Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 — Estructuras de Datos y Algoritmos
2022 - 1

Pauta Interrogación 2

Notas Generales

Pregunta 1

La siguiente es solo una de las soluciones propuestas, existen varias, aunque para optar a puntaje completo se debe describir alguna estructura con una performance similar a $\mathcal{O}(1+\epsilon)$

(a) Una solución eficiente para el problema presentado es utilizar tablas de hash, donde el numero de alumno de hashea hacia algun indice, y posteriormente se accede a la tabla de hash. Ahi existen varias opciones. La preferida en este caso para gestionar las actualizaciones es asumir que existiran colisiones y por ende se anida otra tabla de hash que llamaremos *Indice*. Finalmente el procedimiento es

$$\text{NumeroAlumno} \rightarrow \text{HashTable} \rightarrow \text{Indice} \rightarrow \text{Data}$$

De esa forma la complejidad de busqueda, insercion y actualizacion no superara la complejidad de cada funcion de hash. (Se podria usar una tabla de referencia como funcion de hash). Y finalmente mantiene complejidad similar a $\mathcal{O}(1)$ versus el $\mathcal{O}(log(n))$ entregado por el BST. Notar que las actualizaciones son simples, ya que basta con agregar o eliminar algo a las tablas de indice.

(b) En resumen los algoritmos son los siguientes:

- Hashear el numero de alumno para obtener un indice en tabla, acceder a dicha celda, y utilizar la nueva estructura de esa celda (BST u otra tabla de hash). Finalmente encontrar la entrada correspondiente (o crear, o eliminar)

(c) El algoritmo mas simple, es realizar un traverse del BST, y luego utilizar los indices de orden para asi definir su resultado en la tabla de hash. Aca hay muchas formas de realizarlo

Distribucion de puntaje

(a)
- 0,2pts Por justificar mejoras en busqueda e insercion
- 0,3pts Por Justificar mejoras en necesidades de actualizacion
- 0,5pts Por utilizar tablas de hash, o alguna estructura que permita complejidad **igual o mejor**[1]

Total: 1pts

[1] El enunciado solicita explicitamente una solucion rapida y eficiente, por lo que si o si ha de ser considerablemente mejor a el BST

(*b*) 0.15 por cada operacion si es que el algoritmo es correcto y corresponde con lo descrito en la parte 1. Total, 0.45pts

(*c*) 0.55 puntos en total. Puntaje completo en caso de algoritmo correcto y justificado. Parcial a criterio del corrector[2]

[2]En caso de que el algoritmo no quede claro, se aplicaran descuentos

Pregunta 2

Parte A 1pt

El input es fecha llamada, hora llama, número origen, número destino y duración. El output por cada número de origen, entregar sus llamadas en forma cronológica y su costo total.

Para esto se utilizará el algoritmo de ordenamiento radixSort (0.6 pts por implementación), donde se aplica por cada campo, esto usando como subrutina un algoritmo de ordenación estable (0.2pts mencionar que sea estable) (por ejemplo countingSort). Para tener un orden final respecto a los números de origen se puede aplicar como último campo de ordenamiento a origen[3]. El tiempo es lineal (0.1 pts complejidad)

Ya con el listado de números ordenados de forma cronológica se realiza una última pasada por todos los registros para calcular el costo total respecto a cada número, esto es lineal. (0.1 pts por el costo total)

En caso de resolver el problema con tablas de hash se tienen que tener las siguientes consideraciones:

Si solo se menciona el uso de una tabla de hash el puntaje máximo a obtener es de 0.6 pts, esto es porque no se menciona como lidiar con las colisiones entre números distintos y se está asumiendo que la forma de realizar hash solamente tendrá colisiones para el mismo número, cosa que no es cierta por las diferencias de tamaño de tabla y dominio.

Si las inserciones de la tabla de hash solo hacen mención a que son de forma cronológica esto está incorrecto, ya que el objetivo es ordenar los datos.

Si sobre las colisiones de un mismo número se aplica una estructura de árbol para ordenar los nodos se tiene que mencionar que se ordena por múltiples criterios (0.4 pts), para manejar los casos en que los valores de los nodos sean iguales, por ejemplo para un mismo día se realizan llamadas a distintas horas ¿cómo se diferencian estas horas?

0.2pts por complejidad y del costo total.

Parte B 0.5 pts

Para mostrar que cumple con los requisitos de cada característica, se tiene que cumplir un orden relativo a dicha característica, por ejemplo, que para un mismo día, número de destino, número de origen, duración, pero distintas horas exista este orden. Esto se cumple si se utiliza un algoritmo estable. Es decir se respeta la separación por número de origen, orden cronológico y calculo de costo total.

Si solo se asume que de la parte A vienen ordenados cronológicamente (max 0.35 pts).

Parte C 0.5 pts

La solución planteada es lineal y la del compañero es cuadrática, además de que se tiene que asegurar de su estabilidad. Nuestra solución es más eficiente.

[3] Alternativamente se puede partir separando por el número de origen y luego realizar el resto de operaciones

Pregunta 3

Parte A 0.5 pts

Identificar

1. variables
   a) x, Pos del robot en la grilla.
   b) * Pos inicial del robot
   c) Pos celdas inseguras.
   d) * Cantidad de celdas inseguras.
2. dominio
   a) X_i, Y_j
   b) X_1, Y_j
   c) existen K celdas, donde un elemento puede estar en X_i, Y_j. se genera una matriz de la forma $X_i k, Y_j k$ con k celdas inseguras.
   d) sean k celdas inseguras, k $\epsilon(0, n - (\Sigma(X_i))$
3. restricciones
   a) el robot no puede moverse a una celda insegura ni a ninguna celda adyacente insegura. (Por hacer dem formal)
   b) No puede existir No existir un camino de la izquierda de la columna a la nueva, además por columna debe haber almenos 1 casilla con celda segura NO adyacente a una celda insegura.
   c) El robot debe iniciar en una celda C = X_1, Y_j, esta celda no puede estar sobre una casilla insegura ni adyacente a esta. (por hacer dem formal)
   d) la cantidad de celdas inseguras debe garantizar que el problema sea satisfacible, en caso de ser un camino recto la sol (mejor sol), pueden haber max (por hacer dem formal)

   Notas: los elementos b, d son opcionales (marcados con *) el resto no. distribuir puntaje equitativamente, No otorgar puntaje parcial.

Parte B 0.75 pts

Se le pide al alumno un algoritmo que resuelva algun camino para el robot. Puntaje total si el algoritmo resuelve el problema, se aceptan soluciones recursivas, iterativas y con backtracking. No se otorga puntaje parcial.

Algoritmo de ejemplo :

Parte C 0.75 pts

Se le pide al alumno encontrar el mejor camino, se espera que el alumno adapte su algoritmo de la parte B, agregar todas las soluciones y luego escoger la mejor de estas. No se otorga puntaje parcial.

Algoritmo de ejemplo :

Pregunta 4

a. 1 pto

El árbol Rojo-Negro se desbalancea primero. Un ejemplo a considerar: Se han colocado 3 nodos en orden en ambos árboles AVL y rojo-negro, ambos están balanceados. El árbol rojo-negro tiene negra su raíz y rojo ambos hijos (por sus propiedades). Al colocar un nodo con la misma clave en ambos árboles, por una parte, el árbol AVL seguirá siendo balanceado puesto que la diferencia entre las hojas difiere a lo más 1 (propiedad AVL). Por otra parte, el árbol rojo-negro, dicha inserción (independiente del nodo hoja al que se haya colocado) es un nodo inicialmente rojo, y como tanto su padre como su tío son rojos, ya está violando la propiedad 3 del árbol rojo-negro (si un nodo es rojo, sus hijos son negros), por lo que después de haber colocado el cuarto nodo al árbol rojo-negro, éste debe balancearse.

- 0.3 pts por señalar cual se desbalancea primero
- 0.3 pts por señalar propiedades involucradas
- 0.4 pts por dar justificación

b. 0.5 pto

En general, deben dibujar un árbol rojo-negro, y señalar cual propiedad AVL se violaría al olvidarse de los colores rojo-negro.

- 0.2 ptos por dibujar un árbol rojo-negro que cumple sus propiedades (es un ABB más las 4 propiedades del rojo-negro)
- 0.3 ptos por señalar cual propiedad del AVL estaría incumpliendo al olvidarse de los colores. Que en este caso, sería la propiedad del balance (que al olvidar los colores de nodos, es simplemente un ABB desbalanceado).

Cualquier otra forma de resolver el problema queda a criterio del ayudante.

c. 0.5 ptos

Se puede resolver de distintas formas, pero una explicación puede ser la siguiente: La raíz siempre se pinta de negra, se puede pintar los nodos por nivel (alternándose). Si el árbol tiene todos sus nodos en balance 0, basta con pintar de manera alternada para tener las propiedades de rojo-negro. Luego, es importante que mencionen que el caso crítico sería cuando hay una diferencia de 1 en algún subárbol, donde ahí deben mencionar como podrán pintar los nodos de forma que se pueda mantener las propiedades.
El puntaje se distribuye a criterio del ayudante con respecto a la justificación.