Árboles Generales 6.5 (96 pdf)

El Futuro se moldea

árboles - est. de datos
- conjunto de nodos y aristas que conectan pares de nodos.

Características de un árbol con raíz:
1) la raíz es 1 nodo
2) todos los nodos excepto la raíz tiene un padre
3) camino único desde raíz hasta cada nodo.
longitud de camino - cant de aristas

hojas - nodos sin hijos

hermanos - nodos con el mismo padre

Ej de SO que los usan: DOS, VMS, Unix

la estructura de un fichero se asemeja a un árbol pero no es un árbol en sí

Operaciones: insertar, buscar y eliminar nodos

árboles de expresiones. Las hojas se evalúan a sí mismas

Data Structure and Algorithms in Java

13.1 Abundance of Digitalized Text

Al buscar un patron de texto en un documento se le llama: "The pattern machine problem" método → "brute-force method"

↓

Poco eficiente, fácil de aplicar

Tamaño muy grande de datos de texto → problema de compresión se resuelve con

↓

"greedy method"

efectivo para este tipo de problema

dynamic programming → técnica de programación usada para resolver problemas difíciles de forma rápida y eficiente en tiempo polinomial

13.3 Tries

* Se preprocesa el texto en lugar del patrón
* útil si el texto es fijo → muchas búsquedas diferentes

Trie: estructura de datos en forma de árbol → guarda cadenas eficientemente

↳ uso ppal → recuperación de información
palabras con cierto prefijo

↳ 2 Operaciones → Búsqueda de palabras completas
→ Búsqueda de cadenas que comiencen con cierto prefijo

13.3.1 Árbol Estándar (Σ: alfabeto)

* Conjunto S de s cadenas hechas con Σ

* Ninguna cadena debe ser prefijo de otra | $G_j = \{car, card, cat\}$ no funcionaría

Características: → c/nodo excepto la raíz tiene una rama de Σ
→ los hermanos tienen ≠ etiquetas (≠ letras) - ordenadas x Σ
→ c/hoja representa una cadena de S
camino raíz → hoja (se obtiene una cadena original)
→ máximo de nodos $n+1$: longitud total

* Si hay cadenas prefijos de otras se puede agregar un carácter especial al final para diferenciarlas

* Cada nodo puede tener entre 1 hijo y $|\Sigma|$ hijos

13.1.1 Notaciones para cadenas de Strings

Ejemplos:

S = "CGAGD" → DNA

t = "http://www.coby la mejor.com" → URL El Futuro se moldea

Un string es un substring de sí mismo.

algoritmos para procesar datos → se ven cadenas de caracteres. Formados por símbolos del alfabeto (Σ), el tamaño puede variar → esto afecta el rendimiento de algoritmos que trabajan con texto.

String → cadenas inmutables

StringBuilder → cadenas mutables

char[] → S[i] es ver de S.charAt(i) → + simple

Subcadena → parte de cadena P[i...j]

Si i > j → cadena vacía (longitud 0)

Subcadena → Prefijo: índice empieza en 0 P[0...j]

→ Sufijo: termina en el último índice P[i...n-1]

"La cadena completa es una subcadena de sí misma"

* una subcadena puede ser toda la cadena

* una subcadena propia es ≠ a la cadena completa

* La cadena vacía es tanto un prefijo como un sufijo

* 2 operaciones de los string: modifica cadena propia y devuelve información, sin modificarlo.

* Un trie almacena de forma eficaz los prefijos comunes entre cadenas, desde el nicen hasta donde son ≠

* cuanto más profundo estamos en el árbol, las probabilidades de que más cadenas comparta el mismo prefijo disminuya.

* Un nodo puede tener 0, 1 o más hijos

Altura: longitud de la cadena más larga de sus cadenas

Número de nodos: num máximo es $n+1$, n: longitud total de todas las cadenas en S

Hojas: tiene s hojas, s: número de cadenas del conjunto S

* Peor caso: ninguna cadena comparte prefijo → Similar a lista enlazada c/nodo tiene 1 sólo hijo

* Para buscar seguimos los caracteres de S, si termina en una hoja X o una cadena de S, si termina en un nodo interno X no está en S.

* Orden de tiempo de ejecución $O(m.|\Sigma|)$ → Visitamos como máximo $m+1$ nodos
c/nodo podemos $O(|\Sigma|)$

* Si usamos una tabla secundaria de búsqueda o tabla hash en c/nodo el tiempo para encontrar a su hijo puede ser $O(\log|\Sigma|)$ o $O(1)$ con una tabla tamaño $|\Sigma|$ si $|\Sigma|$ es pequeño como en las cadenas ADN

* longitud m es $O(m)$

word matching - verifica si un patrón coincide con una palabra completa en un texto. No coincidencias en subcadenas

construcción - insertar las cadenas una a una, al insertar un x se sigue el camino, si no se encuentra entonces se agregan los nodos necesarios.

inserción: $O(m \cdot |\Sigma|)$ m: longitud de la cadena, $|\Sigma|$: tamaño del alfabeto } con tablas Hash → $O(m)$

* construcción completa: $O(n)$ n: longitud total de las cadenas

Problema de nodos sólo c/1 hijo → ineficiente → solución: - Trie comprimido
- Patricia Trie

2 partes de un word matching usando Trie:

1) Texto original → keywords - las que se buscan en el trie

2) Trie estándar → c/hoja tiene una lista de posiciones donde la palabra comienza en el texto original

Se eliminan las stop words: a, an, the

13.3.2 Árboles Comprimidos

Es como un árbol estándar pero se asegura de que c/nodo interno tenga al menos 2 hijos. Se elimina el desperdicio de espacio.

* cant nodos proporcional a cant de strings
* Si un nodo (# raíz) tiene 1 solo es redundante

El Futuro se moldea

* Si hay una cadena de redundantes se comprime en un solo borde

Ej: S → t → o → c → k

→ "Stock"

Este se etiqueta con todos los caracteres concatenados entre separadores.

trie estándar → c/nodo 1 sola letra
trie comprimido → c/nodo secuencia de letras

Tamaño de un trie comprimido: cant de cadenas

Propiedades → c/nodo interno tiene de 2 a d hijos (d = tamaño del alfabeto)
→ S hojas, 1 x c/palabra
→ número de nodos proporcional a S, O(s)

* No siempre conviene usarlos → etiquetas + largas = + espacio

* + útiles como estructuras auxiliares para búsquedas rápidas, no para almacenar todos los caracteres.