Tarea 4

Saul Ivan Rivas Vega

Diseño y análisis de algoritmos

Equipo Completo:
Yadira Fleitas Toranzo
Diego de Jesús Isla Lopez
Saul Ivan Rivas Vega

27 de septiembre de 2019

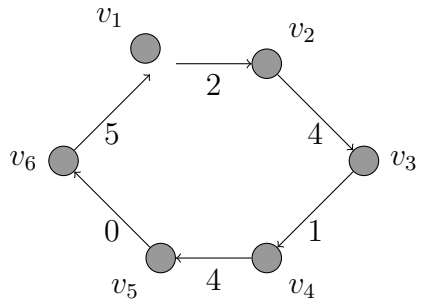
1. Ejercicio 1.

1.1. Da un ejemplo de una familia de digráficas de n vértices con pesos no negativos tal que admita una ejecución del algoritmo de Dijkstra en la cual todos los nodos actualizan a su padre y su estimación de distancia $d(v)$ una sola vez; es decir, la primera oferta de camino que reciben, es la del camino mas corto. Demuestra que tu ejemplo cumple lo que se pide.

Sea F la familia de digráficas que cumpla las siguientes condiciones para cada miembro G de ella:

- La gráfica G tiene un conjunto de aristas ponderadas E tal que $|E| = n$ donde cada arista tiene un peso mayor o igual a cero y un conjunto de vértices V tal que $|V| = n$.
- G es conexa.
- Todo $v \in V$ tiene $inDegree(v) = 1$, es decir que tienen exactamente una sola arista de entrada.
- Todo $v \in V$ tiene $outDegree(v) = 1$, es decir que tienen exactamente una sola arista de salida.
- Solo hay un ciclo y todo $v \in V$ esta en el ciclo.

Un miembro de esta familia con $n = 6$ es:



Demostración Por contradicción.
Supongamos que durante una ejecución el algoritmo de Dijkstra al empezar en cualquier nodo $s \in G$, existe un nodo v_i tal que $v_i \neq s$ y además actualizó su estimación de distancia $d(v_i)$ mas de una vez.
Entonces el algoritmo hizo una primera estimación de distancia $d(v_i)_1$ siguiendo un camino de nodos p desde s hasta v_i. Ahora para realizar una estimación adicional $d(v_i)_2$ tal que $d(v_i)_2 < d(v_i)_1$ debió encontrar un segundo camino p' para llegar a v_i.
Sin embargo, siendo G miembro de la familia F se cumple que todo $v \in G$ tiene una sola arista de salida y una de entrada, y solo hay un ciclo donde todo $v \in G$ esta en el. Le sigue que, para llegar a v_i desde s, se puede:

1. Seguir el ciclo hasta llegar a v_i.

2. Completar el ciclo z veces ($z >= 1$) y volver a llegar a v_i.

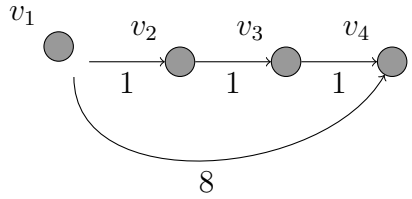
El camino resultante de al usar el segundo método será siempre mayor o igual al primero puesto que los pesos son mayores o iguales a 0 manteniendo o incrementando el camino del primer método.
Por lo tanto existe un solo camino mínimo entre cualquier par de nodos en G. Entonces no existe un segundo camino p' tal que al llegar a v_i su estimación de distancia cumpla que $d(v_i)_2 < d(v_i)_1$ y en la ejecución no hubo una actualización adicional a la primera, pero esto es una contradicción a nuestra suposición inicial. Finalmente, en cualquier ejecución del algoritmo empezando en cualquier nodo $s \in G$, todos los nodos actualizan a su padre y su estimación de distancia $d(v_i)$ una sola vez.

1.2. Da un ejemplo de una familia de digráficas con pesos no negativos tal que cualquier ejecución del algoritmo de Dijkstra requiera que algún vértice actualice su padre y su estimación de distancia dos veces. Demuestra que tu ejemplo cumple lo que se pide.

Sea F la familia de digráficas que cumpla las siguientes condiciones para cada miembro G de ella:

1. Tiene un nodo inicial s el cual no tiene aristas de entrada.
2. s tiene una arista de salida con peso igual a 1.
3. Tiene un nodo final v_f el cual no tiene aristas de salida.
4. v_f tiene una arista de entrada con peso igual a 1.
5. Para todo $v \in G$ tal que $v \neq s$ y $v \neq v_f$, se cumple:
   a) v tiene una sola arista de entrada con peso igual a 1.
   b) v tiene una sola arista de salida con peso igual a 1.
6. Desde s se puede llegar a todos los otros vértices.
7. Hay una arista adicional desde s a v_f con peso $2n$.
8. s tiene exactamente 2 aristas de salida.
9. v_f tiene exactamente 2 aristas de entrada.

Un miembro de esta familia con $n = 4$, donde $s = v_1$ y $v_f = v_4$ es:

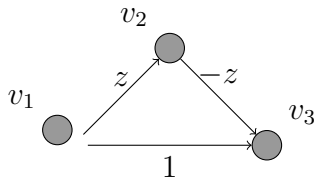


Demostración Propongamos al vértice v_f como aquel nodo que actualice su padre y su estimación de distancia dos veces al comenzar desde el nodo s. La primer estimación de distancia $d(v_f)_1$se realiza al ejecutar el algoritmo iniciando en s, pues v_f es vecino directo por la propiedad 7. $d(v_f)_1$ vale $2n$. Ahora para una segunda estimación se puede seguir el camino de la otra arista de peso 1 de s, dicho camino llegará a v_f por la propiedad 6.
Este camino tendrá como peso total la suma de los pesos de las aristas, cada arista en ese camino tiene peso 1, y como pasa por todos los nodos y todos menos v_f tienen una arista de salida de peso 1, significa que hay $n-1$ aristas. Posteriormente tenemos que esta segunda estimación $d(v_f)_2$ vale $n-1$. Finalmente como $d(v_f)_2 < d(v_f)_1$ habrá forzosamente una segunda actualización.

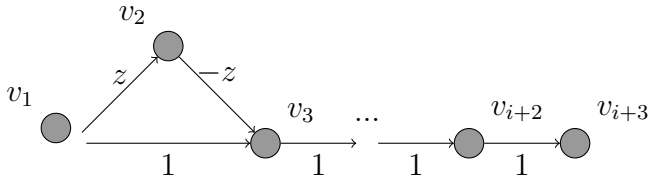
2. Ejercicio 2.

2.1. De acuerdo con el algoritmo de Dijkstra que revisamos en clase, para cada $n \in \mathbb{N}$ con $n > 2$, presenta una familia de gráficas de n vértices con pesos tanto positivos como negativos y sin ciclos dirigidos de longitud negativa para la cual se cumple que si G es una gráfica de la familia, el algoritmo de Dijkstra falla en encontrar el camino más corto entre s y algún otro vértice de G. Demuestra que de hecho existe tal vértice.

Sea F la familia de digráficas que sigue el siguiente proceso de construcción: Para G_0 se tiene la siguiente gráfica:



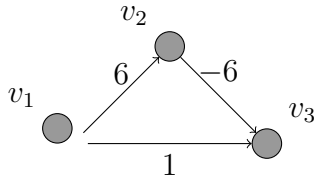
Donde $z = 2n$, en G_0 $z = 6$.
Ahora para cualquier otro G_i con $i > 0$ se agregará una arista del último nodo de G_{i-1}, el cual es v_{i+2}, a un nuevo nodo v_{i+3} con peso 1:



Demostración La ejecución de Dijkstra empieza en v_1 y propongamos que donde falla en encontrar el camino mínimo es para el nodo v_3.
En la primera ocasión donde se estiman distancias a partir de v_1 se marca como explorados a v_2 y a v_3. La estimaciones son: $d(v_2) = z$ y $d(v_3) = 1$. Esas son las primeras estimaciones para cualquier ejecución del algoritmo con nodo inicial en v_1.
Ahora como $z = 2n$ y $2n > 1$ el algoritmo seguirá explorando ahora con v_3. Solo se consideraría ir por v_2 cuando la estimación de distancia para algún nodo v_i cumpla que $d(v_i) > 2n$, sin embargo como las aristas tienen peso 1, y no hay mas de $n - 3$ aristas a partir de v_3, nunca se considerará explorar a partir de v_2.
Por lo tanto al explorar todos los nodos se terminará la ejecución del algoritmo y la estimación de distancia para v_3 es $d(v_3) = 1$, sin embargo al tomar el camino de v_1 a v_2 y luego a v_3 se tendría una distancia de 0, la cual es menor a 1. Finalmente para cualquier gráfica G miembro de la familia al ejecutar el algoritmo de Dijkstra empezando en v_1, el algoritmo fallará en encontrar la distancia mínima para el nodo v_3.

2.2. Toma una gráfica de la familia que propones y muestra que Bellman-Ford si encuentra el camino que Dijkstra no.

Tomemos a G_0:



```
1 Inicializar d(v) = infinito, para toda v ∈ G;
2 Asignar d(s) = 0;
3 for i desde 1 hasta (n-1) do
4 |   foreach Arista (u, v, peso) ∈ E do
5 |   |   if d(u) + peso < d(v) then
6 |   |   |   d(v) := d(u) + peso;
7 |   |   end
8 |   end
9 end
```

Algorithm 1: Bellman-Ford.

Demostración Sea v_1 el nodo s para el algoritmo.
Al inicializar las distancias como en las lineas 1 y 2 tenemos:

- $d(v_1) = 0$.
- $d(v_2) = infinito$.
- $d(v_3) = infinito$.

En la primera ejecución del ciclo en la linea 3 se prueban las siguientes aristas con las correspondientes evaluaciones:

- Arista$(v_1, v_2, 6)$.
  - $d(v_1) + 6 < d(v_2)$.
  - $0 + 6 < infinito$.
  - true. Asignamos $d(v_2) := 6$
- La arista $(v_1, v_3, 1)$ pudo haber sido evaluada en dos casos:
  1. Cuando $d(v_3) = infinito - 6$, es decir evaluamos primero a la arista $(v_2, v_3, -6)$.
     - $d(v_1) + 1 < d(v_3)$.
     - $1 < infinito - 6$.
     - true. Asignamos $d(v_3) := 1$
  2. Cuando $d(v_3) = infinito$, es decir todavía no evaluamos a la arista $(v_2, v_3, -6)$.
     - $d(v_1) + 1 < d(v_3)$.

- $0 + 1 < infinito$.
- true. Asignamos $d(v_3) := 1$

- La arista $(v_2, v_3, -6)$ pudo haber sido evaluada en los mismos dos casos:

  1. Cuando $d(v_2) = infinito$, es decir todavía no evaluamos a la arista $(v_1, v_2, 6)$.
     - $d(v_2) + (-6) < d(v_3)$.
     - $infinito - 6 < infinito$.
     - true. Asignamos $d(v_3) := infinito - 6$
  2. Cuando $d(v_2) = 6$, es decir evaluamos primero a la arista $(v_1, v_2, 6)$.
     - $d(v_2) + (-6) < d(v_3)$.
     - $6 - 6 < infinito$.
     - true. Asignamos $d(v_3) := 0$

De haber sido el caso número 2, la distancia ya es correcta para v_3 siendo $d(v_3) = 0$, sin embargo esa distancia se alcanza en la segunda iteración para el caso número 1:

- La arista $(v_2, v_3, -6)$ tomando el caso numero 1 de la iteración anterior se realiza la siguiente evaluación:
  - $d(v_2) + (-6) < d(v_3)$.
  - $6 - 6 < 1$.
  - true. Asignamos $d(v) := 0$

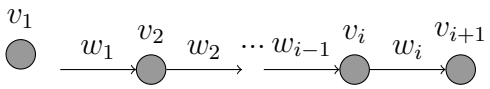
Finalmente se asegura que Bellman-Ford llega a la distancia correcta de v_3 siendo $d(v_3) = 1$, a diferencia de Dijkstra que reportaría erróneamente $d(v_3) = 1$ como se muestra en la demostración del ejercicio 2.1.

3. Ejercicio 3

3.1. De acuerdo con el algoritmo de Dijkstra que revisamos en clase, para cada $n \in \mathbb{N}$ con $n > 2$, presenta una familia de gráfircas de n vértices con pesos positivos y negativos y sin ciclos dirigidos de longitud negativa para la cual se cumple que si G es una gráfica de la familia, el algoritmo de Dijkstra no falla en encontrar el camino más corto entre dos vértices de G. Prueba que en efecto el algoritmo encuentra los caminos más cortos.

Para G_0 se tiene la siguiente gráfica:

Es decir un solo nodo v_1. Ahora para cualquier otro G_i con $i > 0$ se agregará un nuevo nodo v_{i+1} y una arista de v_i a v_{i+1} con peso w_i positivo, negativo o igual a 0.



Demostración En este caso cualquier gráfica G_i miembro de la familia tendrá i aristas así como $i + 1$ vértices, lo que resulta en que solo exista un camino a seguir estando en cualquier vértice $v \in G_i$. En cualquier ejecución de Dijkstra tomando cualquier vértice v como vértice inicial s, se recorrerá ese único camino que empieza en s. Para cualquier vértice en el conjunto de vértices que es posible explorar desde s, se encontrará el camino mínimo pues es único y se recorren todos los vértices sin explorar que son alcanzables desde s. Finalmente sin importar que se tengan pesos positivos, negativos o iguales a 0, si G es miembro de la familia descrita, el algoritmo de Dijkstra no falla en encontrar el camino más corto entre dos vértices de G.

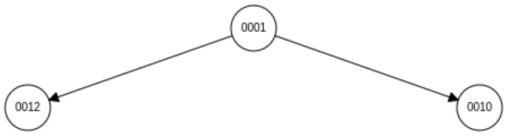
4. Ejercicio 4.

4.1. Considera la estructura de datos de cola de prioridad, implementada con heaps binarios, y el arreglo $A = (10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 7, 4, 11, 13, 2)$. Explica cómo construyes una cola de prioridad que tenga los elementos de A (tienes que dar la construcción del árbol binario); no puedes ordenar el arreglo de entrada antes de crear la cola. Escribe explícitamente la cola resultante.

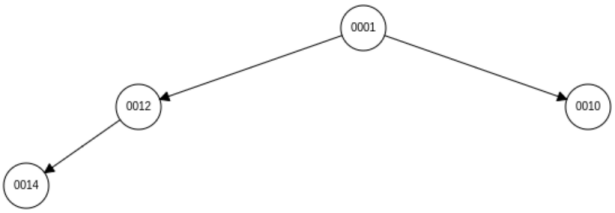
La cola de prioridad se construye agregando cada elemento como la siguiente hoja en el árbol binario, sin embargo en caso que sea menor a su padre cambiará lugar con su padre hasta que su padre no sea mayor o termine siendo la raíz.

(a) Elemento inicial 10.

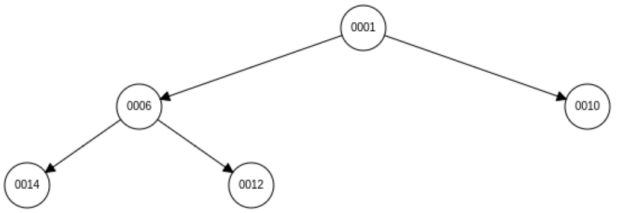
(b) Se agrega 12 y como es mayor solo se agrega como hoja.



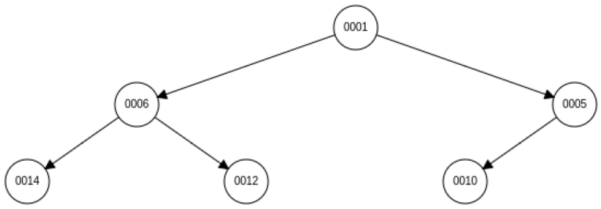
(c) 1 es menor que todos los elementos así que se agrega como hoja, pero sube hasta la raíz.



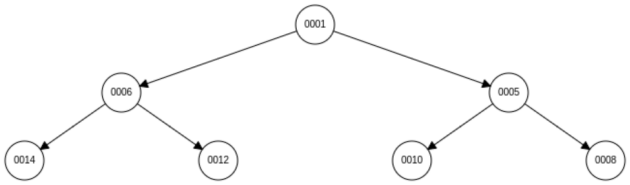
(d) 14 es mayor que todos los otros elementos, se agrega como hoja.



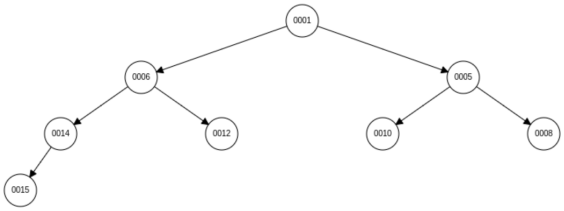
(e) 6 se agrega como hoja pero es menor a 12, así que sube.



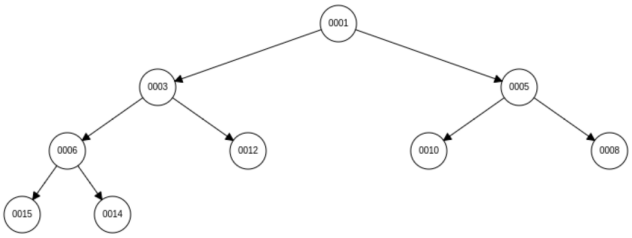
(f) Se agrega 5 como hoja pero es menor a 10 así que sube.



(g) Se agrega 8 como hoja y como es mayor a 5 se queda así.

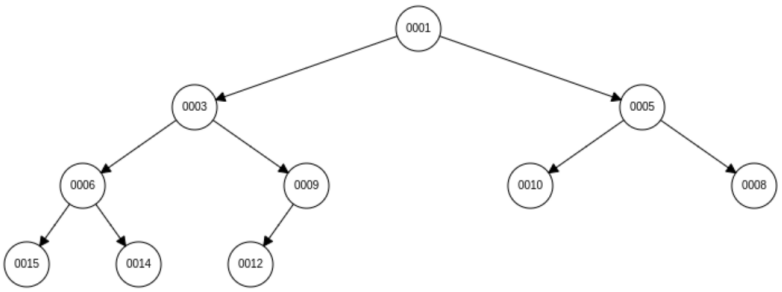


(h) Se agrega 15 como hoja y como es el mayor, se queda así.

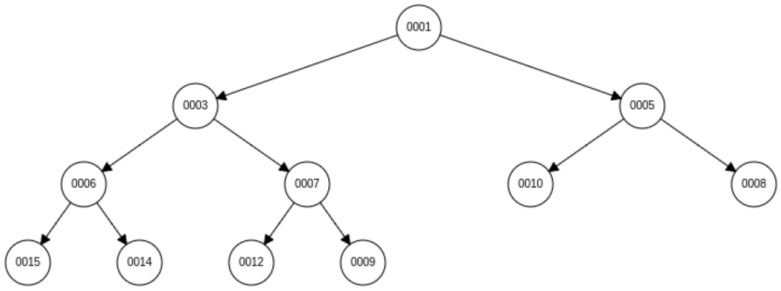


(i) Se agrega 3 como hoja y como es menor a 14 y a 6, sube dos niveles.

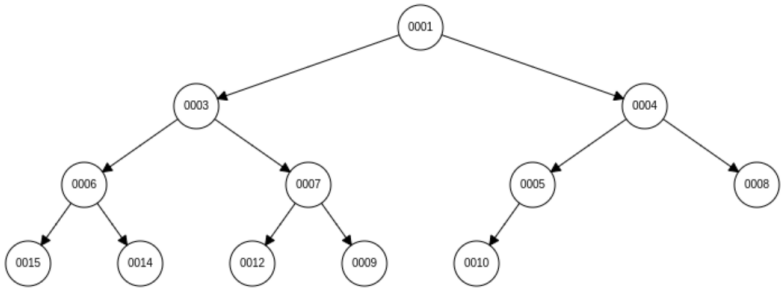
Figura 1: Árboles binarios en las primeras 9 inserciones.



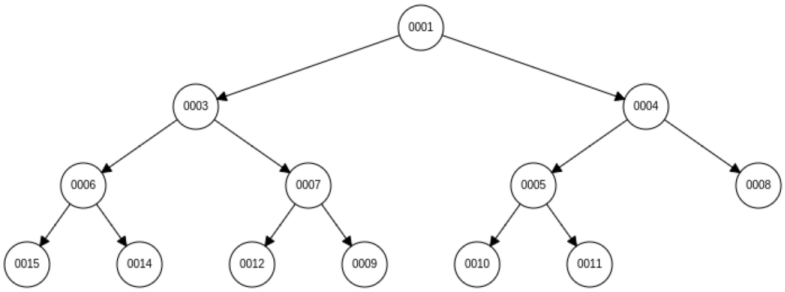
(a) Se agrega 9 como hoja y como es menor a 12, sube.



(b) Se agrega 7 como hoja y como es menor a 9, sube.

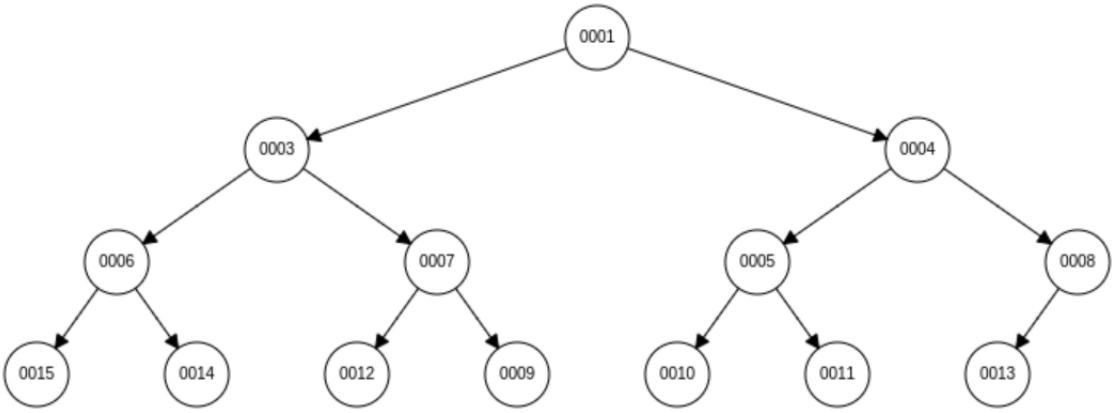


(c) Se agrega 4 como hoja y como es menor a 10 y 5, sube dos niveles.

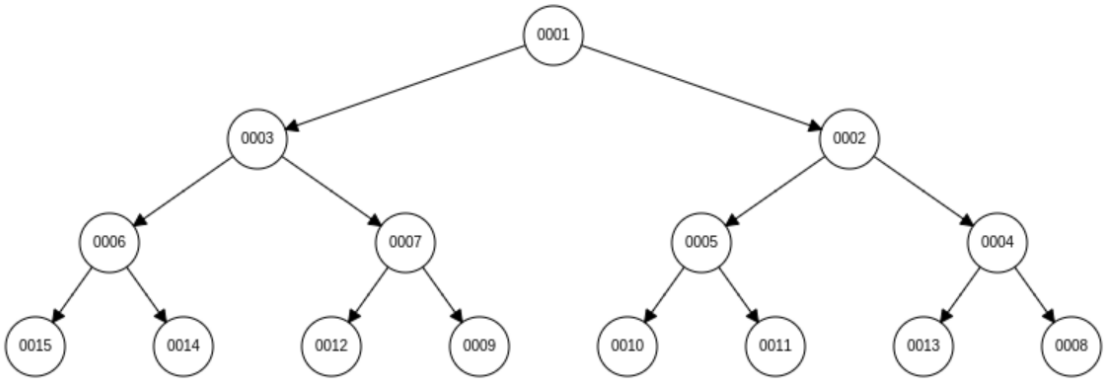


(d) Se agrega 11 como hoja y como es mayor a 5, se queda así

Figura 2: Árboles binarios en las siguientes 4 inserciones.



(a) Se agrega 13 y como es mayor a 8 así se queda.



(b) Se agrega 2 como hoja y como es mayor a 8 y a 4, sube dos niveles.

Figura 3: Árboles binarios en las últimas 2 inserciones.