Algoritmos y Programación II (75.41, 95.15) – Curso Buchwald

3.ra fecha de examen final – 06/02/2020 – Por favor, reciclar este enunciado

Aclaración: salvo que se indique lo contrario, los ejercicios que involucren programar pueden realizarse en pseudocódigo.

1. Implementar un algoritmo de División y Conquista que determine si un arreglo se encuentra ordenado. Indicar y justificar el orden del algoritmo, utilizando el Teorema Maestro.
2. Escribir el algoritmo de Dijkstra para obtener los caminos mínimos desde un vértice hacia todos los demás vértices del grafo. Indicar y justificar el orden del algoritmo. Explicar (detalladamente) cómo podemos modificar el algoritmo de Dijkstra para que, además de encontrar un camino mínimo, devuelva la cantidad de caminos mínimos que hay para llegar a cada vértice (tener en cuenta que puede haber más de un camino mínimo).
3. Implementar un algoritmo que ordene un arreglo con puntos (valores `(x, y)`) que se encuentran dentro del círculo unitario $\left(x^2 + y^2 \leq 1\right)$, sabiendo que la distribución de los puntos **es uniforme en dicho dominio**. El criterio para ordenar es de menor a mayor norma (distancia al origen). Tener en cuenta que los números pueden tener *"infinitos"* decimales. El algoritmo debe ejecutar en tiempo lineal a la cantidad de elementos del arreglo a ordenar. Justificar el orden del algoritmo propuesto.
4. Diseñar un TDA Multiconjunto. Dicho TDA debe permitir agregar y eliminar elementos, y preguntar si pertenecen. La diferencia con el TDA Conjunto es que permite agregar más de una vez un elemento, y para que el elemento deje de pertenecer debe haberse eliminado tantas veces como se haya agregado. Indicar y justificar el orden de cada primitiva.
5. Implementar en C una primitiva que reciba un árbol binario izquierdista, y la cantidad de nodos que tiene, y devuelva el dato del elemento más a abajo y a la derecha del árbol. En los árboles de las figuras del dorso, se debe devolver en ambos casos 4. Para que el ejercicio se pueda considerar como aprobable, debe resolverse en no más que $\mathcal{O}(n)$, sin contar con otros errores. Para que se considere completamente bien, debe ejecutar en $\mathcal{O}(\log n)$. Justificar el orden del algoritmo implementado. A fines del ejercicio, considerar que la estructura del árbol binario es:

```c
typedef struct ab {
    struct ab* izq;
    struct ab* der;
    void* dato;
} ab_t;
```

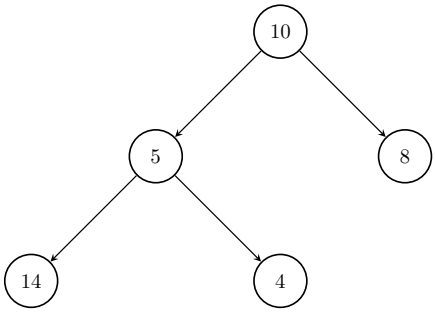
Algoritmos y Programación II (75.41, 95.15) – Curso Buchwald

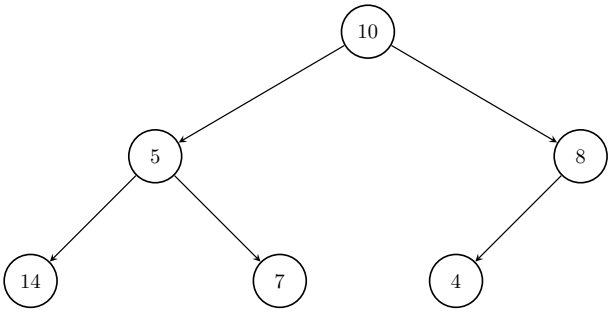
3.ra fecha de examen final – 06/02/2020 – Por favor, reciclar este enunciado

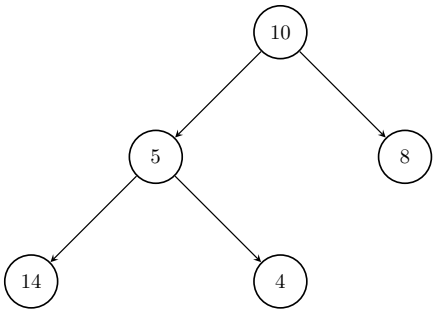
Aclaración: salvo que se indique lo contrario, los ejercicios que involucren programar pueden realizarse en pseudocódigo.

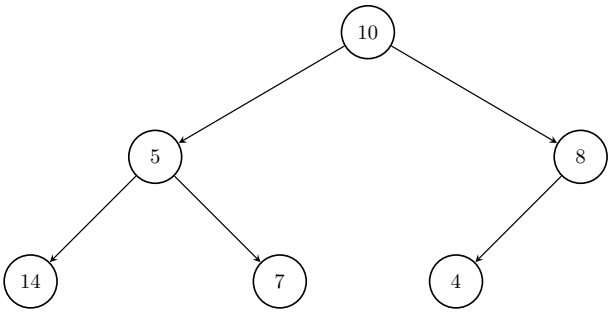
1. Implementar un algoritmo de División y Conquista que determine si un arreglo se encuentra ordenado. Indicar y justificar el orden del algoritmo, utilizando el Teorema Maestro.
2. Escribir el algoritmo de Dijkstra para obtener los caminos mínimos desde un vértice hacia todos los demás vértices del grafo. Indicar y justificar el orden del algoritmo. Explicar (detalladamente) cómo podemos modificar el algoritmo de Dijkstra para que, además de encontrar un camino mínimo, devuelva la cantidad de caminos mínimos que hay para llegar a cada vértice (tener en cuenta que puede haber más de un camino mínimo).
3. Implementar un algoritmo que ordene un arreglo con puntos (valores `(x, y)`) que se encuentran dentro del círculo unitario $\left(x^2 + y^2 \leq 1\right)$, sabiendo que la distribución de los puntos **es uniforme en dicho dominio**. El criterio para ordenar es de menor a mayor norma (distancia al origen). Tener en cuenta que los números pueden tener *"infinitos"* decimales. El algoritmo debe ejecutar en tiempo lineal a la cantidad de elementos del arreglo a ordenar. Justificar el orden del algoritmo propuesto.
4. Diseñar un TDA Multiconjunto. Dicho TDA debe permitir agregar y eliminar elementos, y preguntar si pertenecen. La diferencia con el TDA Conjunto es que permite agregar más de una vez un elemento, y para que el elemento deje de pertenecer debe haberse eliminado tantas veces como se haya agregado. Indicar y justificar el orden de cada primitiva.
5. Implementar en C una primitiva que reciba un árbol binario izquierdista, y la cantidad de nodos que tiene, y devuelva el dato del elemento más a abajo y a la derecha del árbol. En los árboles de las figuras del dorso, se debe devolver en ambos casos 4. Para que el ejercicio se pueda considerar como aprobable, debe resolverse en no más que $\mathcal{O}(n)$, sin contar con otros errores. Para que se considere completamente bien, debe ejecutar en $\mathcal{O}(\log n)$. Justificar el orden del algoritmo implementado. A fines del ejercicio, considerar que la estructura del árbol binario es:

```c
typedef struct ab {
    struct ab* izq;
    struct ab* der;
    void* dato;
} ab_t;
```


10
5
8
14
4


10
5
8
14
7
4


10
5
8
14
4


10
5
8
14
7
4