden Simétrico

2. ***Inorder*** Sea T un árbol **binario** con raíz r ordenado.

- Si T solo consiste en la raíz, entonces r es el *recorrido en orden simétrico* de T.
- En otro caso, supongamos que v_1, v_2 son los hijos de r de izquierda a derecha en T y T_1, T_2 los subárboles correspondientes.
- El recorrido en orden simétrico empieza recorriendo T_1 en orden simétrico, después visita la raíz r y termina recorriendo T_2 en orden simétrico.

4.7 Árboles

Recorridos en Árboles: Orden Simétrico

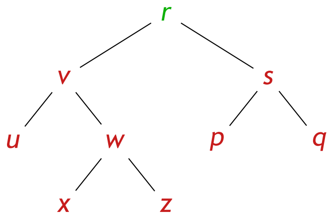
Inorder Sea T un árbol **binario** con raíz r ordenado.

- ***Paso 1***: Recorrer T_1 en orden simétrico.
- ***Paso 2***: Visitar la raíz r.
- ***Paso 3***: Recorrer T_2 en orden simétrico.

4.7 Árboles

Recorridos en Árboles: Orden Simétrico

Ejemplo



$$
\begin{aligned}
T &= \mathbf{T}_v - \mathbf{r} - \mathbf{T}_s \\
&= \mathbf{T}_u - v - \mathbf{T}_w - \mathbf{r} - \mathbf{T}_p - s - \mathbf{T}_q \\
&= u - v - \mathbf{T}_x - w - \mathbf{T}_z - \mathbf{r} - p - s - q \\
&= u - v - x - w - z - \mathbf{r} - p - s - q
\end{aligned}
$$

4.7 Árboles

Recorridos en Árboles: Orden Posterior

3. ***Postorder*** Sea T un árbol con raíz r ordenado.

- Si T solo consiste en la raíz, entonces r es el *recorrido en orden posterior* de T.
- En otro caso, supongamos que $v_1, v_2, \ldots, v_k$ son los hijos de r de izquierda a derecha en T y $T_1, T_2, \ldots, T_k$ los subárboles correspondientes.
- El recorrido en orden posterior empieza por recorrer T_1 en orden posterior, después T_2 en orden posterior y así sucesivamente hasta que T_k es recorrido en orden posterior y termina visitando la raíz r.

4.7 Árboles

Recorridos en Árboles: Orden Posterior

Postorder

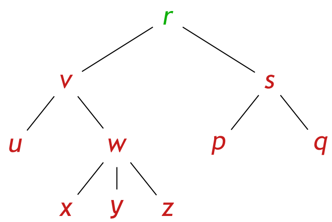
- ***Paso 1***: Recorrer T_1 en orden posterior.
- ***Paso 2***: Recorrer T_2 en orden posterior.

  ⋮
- ***Paso k***: Recorrer T_k en orden posterior.
- ***Paso k+1***: Visitar la raíz r.

4.7 Árboles

Recorridos en Árboles: Orden Posterior

Ejemplo



$$
\begin{aligned}
T &= \mathbf{T}_v - \mathbf{T}_s - \mathbf{r} \\
&= \mathbf{T}_u - \mathbf{T}_w - v - \mathbf{T}_p - \mathbf{T}_q - s - \mathbf{r} \\
&= u - \mathbf{T}_x - \mathbf{T}_y - \mathbf{T}_z - w - v - p - q - s - \mathbf{r} \\
&= u - x - y - z - w - v - p - q - s - \mathbf{r}
\end{aligned}
$$

4.7 Árboles

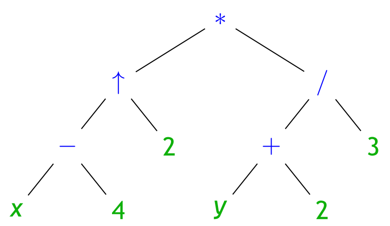
Recorridos en Árboles. Notación polaca

- Los recorridos en preorden, postorden e inorden permiten dar una lista de los vértices de un árbol con raíz ordenado.
- Si este árbol representa una expresión algebraica, los vértices están etiquetados con signos de operaciones, variables y constantes. Y el listado obtenido nos puede servir para evaluar la expresión.
- Por ejemplo, utilizando la notación algebraica usual, la lista $4*3/2$ determina el número 6.
- La lista $4+3*2$ es ambigua, ¿expresa el 10 ó el 14 ?
- A continuación estudiaremos un método, (**la notación polaca**), para definir operaciones algebraicas sin usar paréntesis y sin establecer prioridad entre operadores, utilizando listas obtenidas de árboles.

4.7 Árboles

Recorridos en Árboles. Notación polaca

Ejemplo La expresión algebraica $(x-4)^2\left(\dfrac{y+2}{3}\right)$ se puede escribir $\Big((x-4)\uparrow 2\Big)*\Big((y+2)/3\Big)$ y se representa por el árbol con raíz ordenado



El recorrido en orden previo de este árbol es $*\uparrow - x\; 4\; 2\; / + y\; 2\; 3$

y el recorrido en orden posterior es $x\; 4 - 2 \uparrow y\; 2 + 3\; / \; *$

4.7 Árboles

Recorridos en Árboles. Notación polaca

- Cada uno de estos listados determina de forma única al árbol y, por lo tanto, a la expresión que representa.
- El lógico polaco Lukasiewicz demostró que estas expresiones no son ambiguas sin paréntesis.
- El listado en preorden se conoce como **notación (polaca) prefija**, el listado en orden posterior como **notación (polaca) inversa** o **sufija** y la notación usual, con los paréntesis necesarios, se llama **notación infija.**

4.7 Árboles

Recorridos en Árboles. Notación polaca

Conociendo el número de hijos de cada vértice interior, a partir de la expresión $x\; 4 - 2 \uparrow y\; 2 + 3\; / \; *$ en **notación sufija** podemos recuperar la expresión algebraica inicial.

$$x\; 4 - 2 \uparrow y\; 2 + 3\; / \; *$$

$$\underbrace{x\; 4\; -}\; 2 \uparrow y\; 2 + 3\; / \; *$$

$$\underbrace{(x-4)\; 2 \uparrow}\; y\; 2 + 3\; / \; *$$

$$\Big((x-4)\uparrow 2\Big)\; \underbrace{y\; 2\; +}\; 3\; / \; *$$

$$\Big((x-4)\uparrow 2\Big)\; (y+2)\; 3\; / \; *$$

$$\Big((x-4)\uparrow 2\Big)\; \underbrace{(y+2)\; 3\; /}\; *$$

$$\underbrace{\Big((x-4)\uparrow 2\Big)\; \Big((y+2)/3\Big)\; *}$$

$$\Big((x-4)\uparrow 2\Big)*\Big((y+2)/3\Big) = (x-4)^2\left(\frac{y+2}{3}\right)$$

4.7 Árboles

Recorridos en Árboles. Notación polaca

Ejercicio Dada la expresión

$$(\neg(p \vee (q \wedge r))) \vee ((\neg p \vee q) \wedge (p \to r))$$

1. Represéntala mediante un árbol.
2. Describe algún procedimiento para determinar el recorrido en orden previo y el recorrido en orden posterior y utilízalo en el árbol anterior.

4.7 Árboles

Recorridos en Árboles. Notación polaca

Ejercicio

1. Define *recorrido en orden previo* y *recorrido en orden posterior*.
2. Dada la expresión

$$((p_1 \vee p_2) \to q) \leftrightarrow ((p_1 \to q) \wedge (p_2 \to q))$$

   1. Represéntala mediante un árbol.
   2. Describe algún procedimiento para determinar el recorrido en orden previo y el recorrido en orden posterior y utilízalo en el árbol anterior.

4.7 Árboles

Recorridos en Árboles. Notación polaca

Ejercicio Dada la expresión algebraica

$$((((A + B) * C) + D) * E) - ((A + B) * C)$$

1. Represéntala mediante un árbol binario.
2. Escribe su representación prefija (polaca) y postfija (polaca inversa).

4.7 Árboles

Recorridos en Árboles. Notación polaca

Ejercicio Se sabe que una expresión algebraica se representa en notación sufija como

$$AB + CD * EF / - - A *$$

1. Traza un árbol binario que represente a dicha expresión algebraica.
2. Escribe su expresión en notación infija y en notación prefija.

4.7 Árboles
Árboles de Búsqueda

- Para determinar una propiedad de un grafo, a menudo hay que explorar cada uno de los vértices y aristas en algún orden razonable.
- Son importantes dos recorridos:
  1. ***Búsqueda en anchura*** (B.E.A.)
  2. ***Búsqueda en profundidad*** (B. E. P.)

4.7 Árboles
Árboles de Búsqueda: Búsqueda en anchura (B.E.A.)

- Dado un un grafo conexo simple G, podemos obtener un árbol generador mediante la **búsqueda en anchura** o **por niveles**.
- Se construye un árbol con raíz y el grafo no dirigido subyacente es el árbol generador.

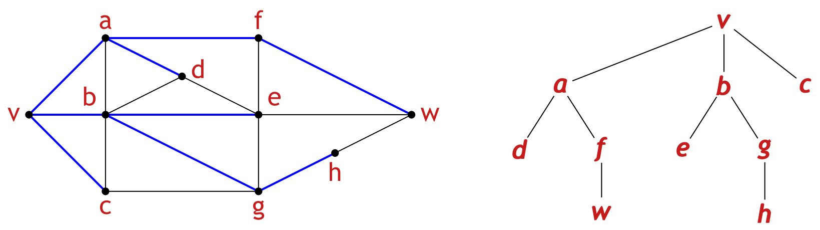
4.7 Árboles
Árboles de Búsqueda: Búsqueda en anchura (B.E.A.)

Algoritmo B. E. A.

- Se elige un vértice arbitrario como raíz y se designa v_0.
- A continuación se añaden todas las aristas incidentes con ese vértice.
- Los nuevos vértices añadidos en esta fase forman los vértices del nivel 1 del árbol generador. Los ordenamos en cualquier orden.
- Para cada vértice del nivel 1, visitados en orden, se añaden todas las aristas incidentes con él (siempre que **no** formen ciclo).
- Los nuevos vértices añadidos a esta fase forman los vértices del nivel 2 del árbol.
- Seguimos el mismo procedimiento hasta que se hayan añadido al árbol todos los vértices del grafo G.
- Este proceso termina, ya que el conjunto de vértices del grafo G es finito.

4.7 Árboles
Árboles de Búsqueda: Búsqueda en anchura (B.E.A.)

Ejemplo Empezando en el vértice ***v***, haz una búsqueda en anchura y representa la secuencia de búsqueda de los vértices con un árbol con raíz.



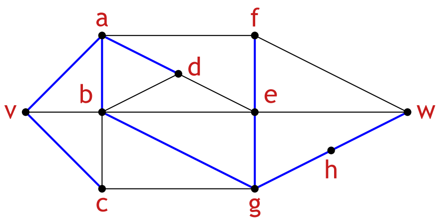
Algoritmo de Búsqueda en Profundidad (B.E.P.)

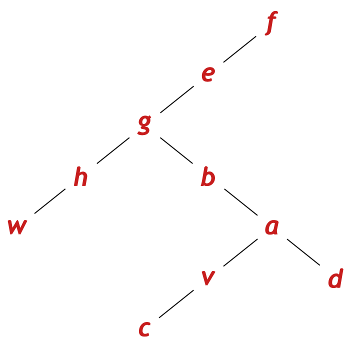
- Se elige como raíz un vértice cualquiera del grafo.
- Formamos un camino que comienza en este vértice añadiendo sucesivamente aristas y vértices, siendo cada nueva arista incidente con el último vértice del camino y un vértice que no está en el camino.
- Añadimos a este camino tantos vértices y aristas como sea posible.
- Si el camino pasa por todos los vértices del grafo, entonces el árbol generador es dicho camino.
- En caso contrario, retrocedemos al penúltimo vértice del camino y, si es posible, formamos un nuevo camino que empiece en este vértice y que pase por vértices no visitados.
- Si esto no se puede hacer, retrocedemos al vértice anterior en el recorrido hasta la raíz y lo intentamos de nuevo.
- Repetimos el proceso hasta que no se puedan añadir más aristas.

4.7 Árboles

Árboles de Búsqueda: Búsqueda en Profundidad (B.E.P.)

Ejemplo Empezando en el vértice f , haz una búsqueda en profundidad y representa la secuencia de búsqueda de los vértices con un árbol con raíz.





4.7 Árboles

Árboles de Búsqueda

1. La búsqueda en anchura se usará para problemas tales como:
   - colorear grafos bipartitos,
   - encontrar el camino más corto en un grafo,
   - determinar la distancia más corta desde un vértice a otro del grafo.
2. La búsqueda en profundidad se aplicará para:
   - encontrar componentes conexas
   - comprobar si un grafo es acíclico.

4.7 Árboles

Árboles de Búsqueda

Ejercicio Considera el grafo dado por las listas de adyacencia

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	1	4	3	1	2	6	9	3	4	6	7	8	9	12	13
5	6	9	10	6	5	12	13	8	9	12	11	12	13	16	15
					7			10			13	14			
					11			14			15	16			

1. Determina si es conexo haciendo una búsqueda en profundidad y representa la secuencia de búsqueda de los vértices con un árbol con raíz.
2. Idem mediante una búsqueda en anchura.

4.8 Grafos Ponderados

Introducción

Muchos problemas se pueden representar usando grafos en los que se asigna un número (peso) a cada una de sus aristas.

Ejemplo Para una línea aérea construimos el modelo básico representando las ciudades mediante vértices y los vuelos mediante aristas.

- Los problemas relacionados con ***distancias*** entre ciudades se pueden representar asignándoles a las aristas las distancias entre ciudades.
- Los problemas relacionados con ***tiempos*** de vuelo se pueden representar asignándoles a las aristas los tiempos de vuelo corespondientes.
- Los problemas relacionados con ***tarifas*** de vuelo se pueden representar asignándoles a las aristas los precios de los billetes.

4.8 Grafos Ponderados

Definición

Sea $G = (V, E)$ un grafo (o digrafo) simple.
*Se llama **peso** a una función*

$$\begin{array}{rcl} w : E & \to & \mathbb{R}^+ \\ e & \mapsto & w(e) \end{array}$$

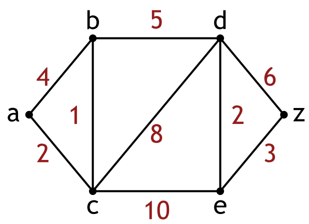
*A la terna $G = (V, E, w)$ se le llama **grafo ponderado**.*

4.8 Grafos Ponderados

Representación de grafos ponderados

- **Gráficamente**, etiquetando cada arista con su peso $w(e)$,
- **Matricialmente**, mediante la matriz de pesos $W(G) = (w_{i,j})$, donde previamente se ha listado ordenadamente el conjunto de vértices. Cada elemento $w_{i,j}$ es
  - el peso de la arista $\{v_i, v_j\}$ o el arco (v_i, v_j) ; o bien
  - un símbolo para indicar que v_i no es adyacente a v_j.

Ejemplo



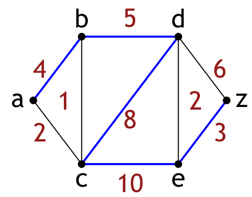
$$W(G) = \begin{pmatrix} 0 & 4 & 2 & - & - & - \\ 4 & 0 & 1 & 5 & - & - \\ 2 & 1 & 0 & 8 & 10 & - \\ - & 5 & 8 & 0 & 2 & 6 \\ - & - & 10 & 2 & 0 & 3 \\ - & - & - & 6 & 3 & 0 \end{pmatrix}$$

4.8 Grafos Ponderados

Definición

*En un grafo ponderado se llama **longitud** de un camino a la suma de los pesos de sus aristas.*

Ejemplo La longitud del camino $a - b - d - c - e - z$ es 30



En un grafo ponderado nos planteamos los siguientes problemas:

1. hallar un árbol generador de peso mínimo.
2. encontrar un camino de longitud mínima entre dos vértices.

Definición

Un ***árbol generador minimal*** *de un grafo conexo ponderado es un árbol generador tal que la suma de los pesos de sus aristas es mínima.*

Para hallar árboles generadores minimales usaremos:

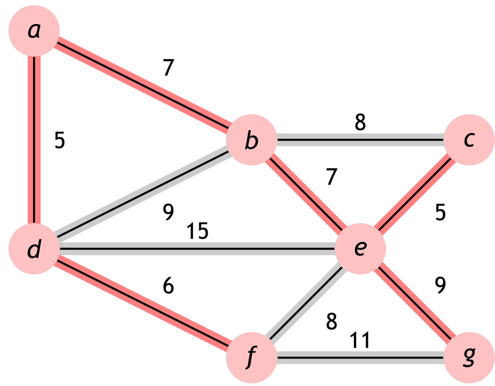
- **Algoritmo de Prim**
- **Algoritmo de Kruskal**

Algoritmo de Prim

Sea $G = (V, E)$ un grafo conexo ponderado, donde $V = \{v_1, v_2, ..., v_p\}$.

1. Sea $i = 1$, $V_1 = \{\mathbf{v}\}$, donde $\mathbf{v}$ es un vértice cualquiera y $E_1 = \varnothing$.
2. Para $U_i = V - V_i$, hallamos $w_i = min\{w(\{\mathbf{x}, \mathbf{y}\}),\ \mathbf{x} \in V_i,\ \mathbf{y} \in U_i\}$.
   Sea $\mathbf{e_i} = \{\mathbf{x_i}, \mathbf{y_i}\}$ una arista para la que se alcanza este mínimo.
3. Se incrementa i en 1.
   Sea $V_i = V_{i-1} \cup \{\mathbf{y_i}\}$ y $E_i = E_{i-1} \cup \{\mathbf{e_i}\}$.
4.
   - Si $i < p$, volver al paso 2.
   - En otro caso, **parar** porque el subgrafo $T = (V_p, E_p)$ es un árbol generador minimal del grafo G.

Ejemplo



Ejercicio El estudio de localización de terminales de ordenadores que van a ser instalados en una empresa viene dado por la siguiente tabla, donde los números representan el coste de instalar las conexiones entre los distintos terminales. El terminal C corresponde al ordenador principal y el resto de los terminales deben estar conectados a él mediante líneas telefónicas.

Halla la manera en que todos los terminales estén conectados a C, (directa o indirectamente), siendo mínimo el coste total de la instalación.

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*
A	—	2	5	10	—	—	—	—
B	2	—	—	—	6	7	—	9
C	5	—	—	11	—	—	12	—
D	10	—	11	—	1	—	—	—
E	—	6	—	1	—	14	13	4
F	—	7	—	—	14	—	—	8
G	—	—	12	—	13	—	—	3
H	—	9	—	—	4	8	3	—

4.8 Grafos ponderados

Caminos de longitud mínima

Nos planteamos el problema de encontrar el camino de longitud mínima entre dos vértices de un grafo ponderado. Por ejemplo,

- en una línea aérea:
  - ¿cual es la combinación de vuelos que tiene el tiempo de vuelo total más pequeño?
  - ¿cuál es la tarifa más barata para viajar entre dos ciudades?
- en una red de ordenadores:
  - ¿cuál es el conjunto de líneas de teléfono menos costosa que se necesita para conectar los ordenadores de Málaga con los de Barcelona?,
  - ¿cuál es el conjunto de líneas de teléfono que da el tiempo de respuesta más rápido para las comunicaciones entre Málaga y Barcelona?

4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

- Sea $G = (V, E, w)$ un grafo conexo simple ponderado.
- G tiene n vértices $a = v_0, v_1, \ldots, v_n = z$ y pesos $w(v_i, v_j) > 0$
- Se busca el camino de longitud mínima entre los vértices a y z.
- El algoritmo de Dijkstra procede determinando
  - la longitud de un camino de longitud mínima entre a y un primer vértice,
  - la longitud de un camino de longitud mínima entre a y un segundo vértice,
  - y así sucesivamente, hasta determinar la longitud de un camino de longitud mínima entre a y el vértice z.

4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

Descripción informal del algoritmo

- El algoritmo actúa realizando una serie de iteraciones.
- Se construye un conjunto de vértices distinguidos, añadiendo un vértice en cada iteración.
- También se realiza un proceso de etiquetado en cada iteración.
- En este proceso de etiquetado, a cada vértice v se le pone una etiqueta que es la longitud de un camino de longitud mínima entre a y v, que contenga sólo vértices del conjunto de vértices distinguidos.
- El vértice que se añade al conjunto de vértices distinguidos es cualquiera que tenga una etiqueta minimal entre aquellos que no están en el conjunto.

4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

- Se empieza asignando la etiqueta 0 al vértice inicial y a los demás vértices la etiqueta ∞.
  - ✓ Usamos la notación $l_0(a) = 0$, $l_0(v) = \infty$ para estas etiquetas (el subíndice 0 indica la iteración 0).
  - ✓ En cada iteración consideramos los caminos que contienen vértices del conjunto de vértices distinguidos.
  - ✓ Las etiquetas $l_0(v)$ indican las longitudes de los caminos más cortos desde a hasta cada vértice v, (considerando los caminos que sólo contienen el vértice a).
  - ✓ Ya que todavía no existe ningún camino desde a hasta vértices distintos de a, la longitud del camino más corto entre a y cada vértice distinto de a es ∞.

4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

- El algoritmo de DIJKSTRA prosigue formando un conjunto de vértices distinguidos o destacados.
  Se denota S_k el conjunto de vértices distinguidos después de la iteración k del proceso de etiquetado.
- Empezamos con $S_0 = \varnothing$. Para $k \geq 1$, el conjunto $S_k = S_{k-1} \cup \{u\}$, donde $u \notin S_{k-1}$ y la etiqueta $l_{k-1}(u)$ es mínima entre los vértices que no están en S_{k-1}.
- Una vez que u es añadido a S_{k-1}, se actualizan todas las etiquetas de los vértices que no están en S_k de modo que $l_k(v)$, la etiqueta del vértice v en la etapa k, es la longitud de un camino de longitud mínima entre a y v, (que sólo contiene vértices de S_k).

4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

- La actualización se puede realizar eficientemente teniendo en cuenta que el camino más corto entre a y v que solo contiene vértices de S_k es:
  - ✓ el camino más corto entre a y u en la etapa $k-1$ añadiéndole la arista $\{u, v\}$; o bien
  - ✓ el camino más corto entre a y v que contiene sólo elementos de S_{k-1} (es decir, u no está incluido).

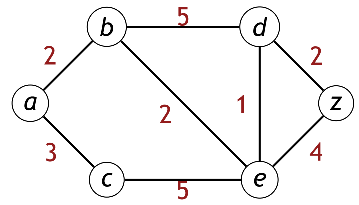
$$l_k(v) = min\{l_{k-1}(v), l_{k-1}(u) + w(u, v)\}$$

- Este proceso se repite añadiendo sucesivamente vértices al conjunto de vértices distinguidos hasta que se añade el vértice z.
- Cuando se añade z al conjunto de vértices distinguidos, su etiqueta es la longitud del camino más corto entre a y z.

4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

Ejemplo 1 Usa el algoritmo de Dijkstra para hallar el camino más corto entre los vértices a y z en el grafo



4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

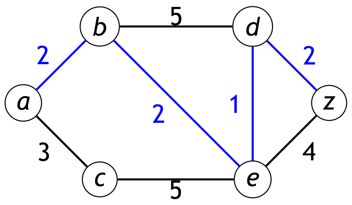
Solución Ejemplo 1: Al aplicar el algoritmo, usamos una tabla para escribir las etiquetas de cada vértice y los conjuntos de vértices distinguidos.

Los subíndices de las etiquetas indican el penúltimo vértice en el camino de longitud mínima.

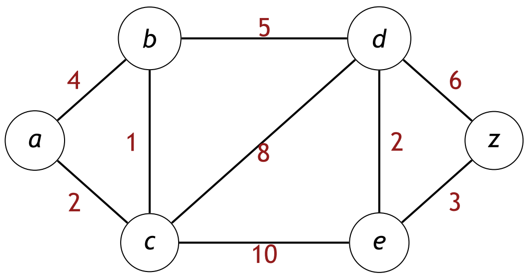
Vértices	a	b	c	d	e	z	**Caminos**
$S_0 = \varnothing$	$\boxed{0}$	∞	∞	∞	∞	∞	a
$S_1 = \{a\}$	–	$\boxed{2}_a$	3_a	∞	∞	∞	ab, ac
$S_2 = S_1 \cup \{b\}$	–	–	$\boxed{3}_a$	7_b	4_b	∞	abd, abe
$S_3 = S_2 \cup \{c\}$	–	–	–	7_b	$\boxed{4}_b$	∞	
$S_4 = S_3 \cup \{e\}$	–	–	–	$\boxed{5}_e$	–	8_e	$abed, abez$
$S_5 = S_4 \cup \{d\}$	–	–	–	–	–	$\boxed{7}_d$	$abedz$

El camino buscado es: $\mathbf{a - b - e - d - z}$, que tiene longitud **7**.

Solución Ejemplo 1:



Ejemplo 2 Usa el algoritmo de Dijkstra para hallar el camino más corto entre los vértices a y z en el grafo

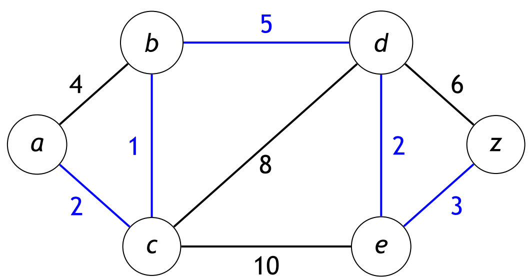


Solución Ejemplo 2:

Vértices	a	b	c	d	e	z	**Caminos**
$S_0 = \varnothing$	$\boxed{0}$	∞	∞	∞	∞	∞	a
$S_1 = \{a\}$	$-$	4_a	$\boxed{2}_a$	∞	∞	∞	ab, ac
$S_2 = S_1 \cup \{c\}$	$-$	$\boxed{3}_c$	$-$	10_c	12_c	∞	acb, acd, ace
$S_3 = S_2 \cup \{b\}$	$-$	$-$	$-$	$\boxed{8}_b$	12_c	∞	$acbd$
$S_4 = S_3 \cup \{d\}$	$-$	$-$	$-$	$-$	$\boxed{10}_d$	14_d	$acbde,\ acbdz$
$S_5 = S_4 \cup \{e\}$	$-$	$-$	$-$	$-$	$-$	$\boxed{13}_e$	$acbdez$

El camino buscado es: $\boldsymbol{a - c - b - d - e - z}$, que tiene longitud **13**.

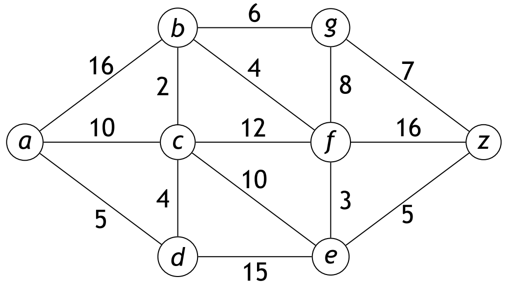
Solución Ejemplo 2:



4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

Ejemplo 3 En el grafo de la figura se representa una red ferroviaria donde la distancia entre cada par de ciudades se expresa en km:



- Halla el camino más corto para viajar de a hasta z.

4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

Solución Ejemplo 3:

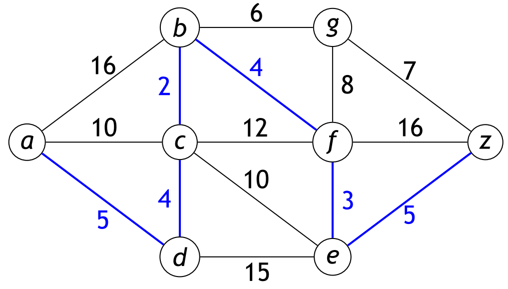
Vértices	a	b	c	d	e	f	g	z	Caminos
$S_1 = \{a\}$	–	16_a	10_a	$\boxed{5}_a$	∞	∞	∞	∞	ab, ac, ad
$S_2 = S_1 \cup \{d\}$	–	16_a	$\boxed{9}_d$	–	20_d	∞	∞	∞	adc, ade
$S_3 = S_2 \cup \{c\}$	–	$\boxed{11}_c$	–	–	19_c	21_c	∞	∞	$adcb, adc$
$S_4 = S_3 \cup \{b\}$	–	–	–	–	19_c	$\boxed{15}_b$	17_b	∞	$adcbf, ad$
$S_5 = S_4 \cup \{f\}$	–	–	–	–	18_f	–	$\boxed{17}_b$	31_f	$adcbfe, a$
$S_6 = S_5 \cup \{g\}$	–	–	–	–	$\boxed{18}_f$	–	–	24_g	$adcbgz$
$S_7 = S_6 \cup \{e\}$	–	–	–	–	–	–	–	$\boxed{23}_e$	$adcbfez$

El camino buscado es: $\boldsymbol{a - d - c - b - f - e - z}$, que tiene longitud **23**.

4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

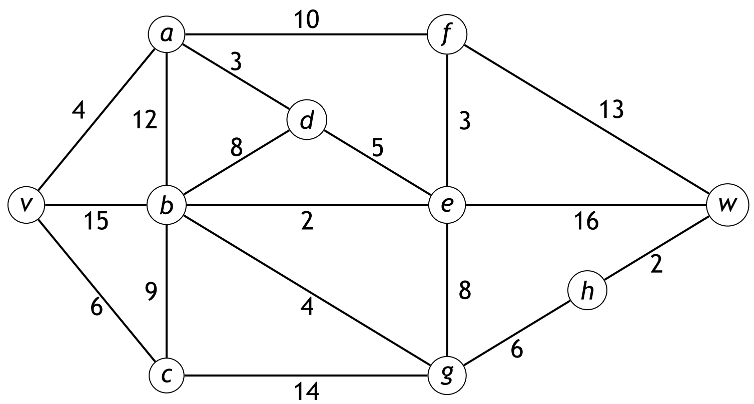
Solución Ejemplo 3:



4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

Ejemplo 4 Usa el algoritmo de Dijkstra para hallar el camino más corto entre los vértices v y w en el grafo



4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

Solución Ejemplo 4:

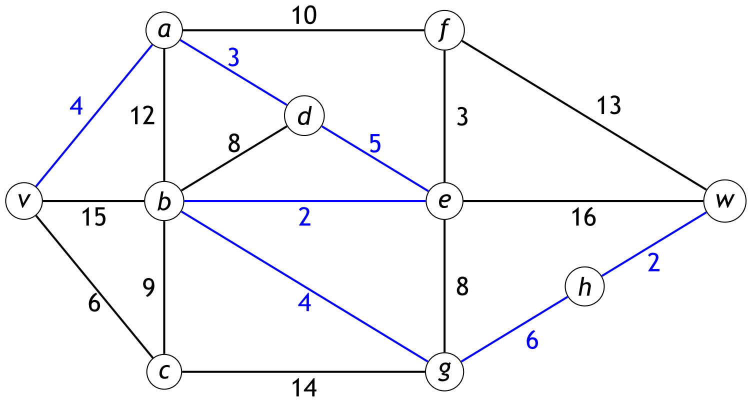
Vértices	v	a	b	c	d	e	f	g	h	w	
$S_1 = \{v\}$	—	$\boxed{4}_v$	15_v	6_v	∞	∞	∞	∞	∞	∞	
$S_2 = S_1 \cup \{a\}$	—	—	15_v	$\boxed{6}_v$	7	∞	14_a	∞	∞	∞	
$S_3 = S_2 \cup \{c\}$	—	—	15_v	—	$\boxed{7}_a$	∞	14_a	20_c	∞	∞	
$S_4 = S_3 \cup \{d\}$	—	—	15_v	—	—	$\boxed{12}_d$	14_a	20_c	∞	∞	
$S_5 = S_4 \cup \{e\}$	—	—	$\boxed{14}_e$	—	—	—	14_a	20_c	∞	28_e	va
$S_6 = S_5 \cup \{b\}$	—	—	—	—	—	—	$\boxed{14}_a$	18_b	∞	28_e	
$S_7 = S_6 \cup \{f\}$	—	—	—	—	—	—	—	$\boxed{18}_b$	∞	27_f	
$S_8 = S_7 \cup \{g\}$	—	—	—	—	—	—	—	—	$\boxed{24}_g$	27_f	
$S_9 = S_8 \cup \{h\}$	—	—	—	—	—	—	—	—	—	$\boxed{26}_h$	

El camino buscado es: $v - a - d - e - b - g - h - w$, que es de longitud 26.

4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

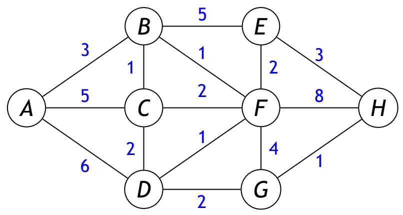
Solución Ejemplo 4:



4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

Ejercicio El grafo de la figura muestra la conexión entre 8 centros de comunicación. Los vértices representan a los centros y las aristas a los canales de comunicación. Los tiempos de transmisión están representados por los pesos de las aristas. Supongamos que a las 7:00 el centro de comunicaciones A transmite una noticia a través de todos sus canales. Los otros centros transmitirán esa noticia tan pronto como la reciban. Usa el algoritmo de Dijkstra para determinar el menor tiempo en que cada uno de los centros B, C, D, E, F, G y H recibe la noticia.



4.8 Grafos ponderados

Caminos de longitud mínima: ALGORITMO DE DIJKSTRA

Ejercicio En la tabla siguiente se indican las conexiones, (en coste por unidades de longitud de cable), de los ordenadores $A, B, C, D, E, F, G, H, I, J$ de los empleados de una empresa, conectados entre sí en una red.

	B	C	D	E	F	G	H	I	J
A	22	15	—	—	14	—	—	8	—
B	—	—	10	12	—	9	—	—	11
C	—	—	—	18	—	—	—	—	—
D	—	—	—	—	—	—	11	—	—
E	—	—	—	—	13	—	—	7	—
G	—	—	—	—	—	—	16	—	—

En la red se producen fallos y se han contratado los servicios de un técnico para localizarlos. Como el coste de reparación es demasiado elevado, se decide reparar lo indispensable para que los ordenadores A y H queden conectados por tramos renovados. ¿Cuáles serían los tramos a reparar?