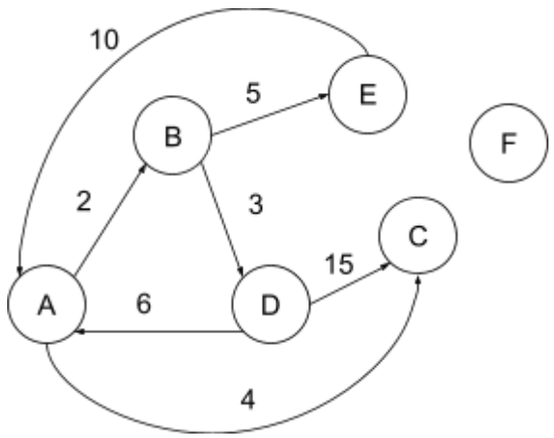
Ejercicio 4

a) **Muestre mediante una tabla el seguimiento del algoritmo de Dijkstra** sobre el grafo de la derecha **comenzando desde el vértice B**. Deberá mostrarse claramente la evolución del arreglo de distancias y del arreglo de padres, y el estado final de cada uno.



Paso	u	S	dist[A]	dist[B]	dist[C]	dist[D]	dist[E]	dist[F]	padre[A]	padre[B]	padre[C]	padre[D]	padre[E]	padre[F]
Inicialización	-	{ }	inf	0	inf	inf	inf	inf	indef	-	indef	indef	indef	indef
Loop 1 - r(u)	B	{B}	inf	0	inf	3	5	inf	indef	-	indef	B	B	indef
Loop 2 - r(u)	D	{B, D}	9	0	18	3	5	inf	D	-	D	B	B	indef
Loop 3 - r(u)	E	{B, D, E}	9	0	18	3	5	inf	D	-	D	B	B	indef
Loop 4 - r(u)	A	{B, D, E, A}	9	0	13	3	5	inf	D	-	A	B	B	indef
Loop 5 - r(u)	C	{B, D, E, A, C}	9	0	13	3	5	inf	D	-	A	B	B	indef
Loop 6 - r(u)	F	{B, D, E, A, C, F}	9	0	13	3	5	inf	D	-	A	B	B	indef

El algoritmo de **Dijsktra** es un algoritmo greedy para resolver el **problema de los caminos más cortos desde un vértice origen hacia el resto de los vértices del grafo**.

La técnica greedy encuentra **siempre** la mejor solución (solución óptima).

Condición para su aplicación → Los arcos deben tener asignado un valor **POSITIVO**.

b) Se tiene un **árbol binario de búsqueda A** que **es un árbol completo**, y se sabe que el método de la derecha al ser invocado con la raíz del árbol A **(buscahoja(raiz, 100))** imprime el valor **93**.

i) ¿Cuál es la cantidad máxima de hojas de A?
ii) ¿Cuál es la cantidad máxima de nodos internos de A?

```
// private void buscahoja(TreeNode nodo, int k) {
        If (nodo.getIzq() != null) {
                buscahoja(nodo.get.Izq(), k - 1);
        } else {
                imprimir(k);
        }
}
```

→ Tiene 7 niveles!

i) 2^7 = 128
ii) (2^7) - 1 = 127 → Se resta la raíz

Estructuras Especiales de Árboles Binarios

- Arboles Degenerados (enredaderas) → nodos internos con sólo un hijo.
- Balanceados → las alturas de los dos subárboles de cualquier nodo difieren a lo sumo en 1.
- Completos → todos sus nodos tienen dos hijos, excepto los del último nivel que no tienen hijos.

Ejercicio 5

Responda Verdadero o Falso, justificando su respuesta:

i) En notación big-O si un algoritmo es O(1) quiere decir que tardará siempre 1 milisegundo en ejecutar.

Falso → Si un algoritmo tiene una complejidad de O(1) quiere decir que su costo de ejecución es constante, independientemente del tamaño de la entrada del problema.

ii) Un proceso con un solo Thread no requerirá de la sincronización de acceso a métodos.

Verdadero → Al haber varios Threads accediendo al mismo recurso, usamos sincronización. Si hay un solo Thread no tiene sentido usar sincronización.

iii) La estrategia de lista de factoreo no tiene sentido si usamos una estructura HashTable para guardar la información.

Falso → Ojo confusión!

Lista de rebalse → Dos o más claves que colisionan y van a parar al mismo balde aparece la lista de rebalse que guarda la información asociada a esas claves.

Lista de factoreo → Se usa cuando se tiene información, distintos datos que comparten la misma clave, es decir, cuando se usa de clave para hashing algo que no es único. Por ejemplo, si para guardar alumnos se usa como clave el DNI, no habrán claves repetidas, pero si la clave es Año de Nacimiento, pueden muchos alumnos cumplir en el mismo año.

La lista de factoreo es una decisión propia. Tiene sentido usarla en cualquier estructura de hashing, siempre y cuando la clave que utilizo no sea única, que varios elementos puedan tener la misma clave.