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

2.da fecha de examen final – 19/12/2019 – Por favor, reciclar este enunciado

1. Implementar en C una primitiva para el ABB que nos permita obtener el segundo máximo elemento del ABB, con firma `const char* abb_segundo_maximo(const abb_t*)`. En caso de tener menos de dos elementos, devolver `NULL`. La primitiva debe ejecutar en $\mathcal{O}(\log n)$. Justificar el orden del algoritmo propuesto.

2. Implementar una función que reciba un arreglo de n números y un número k, y determine si hay dos elementos a exactamente distancia k en $\mathcal{O}(n)$, siendo n la cantidad de elementos del arreglo. Justificar el orden del algoritmo implementado. Ejemplos: `[1, 2, 13, 40, 10, 20] K = 2 --> False` `[1, 2, 13, 40, 10, 20] K = 3 --> True (13 y 10)`

   ¿Cómo cambiarías tu algoritmo si, en vez de determinar si hay dos elementos a exactamente distancia k, se pidiera que *cómo máximo* fuera distancia k? ¿Cómo sería la complejidad del nuevo algoritmo?

3. Responder justificando de forma breve y concisa las siguientes preguntas:

   a. ¿Cuál es la complejidad de heapsort? ¿es posible que conociendo información adicional logremos mejorar su complejidad?

   b. ¿Qué implica que un algoritmo de ordenamiento sea estable? ¿Es mergesort un algoritmo de ordenamiento estable?

   c. ¿Es siempre mejor utilizar Counting Sort para ordenar un arreglo de números enteros por sobre utilizar un ordenamiento por Selección?

4. Definimos el siguiente algoritmo de División y Conquista que, esperamos, nos permita obtener el árbol de tendido mínimo de un Grafo. El algoritmo divide el conjunto de los vértices en dos, y llama recursivamente con un *nuevo grafo* sólo conformado por los vértices de una mitad, y las aristas que unen vértices de esa mitad. Llama para ambas mitades, y resuelve recursivamente el árbol de tendido mínimo de cada mitad (el caso base sería un grafo con un único vértice). Luego de resolver el árbol de tendido mínimo de ambas mitades, une ambos con la arista mínima que une los vértices de un lado con los del otro. Indicar y Justificar detalladamente cuál sería la complejidad de dicho algoritmo. ¿El algoritmo propuesto permite obtener el árbol de tendido mínimo? (suponer que el grafo es conexo). En caso de ser cierto, justificar, en caso de ser falso dar un contraejemplo.

5. Tenemos un conjunto de Mariposas que queremos determinar si pertenecen a tan sólo dos tipos. Se tienen M mariposas, y se obtuvieron D comparaciones que indican que dos mariposas pertenecen a grupos diferentes. Implementar un algoritmo que permita determinar si efectivamente podemos separar en dos grupos a las mariposas (para esto, no pueden dos mariposas "diferentes" estar en el mismo grupo). El algoritmo debe funcionar en tiempo linea a M y D. Justificar la complejidad del algoritmo implementado.

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

2.da fecha de examen final – 19/12/2019 – Por favor, reciclar este enunciado

1. Implementar en C una primitiva para el ABB que nos permita obtener el segundo máximo elemento del ABB, con firma `const char* abb_segundo_maximo(const abb_t*)`. En caso de tener menos de dos elementos, devolver `NULL`. La primitiva debe ejecutar en $\mathcal{O}(\log n)$. Justificar el orden del algoritmo propuesto.

2. Implementar una función que reciba un arreglo de n números y un número k, y determine si hay dos elementos a exactamente distancia k en $\mathcal{O}(n)$, siendo n la cantidad de elementos del arreglo. Justificar el orden del algoritmo implementado. Ejemplos: `[1, 2, 13, 40, 10, 20] K = 2 --> False` `[1, 2, 13, 40, 10, 20] K = 3 --> True (13 y 10)`

   ¿Cómo cambiarías tu algoritmo si, en vez de determinar si hay dos elementos a exactamente distancia k, se pidiera que *cómo máximo* fuera distancia k? ¿Cómo sería la complejidad del nuevo algoritmo?

3. Responder justificando de forma breve y concisa las siguientes preguntas:

   a. ¿Cuál es la complejidad de heapsort? ¿es posible que conociendo información adicional logremos mejorar su complejidad?

   b. ¿Qué implica que un algoritmo de ordenamiento sea estable? ¿Es mergesort un algoritmo de ordenamiento estable?

   c. ¿Es siempre mejor utilizar Counting Sort para ordenar un arreglo de números enteros por sobre utilizar un ordenamiento por Selección?

4. Definimos el siguiente algoritmo de División y Conquista que, esperamos, nos permita obtener el árbol de tendido mínimo de un Grafo. El algoritmo divide el conjunto de los vértices en dos, y llama recursivamente con un *nuevo grafo* sólo conformado por los vértices de una mitad, y las aristas que unen vértices de esa mitad. Llama para ambas mitades, y resuelve recursivamente el árbol de tendido mínimo de cada mitad (el caso base sería un grafo con un único vértice). Luego de resolver el árbol de tendido mínimo de ambas mitades, une ambos con la arista mínima que une los vértices de un lado con los del otro. Indicar y Justificar detalladamente cuál sería la complejidad de dicho algoritmo. ¿El algoritmo propuesto permite obtener el árbol de tendido mínimo? (suponer que el grafo es conexo). En caso de ser cierto, justificar, en caso de ser falso dar un contraejemplo.

5. Tenemos un conjunto de Mariposas que queremos determinar si pertenecen a tan sólo dos tipos. Se tienen M mariposas, y se obtuvieron D comparaciones que indican que dos mariposas pertenecen a grupos diferentes. Implementar un algoritmo que permita determinar si efectivamente podemos separar en dos grupos a las mariposas (para esto, no pueden dos mariposas "diferentes" estar en el mismo grupo). El algoritmo debe funcionar en tiempo linea a M y D. Justificar la complejidad del algoritmo implementado.