



UNIDAD DE ESPECIALIDADES ESPÍRITU SANTO

Facultad de Ingeniería

Escuela de Computación

Análisis de algoritmo

TÍTULO:

Algoritmo de búsqueda binaria

NOMBRES DE LOS ESTUDIANTES:

Isaac David Antepara Cerezo

Claire Daba Diop

NOMBRE DEL TUTOR:

Ing. Marco Vinicio Sotomayor Sánchez

SAMBORONDÓN, Mayo 26, 2022

Introducción

Este proyecto trata la implementación de un algoritmo de búsqueda de medicamentos basado en la información seleccionada por el usuario.

La realización del proyecto Pastilla contribuirá a la comodidad de los pacientes con pequeños síntomas y también a los que tienen medios precarios para pagar las consultas.

En el documento presente, hablará de los antecedentes, desarrollar la problemática en detalle antes de hablar de la metodología, dominio del problema y hacer la implementación del pseudocódigo.

A partir de allí, se procederá a hablar de la notación asintótica BIG O, hacer el diagrama de sistema y terminar con la complejidad del tiempo y espacial de los algoritmos que usaremos en la realización del proyecto Pastilla.

Antecedentes

Existen varias plataformas que se dedican a la recomendación de pastillas o sugerencias de tratamientos a pacientes virtuales y que usan algoritmos similares. Tratan de encontrar o buscar items similares solicitados por el usuario. Para comprobar esta similitud se requiere de una descripción completa (conjunto de características) de los ítems.

La famosa plataforma francesa Vidal se dedica en la recomendación de tratamiento o de medicamentos dependiendo del estado de salud del paciente.

Cuando un paciente sufre de una enfermedad, Vidal no le hace la recomendación de una pastilla con restricción a personas que sufren de esta enfermedad. ¿Cómo procede Vidal?

primero hay que hacer que la plataforma cuente 100% con la colaboración del usuario que solicita asistencia. Una vez que el usuario haya elegido sus síntomas, la plataforma cuenta con un sistema de búsqueda de los remedios que cuenten con las descripciones similares a las ingresadas por el usuario.

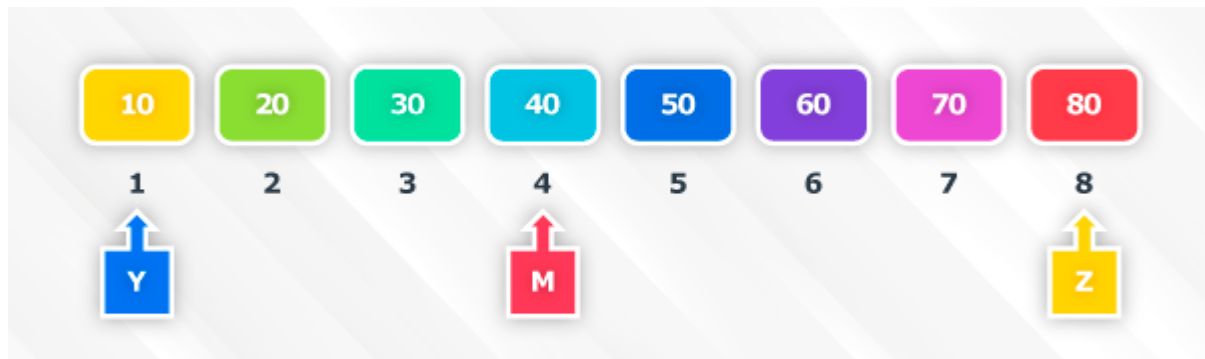
Metodología

Algoritmo de búsqueda binaria o búsqueda dicotómica

La búsqueda binaria es un algoritmo eficiente para encontrar un elemento en una lista de elementos.

Funciona de la siguiente manera: el algoritmo comienza comparando el valor objetivo con el elemento que se encuentra en la posición central del conjunto ordenado. Si el valor resulta ser el buscado, la búsqueda termina y se devuelve la posición donde se encuentra el valor. En el caso de que el valor

buscado sea menor que el valor del elemento central la búsqueda continua en el subconjunto inferior, o en el superior en el otro caso. Este proceso es iterativo y continúa hasta que el valor es encontrado o el conjunto es procesado por completo.

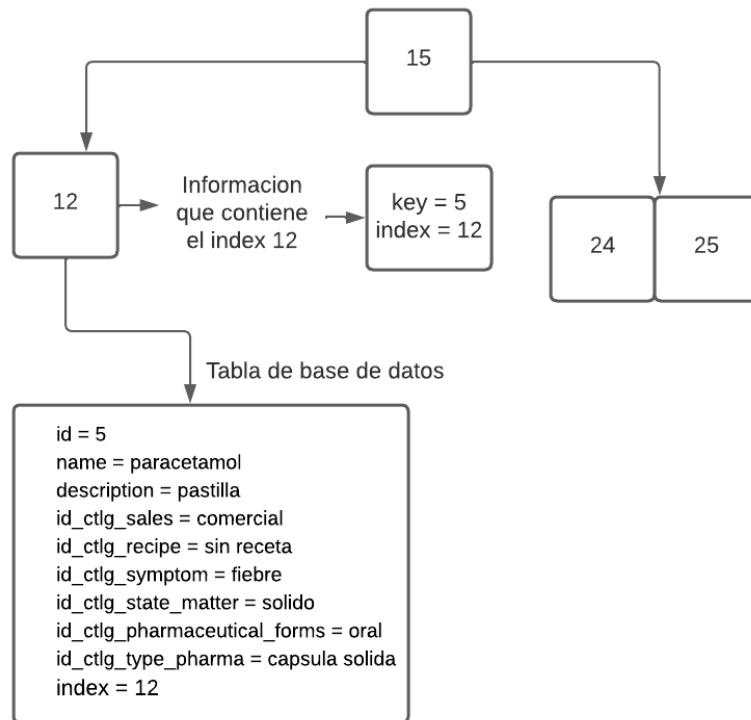


Problematica tecnica

Se creó una base de datos llamada `pastilladb` que contienen las siguientes tablas `ctlg_pharmaceutical_forms`, `ctlg_recipe`, `ctlg_sales`, `ctlg_state_matter`, `ctlg_type_pharma`, `ctlg_symptom` y `medicine`, el problema surge que al buscar una medicina referente al síntoma seleccionado el rendimiento de la base de datos tendría una notación asintótica de $O(n)$.

como solución a este problema de rendimiento se aplicara la indexación de base de datos en una base de datos, para realizar la solución se usará algoritmo de búsqueda binaria con `b-tree`.

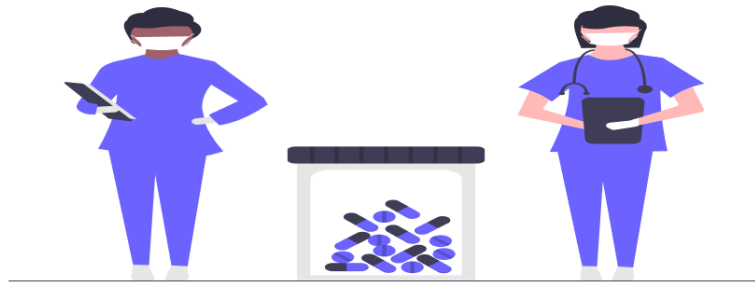
Cuando `b-tree` llega a la indexación esta estructura, la base de datos crea un índice aleatorio único(clave principal), para cada uno de los registros dados y convierte las filas relevantes en un flujo de bytes. Luego almacena cada una de las claves y registra flujos de bytes en un `b-tree`. Aquí, el índice aleatorio utilizado como clave para la indexación. El flujo de bytes y claves se conoce en conjunto como carga útil.



Aquí puede ver que todos los registros se almacenan en los nodos de hoja del b-tree y el índice se utiliza como clave para crear un b-tree. No se almacenan registros en nodos que no sean hoja. Cada uno de los nodos hoja tiene referencia al siguiente registro en el árbol. Una base de datos puede realizar una búsqueda binaria usando el índice o búsqueda secuencial buscando a través de cada elemento viajando solo a través de los nodos hoja.


- **Funcionalidades**

- Seleccionar opciones
- Visualizar los medicamentos de acuerdo a las opciones seleccionadas
- Si necesario, consultar doctor



your health in safe hands, Paztilla

Get started

 **Try a demo**

contact us

Legal

Learn more

Paztilla



Anna laure, 29 years old

symptoms

Search for symptoms

Headache



Fever



Muscular pains



Cough



Do you suffer from an illness?

Yes

No

Submit



Anna laure, 29 years old

Ibuprofen

Ibuprofen is used to relieve pain from various conditions such as headache, dental pain, menstrual cramps, muscle aches, or arthritis. It is also used to reduce fever and to relieve minor aches and pain due to the common cold or flu.

[More](#)

Paracetamol

Paracetamol is a common painkiller used to treat aches and pain. It can also be used to reduce a high temperature. It's available combined with other painkillers and anti-sickness medicines. It's also an ingredient in a wide range

[More](#)

Aspirin

Aspirin is a common drug for relieving minor aches, pains, and fevers. People also use it as an anti-inflammatory or a blood thinner. Taken daily, aspirin can lower the risk of cardiovascular events, such as a heart attack or stroke, in people with a

[More](#)

Benzonatate

Benzonatate is used to relieve cough. Benzonatate is in a class of medications called antitussives (cough suppressants). It works by reducing the cough reflex in the lungs and air passages. Benzonatate comes as a liquid-filled capsule and a

[More](#)

[See more](#)

[Contact a doctor](#)

Tecnologías a usar

- Base de datos relacional
 - MySQL
- Lado del servidor
 - Lenguaje de programación Python
 - Framework Flask
- Diseño de Interfaz gráfica
 - Adobe Experience Design
- Lado del cliente
 - Reactjs

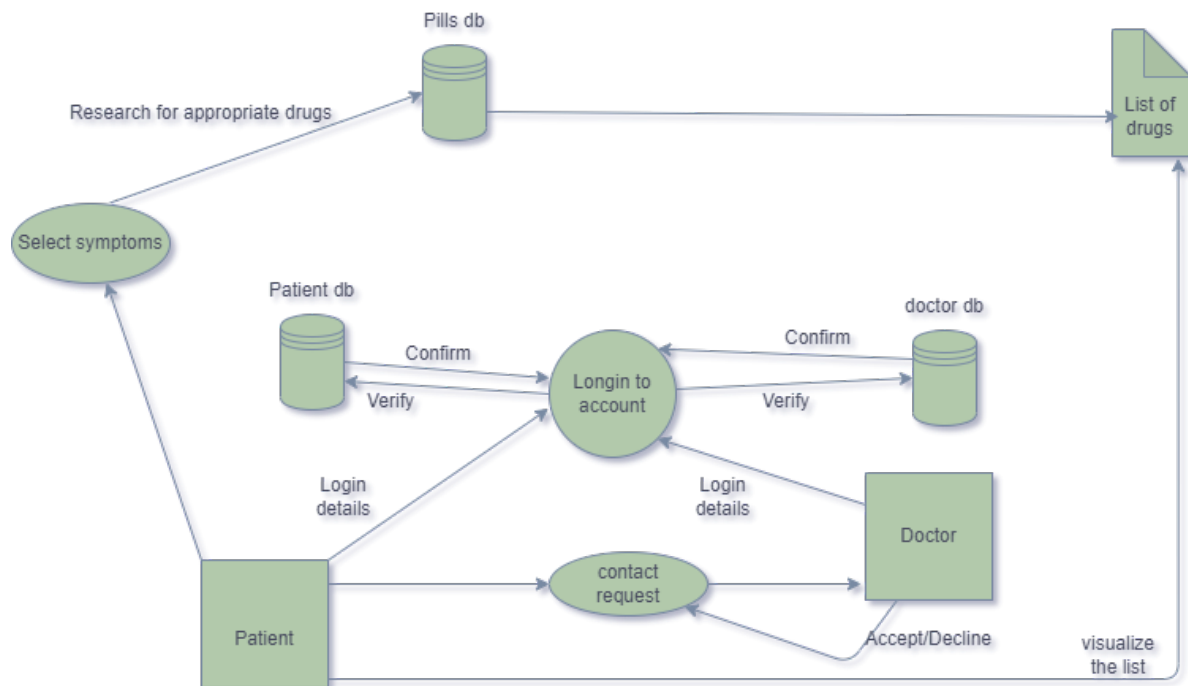
Dominio del problema

En la actualidad existen personas que cuando tiene alguna dolencia en el cuerpo se automedican o acuden a la farmacia y le preguntan al auxiliar de farmacia que medicamento puedo tomar para dolencia en alguna parte de su cuerpo, si la persona toma un medicamento incorrecto ocasionaría resistencia a los antibióticos, enmascaramiento de procesos clínicos graves y consecuentemente retraso en el diagnóstico, dependencia o adicción a los medicamentos, reacciones adversas o efectos secundarios y falta de efectividad.

Pseudocódigo

1. Sea $\text{min} = 0$ y $\text{max} = n-1$.
2. Calcula guess como el promedio de max y min , redondeado hacia abajo (para que sea un entero).
3. Si $\text{array}[\text{guess}]$ es igual a target , entonces detente. ¡Lo encontraste! Regresa guess .
4. Si el intento fue demasiado bajo, es decir, $\text{array}[\text{guess}] < \text{target}$, entonces haz $\text{min} = \text{guess} + 1$.
5. De lo contrario, el intento fue demasiado alto. Haz $\text{max} = \text{guess} - 1$.
6. Regresa al paso 2

Diagrama de sistema



Notación asintótica Big O

Complejidad temporal del algoritmo

Caso promedio

Cuando realizamos la búsqueda binaria, buscaremos en una mitad y descartamos la otra mitad, reduciendo el tamaño del array a la mitad cada vez.

La expresión para la complejidad del tiempo viene dada por la recurrencia.

$$T(n) = T(n/2) + k, \text{ k es una constante.}$$

Este resultado de esta recurrencia da $\log n$, y la complejidad temporal es del orden de $O(\log n)$. Es más rápido que la búsqueda lineal y la búsqueda por salto.

Mejor caso

El mejor de los casos ocurre cuando el elemento del medio es el elemento que estamos buscando y se devuelve en la primera iteración. La complejidad del tiempo en el mejor de los casos es $O(1)$.

Peor caso

La complejidad del tiempo del caso más desfavorable es la misma que la complejidad del tiempo del caso medio. La complejidad de tiempo en el peor de los casos es $O(\log n)$.

- **Complejidad espacial**

La complejidad espacial de este algoritmo es $O(1)$ en el caso de implementación iterativa porque no requiere ninguna estructura de datos más que variables temporales.

En el caso de la implementación recursiva, la complejidad del espacio es $O(\log n)$ debido al espacio requerido por la pila de llamadas recursivas.

Se usará también el algoritmo **b-tree** que es una variación del algoritmo de búsqueda binaria.

El árbol B es logarítmica del número del número de elementos.

Así, las operaciones de búsqueda, inserción y eliminación se pueden implementar en $O(\log n)$ en el peor de los casos, donde n es el número de elementos.

Referencias

<https://www.vidal.fr/recherche/parapharmacie.html?query=maux%20de%20%C3%AAt>

https://www.researchgate.net/profile/A-Marquez/publication/338393411_Algoritmo_MPPT_basado_en_la_Busqueda_Binaria_para_Sistemas_PV_de_Alta_Potencia/links/5e11cf29a6fdcc283757fcc7/Algoritmo-MPPT-basado-en-la-Busqueda-Binaria-para-Sistemas-PV-de-Alta-Potencia.pdf

https://hmong.es/wiki/Binary_search_algorithm

[https://www.delftstack.com/es/tutorial/algorithm/binary-search/#:~:text=Complejidad%20del%20algoritmo%20de%20b%C3%BAsqueda%20binaria,-Complejidad%20del%20tiempo&text=Copy%20T\(n\)%20%3D%20T,y%20la%20b%C3%BAsqueda%20por%20salto](https://www.delftstack.com/es/tutorial/algorithm/binary-search/#:~:text=Complejidad%20del%20algoritmo%20de%20b%C3%BAsqueda%20binaria,-Complejidad%20del%20tiempo&text=Copy%20T(n)%20%3D%20T,y%20la%20b%C3%BAsqueda%20por%20salto)