Practica de Estrategias de programación y estructura de datos – Junio 2015

Nombre: Miroslav Krasimirov Vladimirov
Correo: mkrasimir4@alumno.uned.es, miro.kv89@gmail.com
Teléfono: 676867565

Preguntas teóricas

1.1 ¿Cómo depende el tamaño de almacenamiento del número de consultas diferentes en el registro de consultas?

-El tamaño de almacenamiento varia con la cantidad de consultas diferentes, a mas consultas distintas, mayor tamaño de almacenamiento.
Tamaño almacenamiento total = (tamaño almacenamiento 1 consulta) * (Nº de consultas distintas).

¿Cómo depende del número de repeticiones?

-El numero de repeticiones en cambio no varia puesto que lo que varia es un el interior atributo de los que contiene Query y no se añaden mas Queries.

1.2 ¿Cómo depende el tiempo de localizar todas las posibles sugerencias de búsqueda del número de consultas diferentes?

-En todos los casos, para encontrar las sugerencias hay que recorrer la lista entera, por lo que el coste es T(N) = O(N). Por lo tanto el tiempo dependerá de la cantidad de consultas diferentes que existen.

¿Y del número de repeticiones?

-El numero de repeticiones no influye para encontrar las distintas sugerencias, tan solo en el orden en el que se muestran, por lo que el tiempo sigue siendo T(N) = O(N).

¿Y de la longitud máxima de las consultas medida como el número de caracteres de su texto?

-La longitud del prefijo a buscar tanto como la longitud de las consultas realizadas con anterioridad no influye en el tiempo.

¿Y del tamaño del conjunto de caracteres permitido?

-La longitud de caracteres de las consultas no influye en el tiempo.

Razona en términos del coste asintótico temporal en el caso peor que se podría conseguir para el método listOfQueries con este diseño.

El caso peor de listOfQueries es que el prefijo buscado sea nulo, por lo que se mostrarian por pantalla todas las consultas realizadas:

T(N) = O(N).

1.3 Consideremos una estructura de datos alternativa en la que disponemos de una lista de consultas similar a la considerada por cada carácter inicial (es decir, tendríamos una lista con todas las consultas que empiezan por el carácter “a”, otra con las que empiezan por “b”, etc.)
¿Cuánto se podría reducir el tiempo de búsqueda en caso peor?

-El coste de tiempo se reduciría, puesto que no habría que buscar en todos los elementos de la lista sino solo en una parte de ellos. En el peor de los casos el coste temporal seria:
T(N) = 26+N/27
T(N) = (conjunto de caracteres-1) + N/(conjunto de caracteres)

¿Cambiaría el coste asintótico temporal?

-El coste asintónico temporal sigue siendo O(N).

1.4 ¿Se ajusta este diseño a los requisitos del problema? Explica por qué.

-El diseño se ajusta a los requisitos, puesto que el coste temporal no es muy elevado, y se

cumple la función que se requiere del programa: imprime la frecuencia de las búsquedas y también sugerencias. En caso de una lista de consultas muy larga pierde eficiencia, pero para listas cortas es ideal.

2.1 ¿Cómo depende el tamaño de almacenamiento del número de consultas diferentes en el registro de consultas?

-El tamaño de almacenamiento es menor que en el caso de las listas, debido a que consultas que tengan prefijos comunes ocuparan menos espacio para dicho prefijo, ya que lo compartirán.

¿Y del tamaño máximo de las consultas?

-A mas consultas, mayor sera la diferencia de tamaño de almacenamiento en comparación con el caso de las Listas, puesto que se compartirán mas prefijos.

¿Y del tamaño del conjunto de caracteres permitido?

-Cuantos mas caracteres se permitan mayor sera el tamaño de almacenamiento, puesto que habría que crear mas nodos en el árbol a niveles mas bajos.

Consideremos un conjunto de 50 caracteres y un tamaño máximo de consulta de 10 caracteres. Compara el tamaño máximo del árbol en la segunda aproximación con el tamaño máximo de la lista en la primera aproximación. ¿La diferencia se agrandará o se reducirá para conjuntos de caracteres mayores y consultas más largas?

-El tamaño máximo del árbol serian 55 nodos, esto son 5 palabras de 10 caracteres y 5 nodos adicionales para guardar la frecuencia. En el caso de las listas el tamaño máximo serian 50 objetos Query de 1 letra cada uno.

En el caso de conjuntos de caracteres mayores, los arboles compartirán prefijos y el tamaño del almacenamiento sera menor que el de las Listas. Por otro lado para consultas mas largas el tamaño de almacenamiento de las Listas sera menor que el de los Arboles.

2.2 ¿Cómo depende el tiempo de localizar una posible sugerencia de búsqueda del número de consultas diferentes?

-En el caso de los arboles, se ahorra tiempo descartando muchos nodos que no coinciden con el prefijo sobre el que se buscan sugerencias, y cuanta mas largo sea el prefijo, menor sera el coste. El descarte de nodos tiene un coste logarítmico O(logN). Sin embargo al acabar de descartar hojas del árbol cuyas letras no corresponden con el prefijo, reconstruir las sugerencias conlleva un coste lineal O(N) para las consultas restantes.

¿Y del número de repeticiones?

-El numero de repeticiones no influye en el tiempo, ya que ese valor se guarda en un nodo, no en varios.

¿Y de la longitud máxima de las consultas medida como el número de caracteres de su texto?

-A mayor longitud de caracteres, mas nodos habrá que recorrer, por lo que el coste temporal aumentará.

¿Y del tamaño del conjunto de caracteres permitido?

-Si el tamaño del conjunto de caracteres varia, también lo hará el coste temporal, a mas caracteres permitidos, mas nodos serán creados y mayor sera el coste.

2.3 ¿Cómo depende el tiempo de localizar todas las posibles sugerencias de búsqueda del número de consultas diferentes?

-El tiempo para localizar las sugerencias depende del conjunto de caracteres usados y la longitud del prefijo buscado, de tal forma que:
T(N) = tamaño del conjunto * longitud del prefijo
Con eso localizamos el árbol con todas las posibles sugerencias. A partir de ahí el coste para reconstruir las distintas Queries es lineal O(N) con N siendo el numero de nodos, para recorrer todos los nodos y crear las sugerencias.

¿Y del número de repeticiones?

-El numero de repeticiones no influye en el coste temporal puesto que solo varia el valor del ultimo nodo, y no se crean ni borran nodos.

¿Y de la longitud máxima de las consultas medida como el número de caracteres de su texto?

-A mayor longitud de las consultas, mayor sera el coste temporal, puesto que se añadirán mas nodos en niveles mas bajos y aumenta el coste de la recursividad.

¿Y del tamaño del conjunto de caracteres permitido?

-A mayor tamaño del conjunto de caracteres, mayor sera el coste temporal, puesto que en todos los niveles habrá un mayor numero de nodos.

Razona en términos del coste asintótico temporal en el caso peor que se podría conseguir para el método listOfQueries con este diseño.

-En el peor de los casos el coste es de N siendo N el numero de nodos. Esto ocurre cuando no se utiliza un prefijo sino que se buscan todas las posibles consultas.

2.4 A la vista de las respuestas a las preguntas anteriores y de los requisitos de nuestro problema, ¿consideras éste un diseño más adecuado para QueryDepot?

-Para casos de muchas consultas el caso de los arboles es mejor, puesto que el tamaño del almacenamiento es menor, ademas si la búsqueda para las sugerencias es larga, el coste también sera muy pequeño. El caso de las listas es mejor para un tamaño histórico de consultas menor, puesto que su coste es T(N) = O(N) es lineal.

En listas de 20 palabras guardar las consultas en forma de Lista resulta unas 50 a 100 veces mas rápido que en arboles, sin embargo en una lista de consultas de 3000 palabras los arboles superan en unas 4 veces la velocidad de las Listas. Para listas de mayor tamaño que es lo que nos interesa, los arboles serán mas y mas eficientes con respeto a las Listas.

2.5 Compara el coste de encontrar todas las sugerencias posibles con el coste de ordenarlas por frecuencia (consideremos que el coste de ordenación en el caso peor sea del orden $O(n \cdot \log(n))$).

-El coste de encontrar todas las sugerencias posibles es mayor, puesto que hay que recorrer todos los nodos del arbol y reconstruir las diferentes consultas, que será O(N). Ordenarlas por frecuencia en cambio se realiza cuando ya se han encontrado las sugerencias y el coste que conlleva es O(N*log(N)).

¿Sería aconsejable comenzar a realizar sugerencias antes incluso de que el usuario comience a teclear, con este diseño? ¿Por qué?

-Se pueden realizar sugerencias de las palabras mas buscadas, pero el coste seria igual al caso peor, por lo que seria muy ineficiente, ademas no sabemos lo que el usuario quiere buscar, ni si es el mismo usuario que el que ha realizado dichas búsquedas. Lo mejor es no realizar sugerencias antes de teclear al menos la primera letra.