Generalidades de los árboles

El árbol como grafo, como estructura de datos y como definición recursiva

Carlos A Delgado S

Pontificia Universidad Javeriana Cali

Contenido

1 Antes de empezar

2 El árbol como grafo

3 El árbol como estructura de datos

4 La especificación recursiva

5 Los recorridos

6 La excentricidad

7 El diámetro de un árbol

8 Errores comunes

9 Ejercicios

10 Lo que sigue

11 Referencias

Contenido

1 Antes de empezar

2 El árbol como grafo

3 El árbol como estructura de datos

4 La especificación recursiva

5 Los recorridos

6 La excentricidad

7 El diámetro de un árbol

8 Errores comunes

9 Ejercicios

10 Lo que sigue

11 Referencias

Los árboles ya aparecieron

Dónde

- La búsqueda en amplitud deja un árbol de descendientes, y las distancias que calcula son las de ese árbol.
- La búsqueda en profundidad deja el árbol que clasifica las aristas en de árbol, de regreso, de avance y cruzadas.
- El árbol de puentes resume un grafo conexo: cada componente sin puentes se vuelve un nodo y lo que queda es un árbol.

Qué falta

Formalizarlos. Hasta ahora el árbol fue el resultado de un recorrido; de aquí en adelante es el objeto de estudio, con sus propiedades y sus algoritmos propios.

Tres formas de mirar lo mismo

Como grafo

Un grafo no dirigido, conexo y acíclico. Es la vista que sirve para demostrar propiedades y para reutilizar los algoritmos de grafos.

Como estructura de datos

Un conjunto de nodos enlazados donde cada nodo tiene un padre y varios hijos. Es la vista que sirve para programar.

Como definición recursiva

Un árbol es vacío, o es un nodo con árboles colgando. Es la vista que sirve para escribir funciones que se llaman a sí mismas.

Tres formas de mirar lo mismo

Pregunta

Las tres describen el mismo objeto. ¿Cuál de ellas necesita que se escoja una raíz?

Contenido

1 Antes de empezar

2 El árbol como grafo

3 El árbol como estructura de datos

4 La especificación recursiva

5 Los recorridos

6 La excentricidad

7 El diámetro de un árbol

8 Errores comunes

9 Ejercicios

10 Lo que sigue

11 Referencias

La definición

Definición (Árbol, CLRS Apéndice B.5.1, p. 1173)

Un árbol es un grafo no dirigido, conexo y acíclico.

Las tres palabras cuentan

- No dirigido: las aristas no tienen sentido.
- Conexo: desde cualquier vértice se llega a cualquier otro.
- Acíclico: no hay ningún camino que salga y vuelva al mismo vértice sin repetir aristas.

Definición (Bosque)

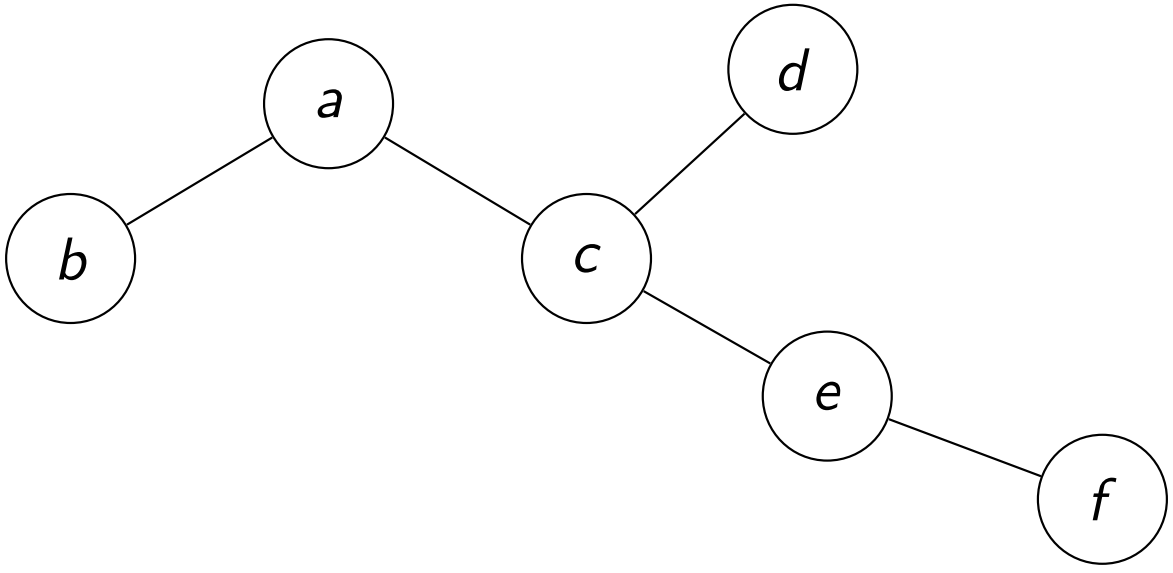
Un grafo no dirigido y acíclico, no necesariamente conexo. Cada una de sus componentes conexas es un árbol.

La definición

Respuesta de la pregunta anterior

Ninguna de las tres necesita raíz para existir, pero las dos últimas sí para escribirse: hay que decir cuál nodo no tiene padre.

Un árbol de seis vértices



$|V| = 6$ y $|E| = 5$. Conexo y sin ciclos.

No hay raíz dibujada ni hace falta: así es como el grafo llega.

Pregunta

¿Cuántos caminos simples hay de b a f? ¿Y de d a b?

Para cada pareja hay un solo camino

Teorema

En un árbol, cada pareja de vértices está unida por exactamente un camino simple.

Demostración

Se procede por contradicción. Por conexidad hay al menos un camino entre u y v. Supóngase que hubiera dos distintos. Entonces se separan en algún vértice y se vuelven a juntar en otro, y el tramo de uno seguido del tramo del otro al revés es un ciclo. Eso contradice que el grafo sea acíclico.

Para cada pareja hay un solo camino

Conclusión

Por lo tanto, se puede concluir que el camino es único. Esa unicidad es la que hace que la distancia entre dos vértices de un árbol no tenga alternativas que comparar.

Hay $n - 1$ aristas y todas son puentes

Teorema

Un árbol de n vértices tiene $n - 1$ aristas, y quitar cualquiera de ellas lo desconecta.

Por qué son puentes

Un puente es una arista cuya eliminación parte el grafo. Si una arista $\{u, v\}$ no fuera puente, al quitarla seguiría habiendo un camino de u a v, y ese camino más la arista cerraría un ciclo.

Por qué son $n - 1$

Se procede por inducción sobre n. Con $n = 1$ no hay aristas. Con $n > 1$, un árbol tiene al menos un vértice de grado 1: quitarlo deja un árbol de $n - 1$ vértices, que por hipótesis tiene $n - 2$ aristas, y se devuelve la que se había quitado.

Hay $n - 1$ aristas y todas son puentes

Lo que esto permite

Un grafo conexo con $n - 1$ aristas es un árbol, y uno acíclico con $n - 1$ aristas también. Son dos formas baratas de reconocerlo sin recorrerlo entero.

Cualquier nodo puede ser la raíz

Qué significa

El grafo no trae raíz. Escoger una es decidir desde dónde se mira: se toma un vértice, se cuelga de él y las aristas quedan apuntando hacia afuera. El grafo no cambió, cambió el dibujo.

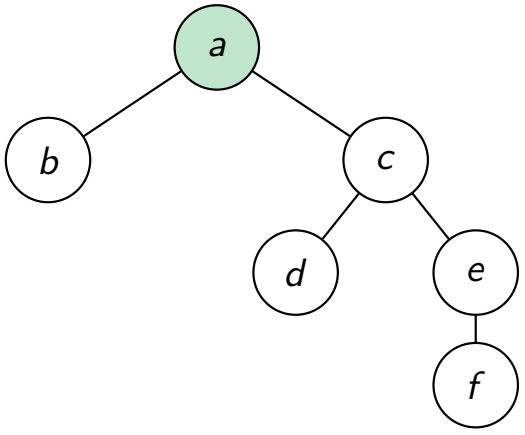
Y sirve

Varios algoritmos piden una raíz para arrancar, y como la elección es libre, se toma la que convenga: el vértice 0, una hoja, o el centro del árbol.

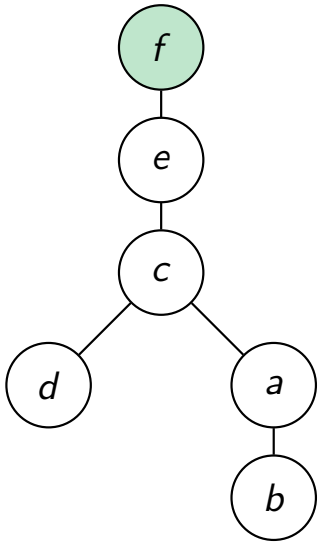
Lo que sí cambia

Quién es padre de quién, la profundidad de cada nodo y la altura del árbol. Lo que no cambia son las aristas, las distancias entre vértices y cuáles son las hojas, salvo la que se haya escogido como raíz.

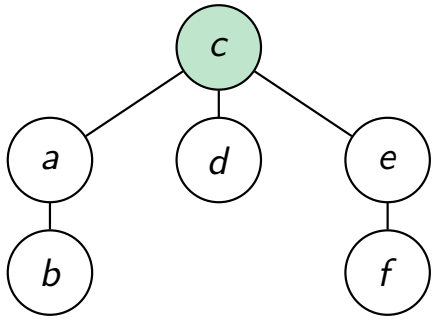
El mismo árbol, tres raíces



colgado de a



colgado de f



colgado de c

Las cinco aristas son las mismas en los tres dibujos. Colgado de a, el árbol tiene altura 3; colgado de c, altura 2.

Las hojas son los vértices de grado 1

En el árbol sin raíz

Una hoja es un vértice con una sola arista. En el árbol de seis vértices son b, d y f.

Con raíz hay una excepción

Una hoja es un nodo sin hijos. La raíz puede tener grado 1 y aun así no ser hoja: colgado de a, el vértice a tiene grado 2, pero colgado de f, el vértice f tiene grado 1 y es la raíz, no una hoja.

Todo árbol con más de un vértice tiene al menos dos hojas

Si todos los vértices tuvieran grado ≥ 2, la suma de los grados sería $\geq 2n$, y como la suma de los grados es $2|E| = 2(n-1)$, quedaría $2n \leq 2n-2$.

Contenido

1 Antes de empezar

2 El árbol como grafo

3 El árbol como estructura de datos

4 La especificación recursiva

5 Los recorridos

6 La excentricidad

7 El diámetro de un árbol

8 Errores comunes

9 Ejercicios

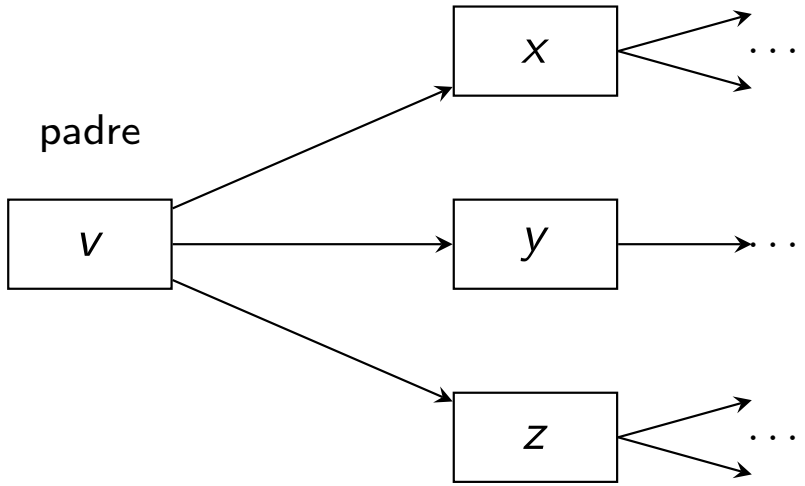
10 Lo que sigue

11 Referencias

Nodos enlazados

La definición

Un conjunto de nodos enlazados donde cada nodo tiene un único padre y varios hijos. El único sin padre es la raíz.



Aquí sí hay sentido: del padre al hijo. Es el mismo árbol de antes, con una raíz escogida y las aristas orientadas.

Cómo se llama cada cosa

Parentesco

Si la arista va de u a v, entonces u es el padre de v y v es hijo de u. Dos hijos del mismo padre son hermanos. Los que quedan en el camino de la raíz a v, sin contarlo, son sus ancestros, y todos los que cuelgan de v son sus descendientes.

Posición

- Hoja: un nodo sin hijos. Interno: uno que sí tiene.
- Subárbol de v: el propio v con todos sus descendientes, que es un árbol por su cuenta.
- Profundidad de v: cuántas aristas hay de la raíz a v. La raíz tiene profundidad 0.
- Altura del árbol: la mayor profundidad.

Cómo se llama cada cosa

Grado

En el árbol sin raíz, el grado de un vértice es cuántas aristas toca. Con raíz, suele interesar cuántos hijos tiene, que es el grado menos uno para todos menos la raíz.

El grado y la aridad

Definición (Grado de un nodo)

En un árbol con raíz, el grado de un nodo es su cantidad de hijos. Una hoja tiene grado 0*.*

Definición (Aridad del árbol)

El mayor de los grados, o la cota que imponga el problema. Un árbol de aridad 2 es binario; uno de aridad m es m-ario; cuando no hay cota se dice que la ramificación es libre.

Para qué sirve saberla

Decide qué representación cabe. Con aridad 2 bastan dos campos por nodo. Con aridad m conocida y pequeña, un arreglo de m casillas. Sin cota, reservar un arreglo por nodo desperdicia o se queda corto, y hay que buscar otra cosa.

El grado y la aridad

No confundir

El grado de un nodo en el árbol sin raíz cuenta todas sus aristas, incluida la del padre. En el árbol con raíz cuenta solo los hijos. Son dos números distintos para el mismo nodo.

Todo nodo es la raíz de un subárbol

Definición (Subárbol)

El subárbol de v es v junto con todos sus descendientes, y las aristas entre ellos.

Y es un árbol de pleno derecho

Es conexo, es acíclico y tiene su propia raíz, que es v. No hay nada que lo distinga de un árbol cualquiera, salvo que está metido dentro de otro.

Lo que esto habilita

Una función sobre un árbol se escribe mirando la raíz y pidiéndole lo mismo a cada subárbol. En el árbol de siete nodos, contar los del subárbol de b da 3, y el que cuenta es el mismo código que cuenta los 7 del árbol entero.

Todo nodo es la raíz de un subárbol

Pregunta

¿Cuántos subárboles distintos tiene un árbol de n nodos?

Cómo se guarda

Respuesta

Exactamente n, uno por nodo, porque cada nodo es la raíz de uno y solo uno.

Qué se le suele pedir a la representación

- El padre de un nodo.
- La lista de sus hijos.
- Si es hoja.
- La profundidad de un nodo o la altura del árbol.

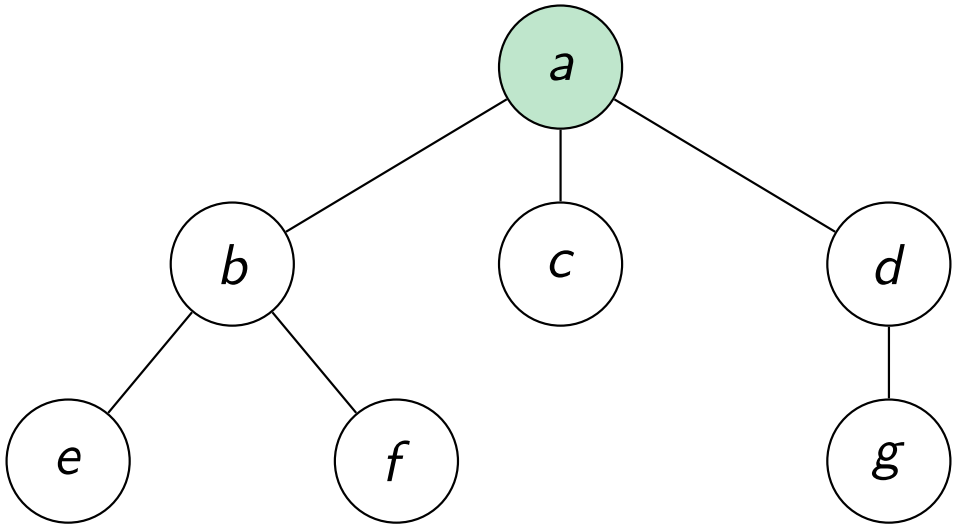
Cómo se guarda

Las que presenta CLRS, Sección 10.4

1. Árboles binarios: tres campos por nodo, padre, izquierdo y derecho.
2. Aridad acotada por m: un arreglo de m hijos por nodo.
3. Ramificación libre: hijo izquierdo y hermano derecho.

A ellas se suman dos que el curso ya viene usando: el arreglo de padres y las listas de adyacencia.

El árbol del ejemplo

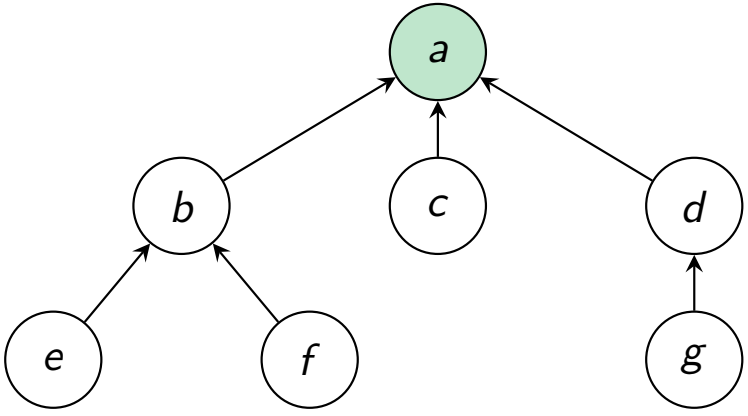


Siete nodos. La raíz a tiene tres hijos, así que la aridad es 3. Las hojas son c, e, f y g.

Pregunta

¿Cuántos campos por nodo hacen falta para guardarlo, si se quiere llegar a los hijos sin recorrer todo?

El arreglo de padres



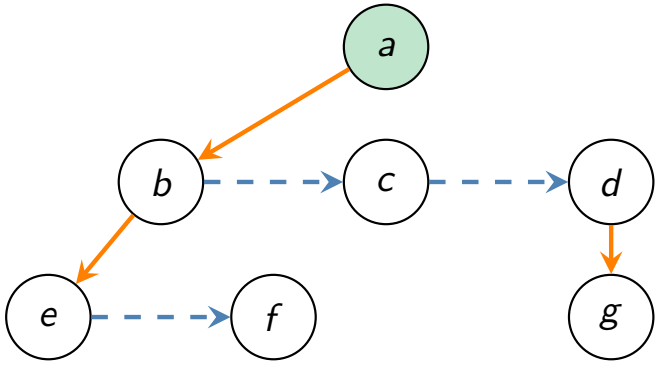
Un campo por nodo: `padre[x]`, con -1 en la raíz. Las flechas van del hijo al padre, al revés de como se dibuja el árbol.

Espacio $\Theta(n)$. El padre sale en $O(1)$ y subir a la raíz cuesta la profundidad.

Lo que no da

Los hijos de un nodo. Para saberlos hay que recorrer el arreglo entero buscando quién lo tiene de padre.

Hijo izquierdo y hermano derecho



Dos campos por nodo, pase lo que pase con la aridad:

- `hijoIzq[x]`, el primero de sus hijos.
- `hermanoDer[x]`, el siguiente hermano.

La flecha naranja va al primer hijo; las punteadas encadenan a los hermanos.

Por qué alcanza con dos

Los hijos de un nodo quedan en una lista enlazada por `hermanoDer`, y al primero se llega desde el padre. Cada nodo está en una sola de esas listas, así que el espacio es $\Theta(n)$ sin importar el grado.

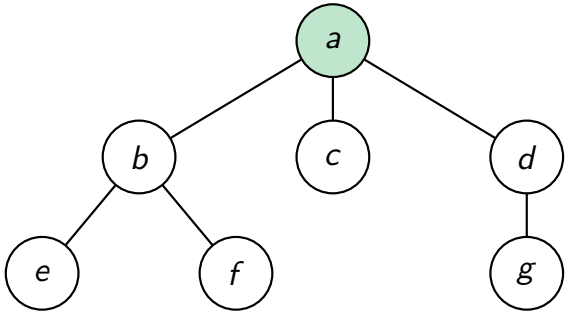
Hijo izquierdo y hermano derecho

Contra el arreglo de hijos

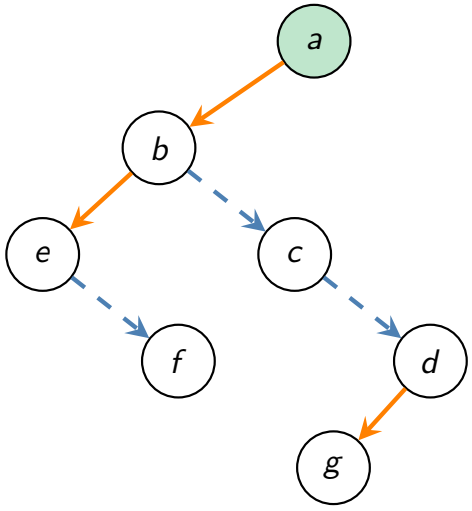
Reservar m casillas por nodo cuesta $\Theta(n \cdot m)$ y exige conocer la aridad. Con hijo izquierdo y hermano derecho no hace falta ninguna de las dos cosas.

Lo que se guardó es un árbol binario

el árbol de aridad 3



los mismos punteros, acomodados



Cada nodo guarda dos punteros, igual que en un árbol binario. Leyendo `hijoIzq` como hijo izquierdo y `hermanoDer` como hijo derecho, lo que hay en memoria es el árbol de la derecha.

Lo que cambia

Lo que era profundidad en el árbol de aridad 3 deja de serlo: *d* está a un paso de *a* a la izquierda y a tres a la derecha. El recorrido tiene que saber cuál de las dos flechas está siguiendo.

El código de las dos representaciones

```
14 padre      = [-1,  0,  0,  0,  1,  1,  3]
15 hijoIzq    = [ 1,  4, -1,  6, -1, -1, -1]
16 hermanoDer = [-1,  2,  3, -1,  5, -1, -1]
17
18 # el mismo arbol como listas anidadas: [valor,
   ↪  subarbol, ..., subarbol]
19 T = ['a', ['b', ['e'], ['f']], ['c'], ['d',
   ↪  ['g']]]
20
21 def caminoALaRaiz(x):
22   ans = [x]
23   while padre[x] != -1:
24     x = padre[x]
25     ans.append(x)
26   return ans
27
28 def profundidad(x):
29   ans = 0
30   while padre[x] != -1:
31     x = padre[x]
32     ans = ans + 1
33   return ans
```

```
35 def hijos(x):
36   ans = []
37   v = hijoIzq[x]
38   while v != -1:
39     ans.append(v)
40     v = hermanoDer[v]
41   return ans
42
43 def esHoja(x):
44   return hijoIzq[x] == -1
45
46 def altura(x):
47   ans = 0
48   if hijoIzq[x] != -1:
49     mayor = -1
50     for v in hijos(x):
51       h = altura(v)
52       if h > mayor: mayor = h
53     ans = 1 + mayor
54   return ans
55
56 def aridad():
57   ans = 0
58   for x in range(len(padre)):
59     if len(hijos(x)) > ans: ans = len(hijos(x))
60   return ans
```

`hijos` recorre la lista de hermanos, y ni ella ni `aridad` necesitan saber de antemano cuántos hay.

Las listas anidadas

El nodo se lleva adentro a sus subárboles

```
18 # el mismo arbol como listas anidadas: [valor, subarbol, ..., subarbol]
19 T = ['a', ['b', ['e'], ['f']], ['c'], ['d', ['g']]]
```

Con aridad libre, la lista arranca con el valor y sigue con los subárboles, tantos como haya.

```
62 def contarNodos(A):
63   ans = 1
64   for i in range(1, len(A)):
65     ans = ans + contarNodos(A[i])
66   return ans
```

Las listas anidadas

Dónde está el caso base

En una hoja la lista tiene un solo elemento, el ciclo no da ninguna vuelta y la cuenta queda en 1. No hace falta escribirlo aparte.

Para árboles binarios

La misma idea con las dos posiciones fijas, que es la gramática de la sección siguiente: $[\, n \ \langle \text{arbol} \rangle \ \langle \text{arbol} \rangle \,]$, con la lista vacía como árbol vacío.

Contenido

1 Antes de empezar

2 El árbol como grafo

3 El árbol como estructura de datos

4 La especificación recursiva

5 Los recorridos

6 La excentricidad

7 El diámetro de un árbol

8 Errores comunes

9 Ejercicios

10 Lo que sigue

11 Referencias

Un árbol es un nodo con árboles colgando

La gramática

$$\langle \text{arbol} \rangle ::= [\,] \mid [\, n \;\langle \text{arbol} \rangle \;\langle \text{arbol} \rangle\,]$$

Un árbol binario es la lista vacía, o una lista con el valor del nodo y sus dos subárboles. La regla se nombra a sí misma, y por eso describe árboles de cualquier tamaño con dos renglones.

Tres ejemplos

```
9  a = []
10 b = [5, [], []]
11 c = [6, [3, [], []], []]
```

El primero es el árbol vacío. El segundo tiene un nodo con valor 5 y dos subárboles vacíos. El tercero tiene raíz 6, un hijo izquierdo con valor 3 y ningún hijo derecho.

Un árbol es un nodo con árboles colgando

Pregunta

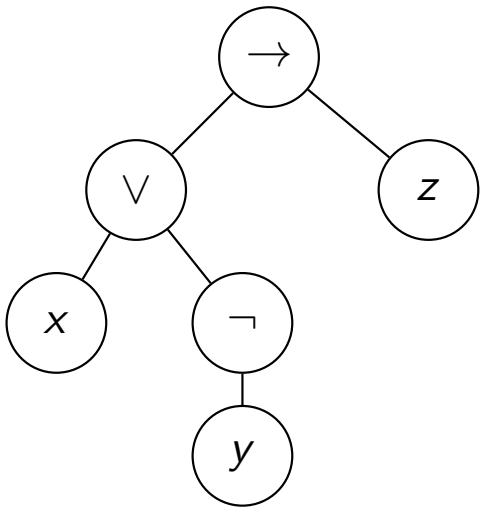
¿Cuántos corchetes tiene un árbol de n nodos?

La misma forma en otra parte

Una fórmula de la lógica

$$\varphi ::= \top \mid \bot \mid p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \to \varphi_2$$

Es la misma clase de definición: un caso base y unos casos que se arman con fórmulas más chicas. Por eso toda fórmula se dibuja como un árbol.



El árbol de $(x \vee \neg y) \to z$. Las hojas son las variables y los nodos internos son los conectivos. Evaluar la fórmula es recorrer el árbol de abajo hacia arriba.

Contar los nodos

```
22 def contarNodos(A):
23   if len(A) == 0: ans = 0
24   else:
25     ans = 1 + contarNodos(A[1]) + contarNodos(A[2])
26   return ans
```

Cómo se lee

La función sigue la gramática renglón por renglón. El caso base es la lista vacía, que no tiene nodos. El otro caso cuenta la raíz y pregunta por cada subárbol.

Contar los nodos

Cómo se desarma

$$\text{contarNodos}(T) = 1 + \text{contarNodos}(T_{izq}) + \text{contarNodos}(T_{der}),$$

y cada uno de esos dos vuelve a abrirse hasta llegar a listas vacías. Para el árbol de la clase da 10.

Lo que cuesta

Cada nodo se visita una vez y hace trabajo constante: $\Theta(n)$. La pila de llamadas llega a la altura del árbol.

La altura, con la misma forma

```
28 def altura(A):
29   if len(A) == 0: ans = -1
30   else:
31     ans = 1 + max(altura(A[1]), altura(A[2]))
32   return ans
```

Por qué el caso base vale -1

La altura se cuenta en aristas. Un árbol de un solo nodo tiene altura 0, así que el vacío tiene que valer -1 para que la suma dé 0.

El árbol en fila

Si cada nodo tiene un solo hijo, la altura es $n - 1$ y la pila de llamadas también. Con n grande, Python corta la recursión antes de terminar, y ahí hay que pasar a una pila explícita.

Contenido

1 Antes de empezar

2 El árbol como grafo

3 El árbol como estructura de datos

4 La especificación recursiva

5 Los recorridos

6 La excentricidad

7 El diámetro de un árbol

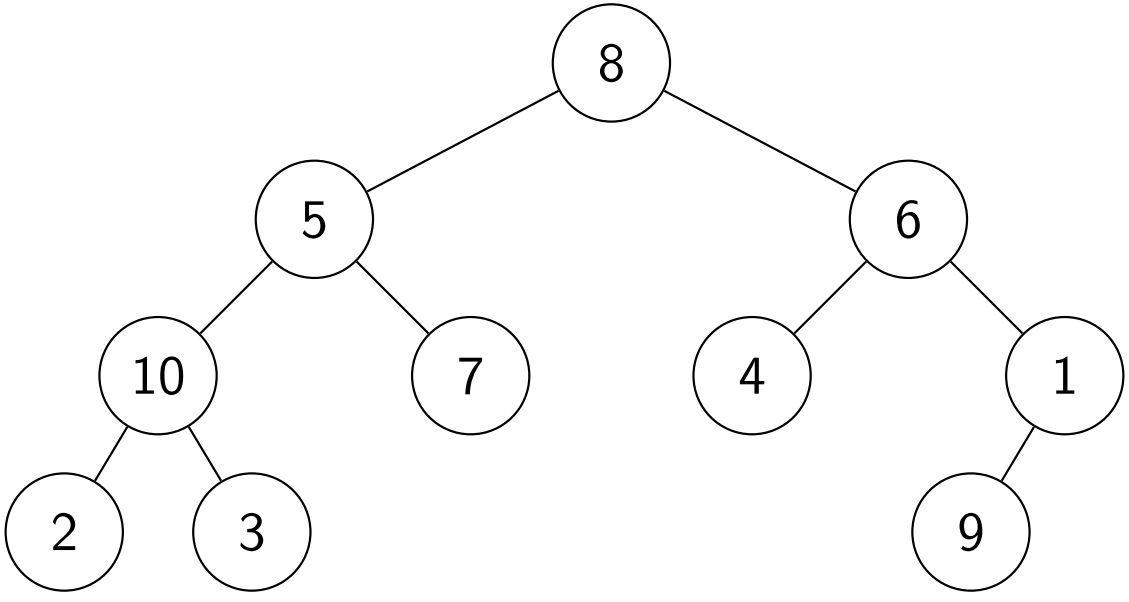
8 Errores comunes

9 Ejercicios

10 Lo que sigue

11 Referencias

Un árbol para recorrer



Diez nodos. El hijo izquierdo de 1 es 9, y 1 no tiene hijo derecho.

Pregunta

Si se visitan los nodos y se escriben sus valores, ¿en qué orden salen? Depende de cuándo se escriba la raíz: antes de bajar, después de bajar, o entre los dos hijos.

Los tres de profundidad

Dónde va la raíz

- Preorden: raíz, izquierdo, derecho.
- Posorden: izquierdo, derecho, raíz.
- Inorden: izquierdo, raíz, derecho.

Los tres visitan los mismos nodos y bajan por los mismos lados. Lo único que se mueve es el renglón que escribe la raíz.

```
19 def preorden(A, ans):
20   if len(A) > 0:
21     ans.append(A[0])
22     preorden(A[1], ans)
23     preorden(A[2], ans)
24   return ans
25
26 def posorden(A, ans):
27   if len(A) > 0:
28     posorden(A[1], ans)
```

Los tres de profundidad

```
29      posorden(A[2], ans)
30      ans.append(A[0])
31    return ans
32
33  def inorden(A, ans):
34    if len(A) > 0:
35      inorden(A[1], ans)
36      ans.append(A[0])
37      inorden(A[2], ans)
38    return ans
```

El de amplitud

```
40 def niveles(A):
41   ans, queue = [], deque()
42   if len(A) > 0: queue.append(A)
43   while len(queue) > 0:
44     B = queue.popleft()
45     ans.append(B[0])
46     if len(B[1]) > 0: queue.append(B[1])
47     if len(B[2]) > 0: queue.append(B[2])
48   return ans
```

Por niveles

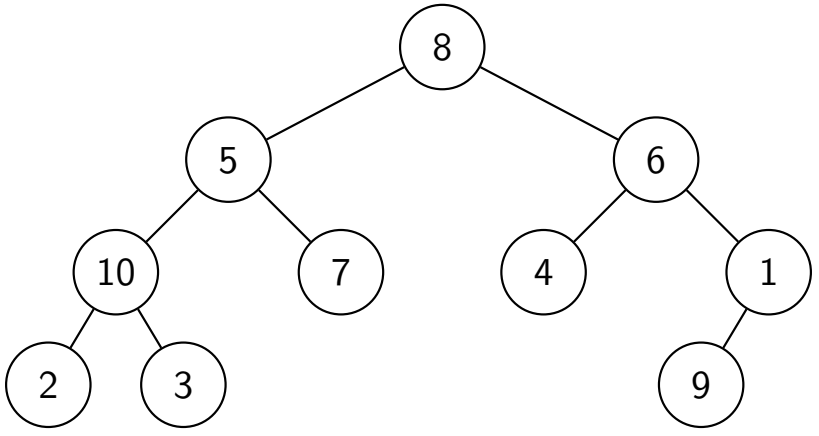
No hay recursión: una cola guarda los subárboles pendientes. Sale el primero que entró, se escribe su raíz y entran sus dos hijos. Es la búsqueda en amplitud, con el árbol como grafo.

El de amplitud

Lo que cambia frente a los otros tres

Los de profundidad terminan una rama antes de empezar la otra; este termina un nivel antes de bajar al siguiente.

Las cuatro salidas



Preorden	8 5 10 2 3 7 6 4 1 9
Posorden	2 3 10 7 5 4 9 1 6 8
Inorden	2 10 3 5 7 8 4 6 9 1
Por niveles	8 5 6 10 7 4 1 2 3 9

En preorden la raíz sale primero; en posorden, última. En inorden la raíz queda en la mitad, con los cinco nodos del subárbol izquierdo antes.

Para qué sirve cada uno

Preorden

Copiar o escribir el árbol: la raíz se necesita antes que los hijos. Es también el orden en que la búsqueda en profundidad descubre los nodos.

Posorden

Calcular algo del nodo que depende de sus hijos: el tamaño del subárbol, la altura, el valor de una fórmula. Cuando le toca a la raíz, los dos subárboles ya están resueltos.

Inorden

En un árbol binario de búsqueda deja los valores ordenados. Fuera de ese caso no tiene lectura especial, y en árboles de más de dos hijos ni siquiera está bien definido.

Para qué sirve cada uno

Por niveles

Responder qué hay a distancia 1, 2, 3 de la raíz, que es lo mismo que contesta la búsqueda en amplitud.

Contenido

1 Antes de empezar

2 El árbol como grafo

3 El árbol como estructura de datos

4 La especificación recursiva

5 Los recorridos

6 La excentricidad

7 El diámetro de un árbol

8 Errores comunes

9 Ejercicios

10 Lo que sigue

11 Referencias

Qué tan lejos queda lo más lejano

Definición (Excentricidad)

La excentricidad de un vértice v es la mayor de sus distancias:

$$\varepsilon(v) = \max_{u \in V} \delta(v, u).$$

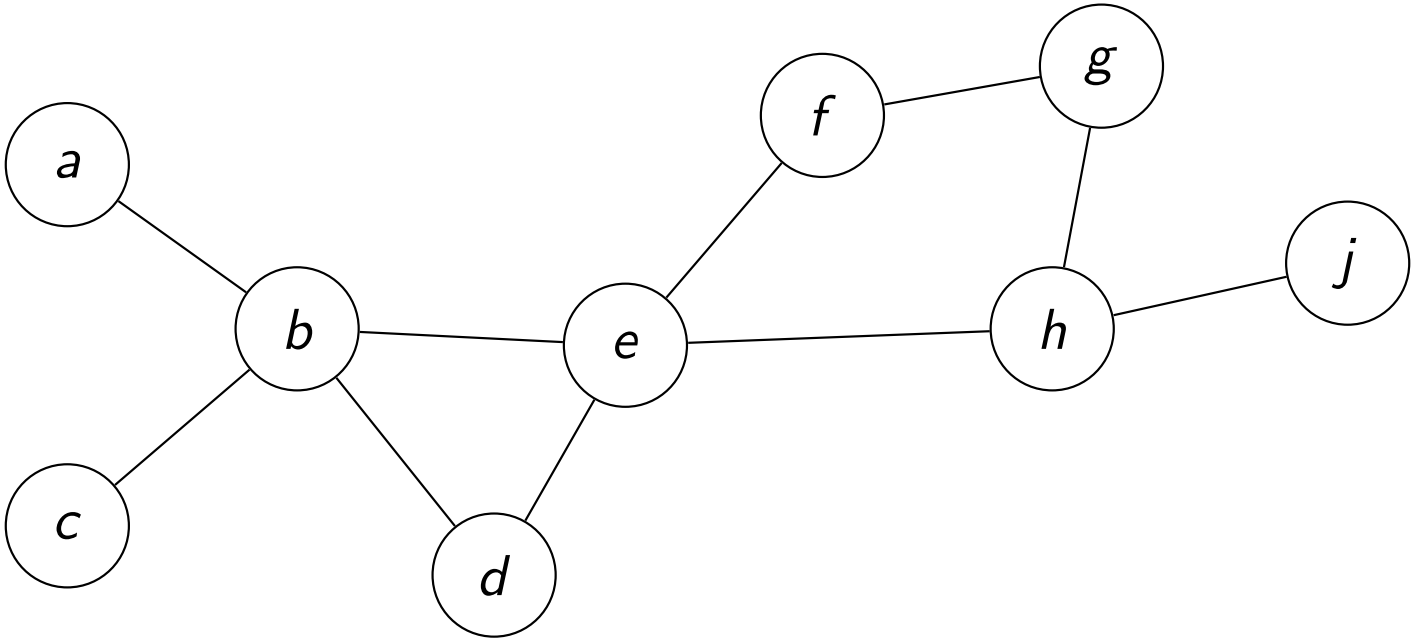
Qué distancia

La cantidad de aristas del camino más corto que une los dos vértices.
Aquí las aristas no pesan, así que la distancia se mide contando pasos.

Cómo leerla

Un vértice con excentricidad 2 alcanza a todos en dos pasos o menos.
Uno con excentricidad 4 tiene al menos un vértice a cuatro pasos.

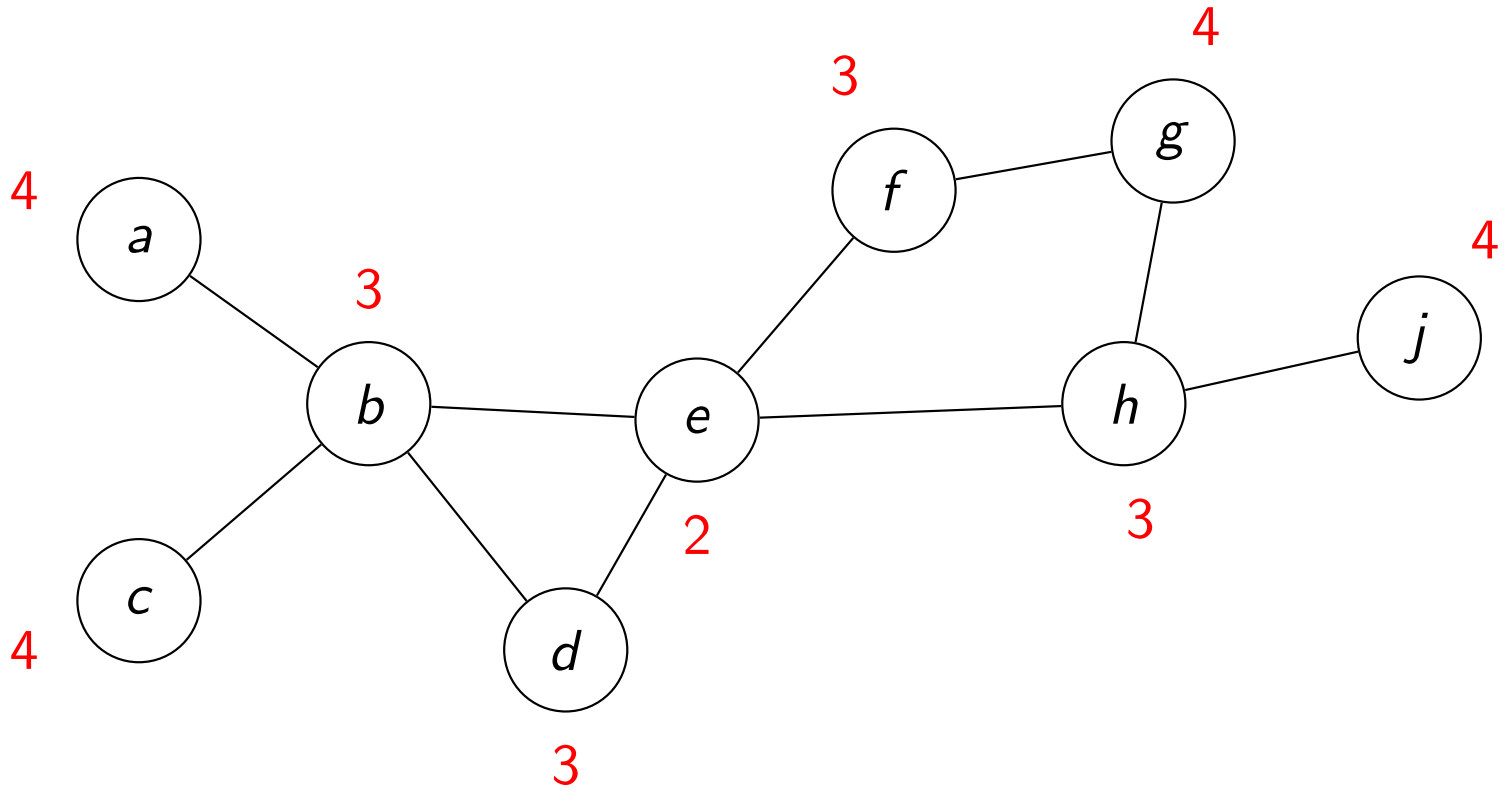
Las excentricidades de un grafo



Pregunta

¿Cuál de los nueve vértices tiene la menor excentricidad? ¿Y cuál la mayor?

Las excentricidades de un grafo



En rojo, la excentricidad de cada vértice. La menor es la de e, con 2; la mayor la tienen a, c, g y j, con 4. Tres de ellos tienen grado 1; g no, y aun así queda a cuatro pasos de a y de c.

Radio, diámetro y centro

Definición

- *Radio: la menor excentricidad,* $r = \min_v \varepsilon(v)$.
- *Diámetro: la mayor,* $d = \max_v \varepsilon(v)$.
- *Centro: los vértices cuya excentricidad es el radio.*

En el grafo del ejemplo

Radio 2, diámetro 4 y centro $\{e\}$.

Una cota que siempre vale

$r \leq d \leq 2r$. La primera desigualdad es inmediata; la segunda sale de pasar por un vértice del centro: dos vértices cualesquiera están a lo sumo a r de él.

Cómo se calculan

```
22 def amplitud(s):
23   dist = [-1 for _ in range(len(G))]
24   dist[s] = 0
25   queue = deque()
26   queue.append(s)
27   while len(queue) > 0:
28     v = queue.popleft()
29     for w in G[v]:
30       if dist[w] == -1:
31         dist[w] = dist[v] + 1
32         queue.append(w)
33   return dist
34
35 def excentricidad(v):
36   dist = amplitud(v)
37   ans = 0
38   for w in range(len(G)):
39     if dist[w] > ans: ans = dist[w]
40   return ans
```

Cómo se calculan

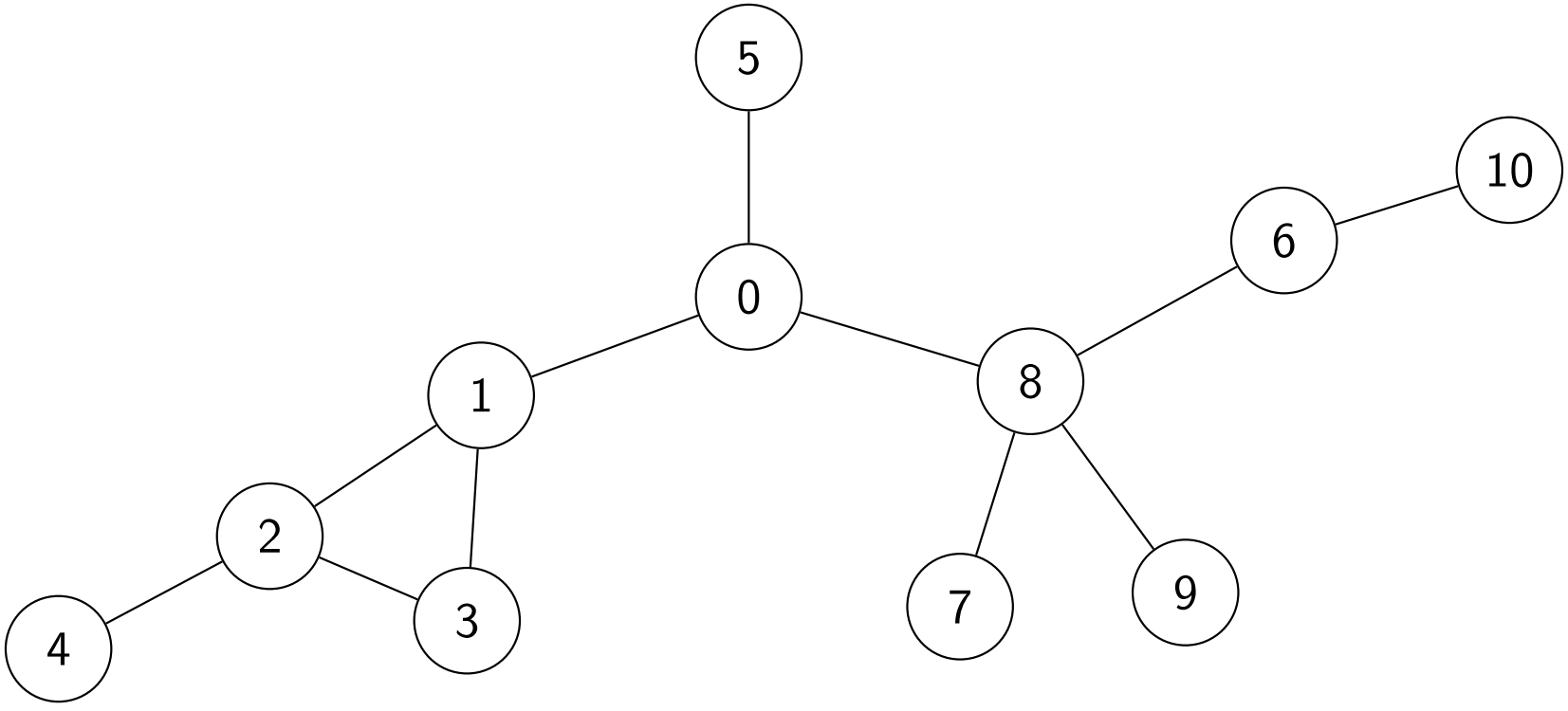
Una amplitud por vértice

La amplitud desde v da todas las distancias desde v en $O(|V| + |E|)$, y la mayor de ellas es $\varepsilon(v)$. Repetirlo desde cada vértice cuesta $O(|V| \cdot (|V| + |E|))$.

En un árbol se puede mejor

Un árbol tiene $n - 1$ aristas y un solo camino entre cada pareja, y con eso el diámetro sale con dos recorridos en vez de n. Es lo que viene.

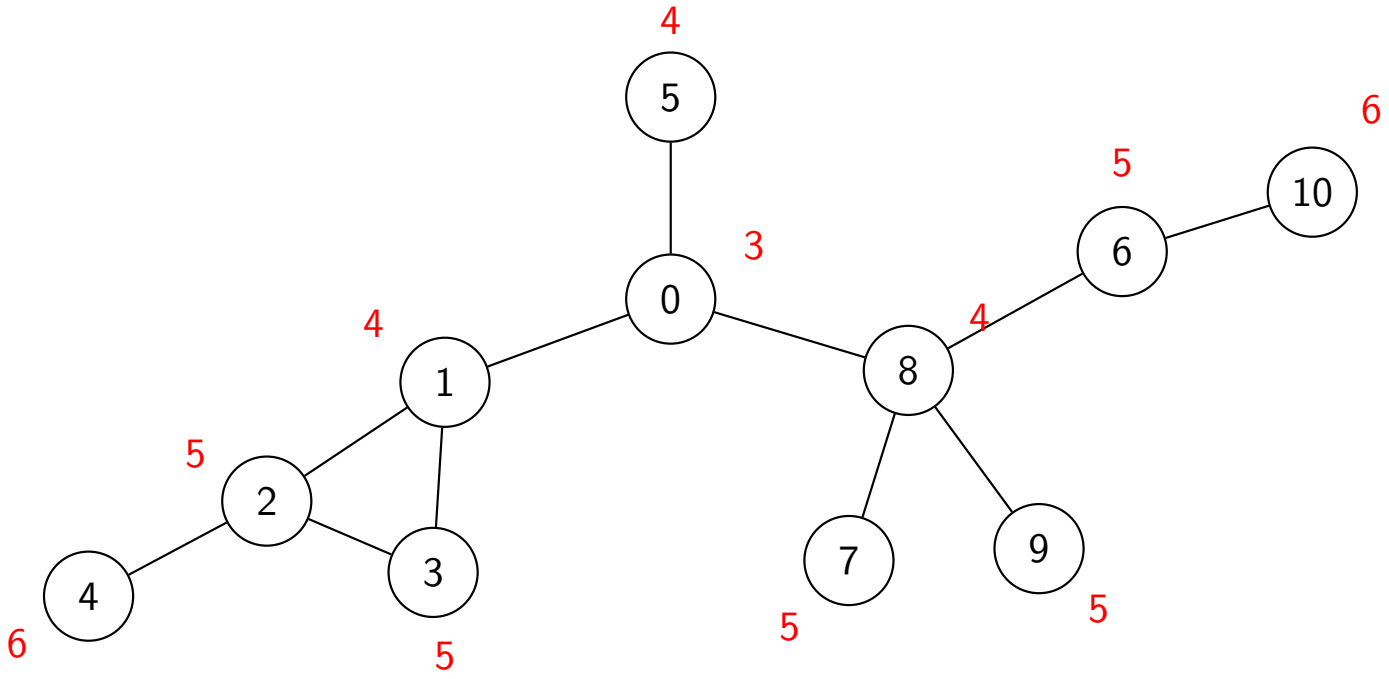
Otro grafo, once vértices



Pregunta

¿Cuánto valen $\delta(0,7)$, $\delta(1,10)$, $\delta(6,10)$, $\delta(4,7)$, $\delta(8,3)$ y $\delta(9,7)$?

Las distancias y las excentricidades



$\delta(0,7) = 2$, $\delta(1,10) = 4$, $\delta(6,10) = 1$, $\delta(4,7) = 5$, $\delta(8,3) = 3$ y $\delta(9,7) = 2$. En rojo la excentricidad de cada vértice: radio 3, diámetro 6 y centro $\{0\}$.

Cómo se lee

Una excentricidad baja dice que el vértice es central; una alta, que es periférico. El 4 y el 10, con 6, son los dos extremos del camino más largo.

Lo que cuesta calcularlos

Con todas las distancias a la mano

Si ya se tiene $\delta(u, v)$ para toda pareja, el radio y el diámetro salen recorriendo la matriz: $O(|V|^2)$.

Si no se tienen

En un grafo cualquiera, todas las distancias salen con Floyd-Warshall en $\Theta(|V|^3)$, o con una amplitud desde cada vértice en $O(|V| \cdot (|V| + |E|))$, que es mejor cuando el grafo es disperso.

En un árbol se puede mejor

Un árbol tiene $n - 1$ aristas y un solo camino entre cada pareja. Con eso, estas medidas no necesitan todas las distancias.

Contenido

1 Antes de empezar

2 El árbol como grafo

3 El árbol como estructura de datos

4 La especificación recursiva

5 Los recorridos

6 La excentricidad

7 El diámetro de un árbol

8 Errores comunes

9 Ejercicios

10 Lo que sigue

11 Referencias

Lo que tiene de particular un árbol

Cuatro cosas que ya se vieron

- Hay un único camino entre cualquier par de nodos.
- Cualquier nodo puede tomarse como la raíz.
- Una hoja es cualquier nodo que solo tiene un adyacente.
- El diámetro es la longitud de un camino que conecta dos hojas.

Por qué la última

Si un extremo del camino más largo no fuera hoja, tendría otro vecino por fuera del camino y el camino se podría alargar. Entonces los dos extremos son hojas.

Lo que tiene de particular un árbol

Pregunta

En el árbol de diez nodos de los recorridos, ¿cuál es el camino más largo y cuánto mide?

Buscar los dos extremos

Respuesta

El camino más largo es $\langle 2, 10, 5, 8, 6, 1, 9 \rangle$, de seis aristas, y sus dos extremos son hojas.

Pregunta

Se toma un vértice cualquiera y se busca el más lejano. ¿Ese vértice tiene algo de especial?

Lo que se sabe de los extremos

Los dos extremos de un camino más largo son hojas, y su longitud es el diámetro. La pregunta es si uno cualquiera de ellos se puede encontrar sin probarlos todos.

El teorema de las dos pasadas

Teorema (Diámetro de un árbol)

Sea $G = (V, E)$ un árbol y sean $u, v \in V$ tales que $\delta(u, v) = \varepsilon(v)$. Entonces $diam(G) = \varepsilon(u)$.

En palabras

Desde cualquier vértice v, el más lejano u es un extremo de un camino más largo del árbol. Y entonces la excentricidad de ese u ya es el diámetro.

El teorema de las dos pasadas

Por qué

Si u no fuera extremo de ningún camino más largo, el camino de v a u y un camino más largo se cruzan o no. Si se cruzan, cambiar de rama en el cruce da un camino más largo que termina en u; y si no se cruzan, el tramo que los une también lo da. En los dos casos se contradice que u sea el más lejano de v.

Dónde se usa que es un árbol

En que entre dos vértices hay un solo camino, que es lo que hace que esos cruces estén bien definidos. En un grafo con ciclos el teorema es falso.

El algoritmo

Diametro(G)

```
1  recorrer G desde un nodo cualquiera s y quedarse
   con el nodo mas lejano u
2  recorrer G desde u y quedarse con el nodo mas
   lejano w
3  devolver la distancia entre u y w
```

Dos recorridos y ya

Cada uno cuesta $O(|V| + |E|) = O(n)$ en un árbol, así que el total es $\Theta(n)$. Contra las n búsquedas que pedía el caso general, se pasa de $O(n^2)$ a $O(n)$.

El algoritmo

El primer nodo da igual

El teorema no pide nada de s. Se arranca en el 0 y el resultado es el mismo que arrancando en cualquier otro.

El código

```
12 distMax, nodoMax = None, None
13 G = None
14
15 def dfsAux(v, p, dist):
16   global distMax, nodoMax
17   dist += 1
18   if dist > distMax: distMax, nodoMax = dist, v
19   for w in  G[v]:
20     if w != p:
21       dfsAux(w, v, dist)
22
23 def diameter():
24   global distMax
25   distMax = 0
26
27   #se escoge arbitrariamente iniciar en el nodo 0
28   #el resultado seria el mismo asi se inicie en un nodo diferente
29   dfsAux(0, -1, -1)
30   dfsAux(nodoMax, -1, -1)
```

El código

```
31   return distMax
```

Cómo lleva la cuenta

`dfsAux` baja por el árbol sumando uno a `dist` y deja en `nodoMax` el nodo más lejano que vio. El parámetro `p` es de dónde viene, y la línea que lo compara es la que evita devolverse por la arista del padre.

Las dos llamadas

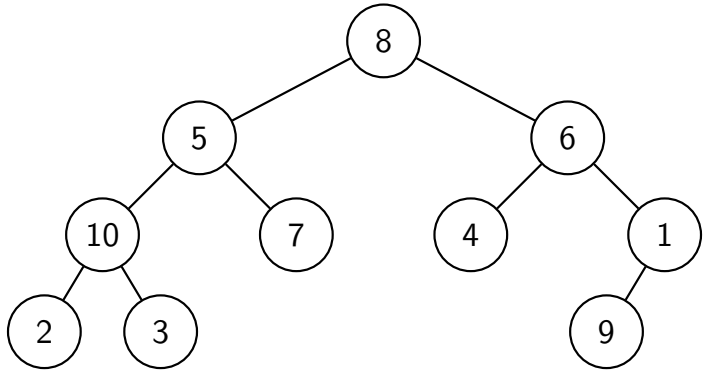
La primera sale del 0 y deja en `nodoMax` un extremo. La segunda sale de ahí. `distMax` no se reinicia entre las dos, y no hace falta: lo que mida la segunda nunca es menor que lo que midió la primera.

El código

El árbol de un solo nodo

Con $n = 1$ ningún `dist` supera el 0 inicial, `nodoMax` se queda sin valor y la segunda llamada falla. Si la entrada lo admite, se responde 0 antes de llamar.

Las dos pasadas, sobre el mismo árbol



Nodo	8	5	6	10	7	4	1	2	3	9
Desde 8	0	1	1	2	2	2	2	3	3	3
Desde 2	3	2	4	1	3	5	5	0	2	6

Desde 8 quedan tres a distancia 3, y sirve cualquiera: los tres son extremos de un camino más largo. Desde el 2, el más lejano es el 9 a distancia 6, y ese es el diámetro. El camino es $\langle 2, 10, 5, 8, 6, 1, 9 \rangle$, el mismo que encontró la versión recursiva.

De ahí sale el radio

El centro está en la mitad del camino más largo

Los vértices que quedan en la mitad de un camino más largo son los centros, y su excentricidad es el radio. De eso sale una cuenta directa:

$$rad(G) = \frac{diam(G) + 1}{2},$$

con la división entera.

En el árbol del ejemplo

$diam = 6$, así que $rad = 3$. Y en efecto $\varepsilon(8) = 3$, que es la menor de las diez: el centro es $\{8\}$, el vértice que queda en la mitad del camino de 2 a 9.

De ahí sale el radio

Cuántos centros

Si el diámetro es par hay un solo centro, y si es impar hay dos, que son los dos de la mitad del camino. Nunca hay más.

La otra forma: con la vista recursiva

Cuando el árbol ya viene con raíz

Las dos pasadas no piden raíz, pero sí dos recorridos sobre listas de adyacencia. Si el árbol ya llega con su raíz y con los hijos de cada nodo, una sola pasada alcanza, y de paso deja la altura de todos.

Los dos casos

Con la raíz escogida, el camino más largo del árbol o pasa por la raíz, o no la toca y entonces está entero dentro de uno de los dos subárboles. No hay más opciones.

- Si pasa por la raíz, mide la altura del subárbol izquierdo más la del derecho, más dos aristas.
- Si no, es el diámetro de uno de los dos subárboles.

La otra forma: con la vista recursiva

Dividir, conquistar y combinar

Cada subárbol devuelve dos cosas: su altura y su diámetro. El nodo las combina con un máximo de tres candidatos.

El código

```
13 def diameter(v):
14   global heights
15   ans, h1, h2, diamLeft, diamRight = None, 0, 0, 0, 0
16   if len(G[v]) == 0:
17     heights[v] = 0
18     ans = 0
19   else:
20     left = G[v][0]
21     diamLeft = diameter(left)
22     heights[v] = heights[left] + 1
23     h1 = heights[left] + 1
24
25     if len(G[v])  > 1:
26       right = G[v][1]
27       diamRight = diameter(right)
28       heights[v] = max(heights[v], heights[right] + 1)
29       h2 = heights[right] + 1
30     ans = max(h1 + h2, diamLeft, diamRight)
31   return ans
```

El código

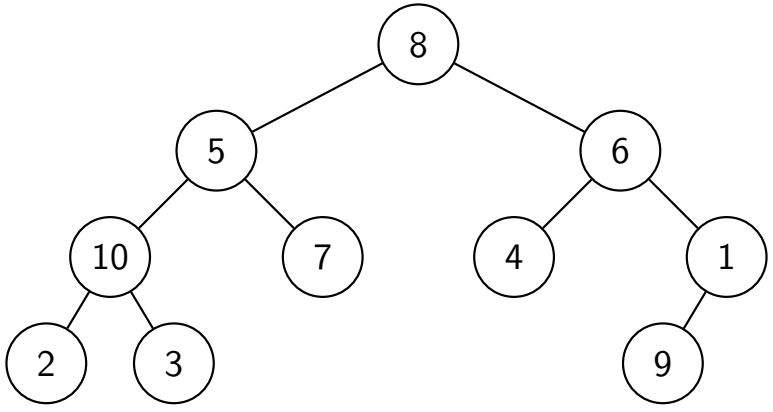
Cómo llega el árbol aquí

Como listas de hijos y con la raíz conocida: `G[v]` trae los hijos de `v` y nada más, sin la arista de vuelta al padre.

Las dos salidas

`heights` queda con la altura de cada nodo, y lo que devuelve la función es el diámetro. La línea 30 escoge entre el camino que pasa por el nodo y los diámetros de los dos lados.

El ejemplo, nodo por nodo



Nodo	2	3	7	4	9	10	1	5	6
Altura	0	0	0	0	0	1	1	2	2

En la raíz 8 la altura es 3, y el candidato que pasa por ella mide $(2+1)+(2+1)=6$, que le gana a los diámetros de los dos lados. El diámetro es 6.

Lo que cuesta

Una visita por nodo

Cada nodo se resuelve con lo que devolvieron sus hijos y trabajo constante: $\Theta(n)$ en tiempo. La pila de llamadas llega a la altura del árbol.

Lo que pide

Este algoritmo necesita la raíz y las aristas en un solo sentido. Si el árbol llega sin raíz, se escoge una cualquiera, que para esto da lo mismo.

Contenido

1 Antes de empezar

2 El árbol como grafo

3 El árbol como estructura de datos

4 La especificación recursiva

5 Los recorridos

6 La excentricidad

7 El diámetro de un árbol

8 Errores comunes

9 Ejercicios

10 Lo que sigue

11 Referencias

Errores comunes

Confundir el grado con la cantidad de hijos

En el árbol con raíz, un nodo interno de grado 3 tiene dos hijos y un padre. La raíz es la única donde grado y cantidad de hijos coinciden.

Recorrer el árbol como grafo sin recordar de dónde se viene

Con listas de adyacencia, el vecino por el que se llegó vuelve a aparecer y el recorrido se devuelve. Se pasa el padre en la llamada y se salta.

Creer que la altura no cambia con la raíz

El mismo árbol colgado de dos nodos distintos da alturas distintas. Lo que no cambia son las aristas ni las distancias entre vértices.

Errores comunes

Pedir inorden en un árbol que no es binario

Con tres hijos no hay un “entre los dos” que tenga sentido. Preorden y posorden sí se extienden sin problema.

La recursión que no cabe

Un árbol en fila de cien mil nodos desborda la pila de Python. La versión con pila explícita es la que se envía al juez.

Contenido

1 Antes de empezar

2 El árbol como grafo

3 El árbol como estructura de datos

4 La especificación recursiva

5 Los recorridos

6 La excentricidad

7 El diámetro de un árbol

8 Errores comunes

9 Ejercicios

10 Lo que sigue

11 Referencias

Ejercicio 1: reconocer árboles

Enunciado

De cada grafo diga si es árbol y por qué, usando la caracterización más barata de comprobar en cada caso.

1. Conexo, $n = 10$, $|E| = 9$.
2. Acíclico, $n = 10$, $|E| = 8$.
3. Conexo, $n = 10$, $|E| = 10$.
4. Acíclico, $n = 10$, $|E| = 9$.

Y además

En el caso 3, ¿cuántos ciclos simples hay como mínimo?

Ejercicio 2: leer un recorrido

Enunciado

De un árbol binario se sabe que su preorden es 7 3 1 5 4 9 8 11 y su inorden es 1 3 4 5 7 8 9 11. Dibuje el árbol y escriba su posorden.

Pista

El primero del preorden es la raíz. En el inorden, lo que queda a su izquierda es el subárbol izquierdo y lo que queda a la derecha, el derecho. Se repite en cada pedazo.

Y además

¿Alcanzan el posorden y el inorden para reconstruir el árbol? ¿Y el preorden con el posorden?

Ejercicio 3: contar hojas

Enunciado

Escriba `contarHojas(A)` sobre la definición recursiva, con un solo `return` y sin librerías. Una hoja es un nodo cuyos dos subárboles son vacíos.

Y después

Escriba `sumaProfundidades(A)`, que devuelve la suma de las profundidades de todos los nodos. ¿Qué parámetro hay que agregar?

Ejercicio 4: excentricidades a mano

Enunciado

Sobre el árbol de seis vértices de la clase, calcule la excentricidad de los seis y diga cuáles son el radio, el diámetro y el centro.

Y además

Compruebe que $r \leq d \leq 2r$ y diga cuántos vértices tiene el centro.
¿Puede un árbol tener tres vértices en el centro?

Ejercicio 5: un problema de juez

UVa 10459, The Tree Root

Dado un árbol, hay que decir cuáles vértices dan la menor altura al colgarlo de ellos y cuáles la mayor. Los primeros son el centro y los segundos son las hojas más lejanas.

Enunciado y envío: `https://vjudge.net/problem/UVA-10459`

Consejo

Colgar el árbol de v y mirar su altura es lo mismo que calcular $\varepsilon(v)$. Con n hasta unos pocos miles, una amplitud desde cada vértice alcanza; con más, hay que esperar a lo que viene.

Contenido

1 Antes de empezar

2 El árbol como grafo

3 El árbol como estructura de datos

4 La especificación recursiva

5 Los recorridos

6 La excentricidad

7 El diámetro de un árbol

8 Errores comunes

9 Ejercicios

10 Lo que sigue

11 Referencias

Lo que sigue

El diámetro sin raíz y el centro

Hay otro camino al mismo número, que no pide escoger raíz: desde un vértice cualquiera se busca el más lejano, y desde ese se vuelve a buscar. La distancia entre los dos es el diámetro. De ahí sale el radio, que es $(diam + 1)/2$, y el centro, que se obtiene quitando las hojas una y otra vez hasta que queden uno o dos vértices.

Para investigar

- ¿Por qué el vértice más lejano a uno cualquiera es siempre un extremo de un camino más largo del árbol?
- Si se quitan todas las hojas de un árbol y se repite, ¿qué queda al final y cuántas vueltas toma?

Contenido

1 Antes de empezar

2 El árbol como grafo

3 El árbol como estructura de datos

4 La especificación recursiva

5 Los recorridos

6 La excentricidad

7 El diámetro de un árbol

8 Errores comunes

9 Ejercicios

10 Lo que sigue

11 Referencias

Referencias



Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. *Introduction to Algorithms*, 3rd ed. MIT Press, 2009. Apéndice B.5, páginas 1173 a 1177, y Sección 10.4, representación de árboles con raíz.



Halim, S., Halim, F., Effendy, S. *Competitive Programming 4*. Lulu, 2020. Capítulo 4: propiedades de los árboles, diámetro y centro.



van Steen, M. *Graph Theory and Complex Networks*. Maarten van Steen, 2010. Capítulo 5: árboles, excentricidad y centro.



UVa Online Judge, problema 10459. `https://vjudge.net`