ANÁLISIS Y DISEÑO DE ALGORITMOS I

PROYECTO DE CURSADA 2021

Segunda parte

Objetivo

La consultora decide extender los servicios de consulta disponibles para los clientes para brindar un nuevo servicio.
Por este motivo, nos solicita realizar las modificaciones necesarias en la aplicación para resolver, eficientemente, el siguiente servicio:

- Obtener la información de las empresas que están dentro de los primeros N puestos del ranking pero que, además, cuentan con una cantidad de empleados dentro de un rango determinado.
  Por ejemplo, para construir un reporte necesitamos las primeras 20 empresas del ranking pero que cuentan con un número de empleados entre 200 y 1000 personas.

Captura de requerimientos

Es necesario construir una nueva aplicación (o adaptar la existente) que, partiendo de lo realizado para el primer proyecto, resuelva eficientemente este nuevo servicio.
La información de las empresas seguirá siendo obtenida desde un único archivo de texto y respetará las mismas restricciones establecidas anteriormente.

Diseño de la solución

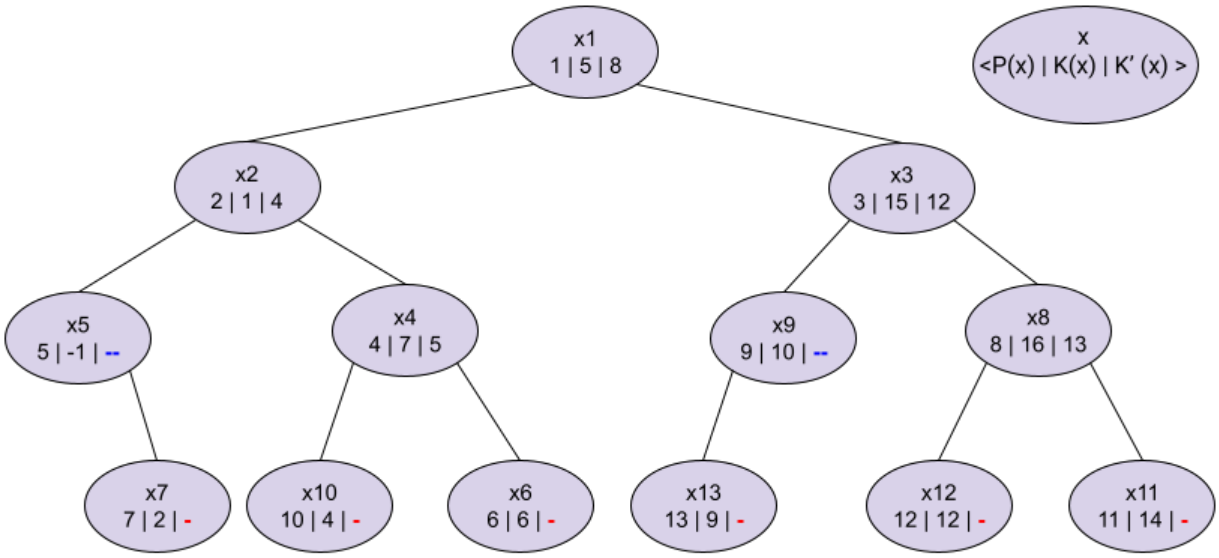
Para resolver eficientemente el servicio es necesario construir una estructura de búsqueda conocida como **Priority Search Tree**.
Las estructuras de búsqueda generalmente se construyen sobre estructuras generales de almacenamiento. De esta forma, la información se encuentra almacenada **una única vez** en la estructura general y las aplicaciones pueden construir las estructuras de búsqueda que consideren apropiadas para acceder a los datos.
Acceder a los datos que se están buscando a través de estas estructuras de búsqueda resulta en resoluciones mucho más eficientes, en comparación con resolver el problema directamente utilizando la estructura general.
Priority Search Tree es una estructura de búsqueda muy aplicada en un contexto particular, como se verá a continuación.

Fundamentación teórica

Un **Priority Search Tree** es una estructura de datos que permite almacenar información según dos coordenadas; originalmente desarrollada para almacenar puntos en un plano de 2 dimensiones y luego buscar rangos entre esos puntos. Esta estructura se puede representar mediante un árbol binario de búsqueda donde cada nodo almacena un valor que representa la prioridad y otro valor clave que indica cómo fueron separados el resto de los elementos, a partir de ese nodo.
Por ejemplo, estableciendo:

- x: un elemento disponible.
- P(x): la prioridad del elemento x.
- K(x): la clave del elemento x.
- K'(x): la clave de distribución de los elementos a partir de x.

podemos construir el árbol de la siguiente figura.



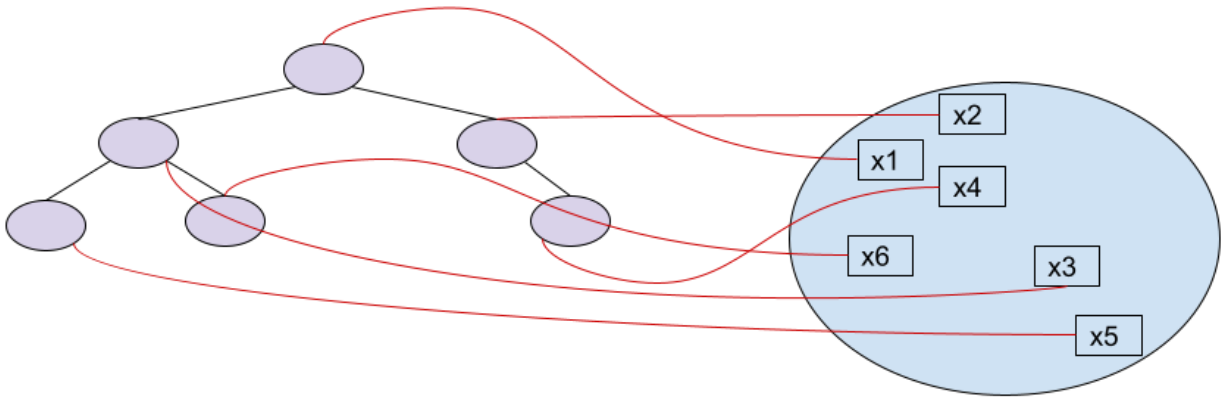
Como se puede ver, a la izquierda de cada nodo se almacenan los elementos con valor menor a la clave del nodo padre (K') mientras que a la derecha se almacenan los elementos con valor mayor a la clave K'.
Es importante notar que hay nodos que no tienen un valor de K'(x) ya que no cuentan con 2 nodos hijos, por lo tanto no se requiere una clave de distribución para saber por dónde avanzar. En el diagrama están marcados con "-" y con "--" (según sean hojas o no) pero es una decisión de implementación cómo se registra esta situación.

Construcción del Priority Search Tree

El Priority Search Tree se construye a partir de una colección de elementos, utilizando las funciones P(x), K(x) y K'(x) definidas en la sección anterior. A partir de esa información podemos esquematizar el siguiente pseudo-código:

```
priority_search_tree construir_pst(conjunto_puntos S)
{
   si vacio(S)
      retornar null
   sino
      E = elemento_menor_prioridad(S)
      S' = S - E
      K' = calcular_mediana_K(S')

      S_i = puntos_izquierda_mediana(S',K')
      S_d = puntos_derecha_mediana(S',K')

      priority_search_tree I = construir_pst (S_i)
      priority_search_tree D = construir_pst (S_d)

      T = construir_nodo(E, K')
      T.izquierda = I
      T.derecha = D

      retornar T
}
```

En el siguiente esquema se puede visualizar el vínculo resultante entre la estructura general de almacenamiento y la estructura de búsqueda, para un caso acotado:



De esta forma, el contenedor de empresas sería la estructura general de almacenamiento mientras que el Priority Search Tree sólo nos brinda un acceso eficiente a esos datos.
Importante: Se debe tomar una buena decisión a la hora de construir el vínculo entre el nodo del Priority Search Tree y el elemento concreto almacenado en el contenedor de empresas, para que su acceso no empeore la eficiencia de resolución de las búsquedas.

Utilización del Priority Search Tree

La resolución del servicio deberá utilizar el Priority Search Tree como punto de acceso a la información. El cliente brindará un rango de cantidad de empleados y una posición máxima en el ranking; utilizando esta estructura de búsqueda, se deberá retornar la información de las empresas que coincidan con el criterio de búsqueda ingresado.

De forma resumida se puede esquematizar en el siguiente pseudo-código el proceso de detectar los nodos del árbol que cumplen con la búsqueda solicitada:

```
solucion buscar_pst(priority_search_tree T, prioridad P,
                                valor_minimo MIN, valor_maximo MAX)
{
   solucion S
   si es_hoja(T)
      si (T.prioridad <= P) y (MIN <= T.valor_elemento <= MAX)
         S.agregar(T.elemento)
         retornar S
   sino
      si T.prioridad > P
         retornar vacio
      sino
         si (MIN <= T.valor_elemento <= MAX)
            S.agregar(T.elemento)
         si MIN < T.valor_separacion_K'
            S.unir(buscar_pst(T.izquierda, P, MIN, MAX)
         si T.valor_separacion_K' < MAX
            S.unir(buscar_pst(T.derecha, P, MIN, MAX)

         retornar S
}
```

Requisitos de la entrega

Correcciones realizadas a la primera entrega según lo pedido por el ayudante asignado.
Un informe que contenga:

- identificación de los integrantes del grupo (nombre de cada integrante, número de grupo y dirección de e-mail),
- analizar la complejidad temporal de resolver el servicio solicitado con la estructura elegida para la primera entrega,
- detallar las estructuras utilizadas para almacenar en memoria la colección de empresas,
- explicación del algoritmo de construcción del Priority Search Tree,
- detallar las decisiones de diseño tomadas para construir las estructuras de almacenamiento y búsqueda, e implementar dicho algoritmo,
- explicación de la utilización del Priority Search Tree para resolver el servicio solicitado,
- calcular la complejidad temporal de la resolución del servicio completo,
- código fuente desarrollado para implementar la consigna de la segunda etapa del trabajo.

Entrega digital:

- identificación de los integrantes del grupo,
- versión digital del informe,
- proyecto con el código fuente de la solución implementada.

Fecha de entrega y modalidad

La entrega final se realizará el **miércoles 30/6 a las 17 hs**, siguiendo los mismos lineamientos que para la primera entrega.
Importante: no se aceptarán entregas de la versión final fuera de término.