Tecnológico de Monterrey.

TC2038. Análisis y diseño de Algoritmos A

M.C. Ramona Fuentes Valdéz

rfuentes@tec.mx

Grafos

Trie

Introducción

Cuando se utilizan números como valores llaves, tenemos elementos de datos de tamaño constante y se pueden comparar en tiempo constante.

En algunas aplicaciones reales, el procesamiento de texto es más importante que el procesamiento de números.

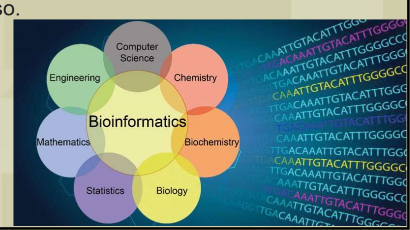
Necesitamos estructuras diferentes para cadenas que para llaves numéricas.

Importancia

- Ejemplo: 112 < 467, comparación numérica en O(1).
- Comparar cadenas lexicográficamente no refleja la similitud de cadenas. Por ejemplo: Western > Easter, es una comparación de cadenas en O(min (|s1|, |s2|)). donde |s| denota la longitud de la cadena *s*.
- Los segmentos de texto tienen una longitud; no son objetos elementales que la computadora pueda procesar en un solo paso.

Aplicaciones

- ✓ Bioinformática (*Datos de secuencias de ADN / ARN o proteínas*).
- ✓ Motores de búsqueda.
- ✓ Corrector ortográfico.



Grafos

Trie

Antecedentes

La herramienta básica para estructuras de datos de cadenas, similar en función al árbol de búsqueda binaria balanceada, se llama "trie“.

Derivar de "recuperación". (Se pronuncia try o tree).

En este árbol, los nodos no son binarios. Contienen potencialmente un arco saliente para cada carácter posible, por lo que el grado es como máximo el tamaño del alfabeto |A|.

Prefijo vs Sufijo

Ejemplo: "computadora“

- Prefijo: (c, co, com)
- Sufijo: (a, ra, ora)

Cada nodo de esta estructura de árbol corresponde a un prefijo de algunas cadenas del conjunto.

Si el mismo prefijo aparece varias veces, solo hay un nodo para representarlo.

La raíz de la estructura de árbol es el nodo correspondiente al prefijo vacío.

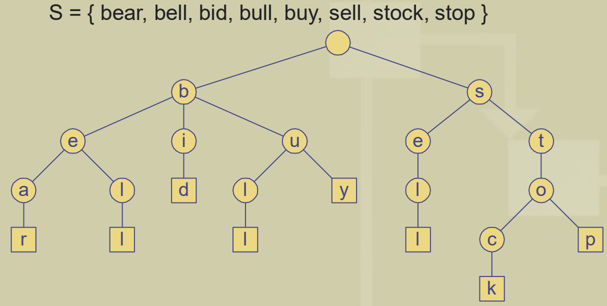
Grafos

Trie

Prefijo vs Sufijo

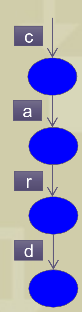
Ejemplo: Trie estándar para el conjunto de cadenas:

S = { bear, bell, bid, bull, buy, sell, stock, stop }



Prefix-free

- ¿Qué pasa si una cadena "completa" es un prefijo de otra cadena?
  - car
  - card



Grafos

Trie

String Termination

Los *strings* son secuencias de caracteres de algún alfabeto. Pero para su uso en la computadora, necesitamos información adicional importante: ¿cómo reconocer dónde termina la cadena?

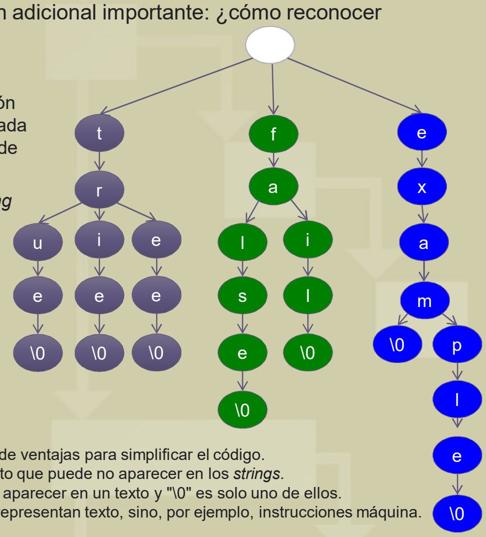
Para ello:

- Podemos tener un carácter de terminación (*explícito*), el cual se agrega al final de cada *string*, pero puede que no ocurra dentro de la cadena "\0" (código ASCII 0).
- Podemos almacenar junto con cada *string* su longitud.

Ejemplo:

Strings:

- exam
- example
- fail
- false
- tree
- trie
- true



El uso del carácter de terminación especial "\0" tiene una serie de ventajas para simplificar el código.
Tiene la desventaja de tener un carácter reservado en el alfabeto que puede no aparecer en los *strings*.
Hay muchos códigos ASCII no imprimibles que nunca deberían aparecer en un texto y "\0" es solo uno de ellos.
También hay muchas aplicaciones en las cuáles los strings no representan texto, sino, por ejemplo, instrucciones máquina.

Grafos

Trie

Find

Para realizar una operación de búsqueda en esta estructura:

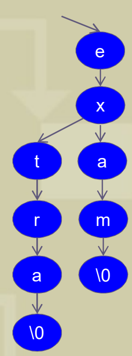
- Comience en el nodo correspondiente al prefijo vacío.
- Lea la cadena de consulta (query), siguiendo para cada carácter leído el apuntador saliente correspondiente de ese carácter al siguiente nodo.
- Después de leer la cadena de consulta, llegamos a un nodo que corresponde a esa cadena como prefijo.
- Si la cadena de consulta está contenida en el conjunto de cadenas almacenadas en el trie, y ese conjunto no tiene prefijos, entonces este nodo pertenece a esa cadena única.

Insert

Para realizar una operación de inserción en esta estructura:

- Realizar búsqueda.
- Cada vez que encontramos un apuntador nulo, creamos un nuevo nodo.

Ejemplo: Insertar "extra"



Delete

Para realizar una operación de eliminación en esta estructura:

- Realizar búsqueda.
- Elimina todos los nodos de la ruta desde "\0" hasta la raíz del árbol (*a menos que lleguemos a un nodo con más de 1 hijo*).

Ejemplo: Elimina "extra"

Grafos

Trie

Tamaño del alfabeto

El problema que se presenta es la dependencia del tamaño del alfabeto la cual impacta en la cantidad de nodos.

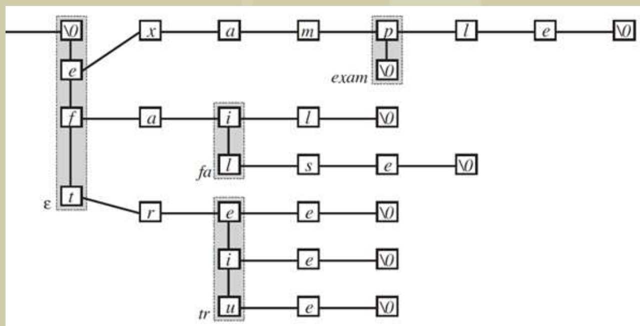
Hay varias formas de reducir o evitar el problema del tamaño del alfabeto.

- Un método simple es reemplazar los nodos grandes por listas enlazadas de todas las entradas que son realmente utilizadas.

Ejemplo: Listas

Trie de los strings:

- exam
- example
- fail
- false
- tree
- trie
- true



Grafos

Aplicaciones de grafos

Árbol de extensión mínima (*Minimum Spanning Tree*).

- Este algoritmo obtiene el costo mínimo en que se puede incurrir para conectar a todos los Nodos del Grafo. Procurando tener la menor cantidad posible de Arcos.

Árbol del camino más corto (*Shortest Path*).

- Calcula cuál es la trayectoria más corta que puede existir de un Nodo a cualquier otro Nodo del Grafo.

Estas dos aplicaciones se utilizan para Grafos Ponderados, principalmente.

Grafos

Árbol de extensión mínima (*Minimum Spanning Tree*)

- Es un subgrafo conectado que contiene a todos los nodos del grafo original y que minimiza la suma de todos los valores de los Arcos involucrados.
- El algoritmo utiliza una cola priorizada para obtener el Arco de MAYOR prioridad (el de MENOR valor).

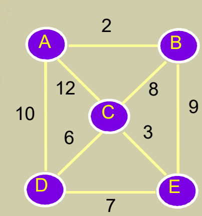
Algoritmo

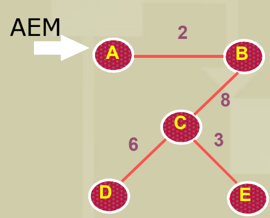
* Crear un grafo vacío (AEM – Árbol de Extensión Mínima).
* Insertar cualquier Nodo del grafo al AEM.
* Insertar en la Cola Priorizada a todos los Vecinos.
* Mientras la Cola Priorizada no se vacíe y NO se hayan insertado todos los Nodos al AEM:
- Sacar un Nodo de la Cola Priorizada (> prioridad)
- Si el Nodo no está aún en el AEM:
  . Agregarlo al AEM con su correspondiente arco.
  . Meter a sus Vecinos a la Cola Priorizada.

Grafos

Árbol de extensión mínima (*Minimum Spanning Tree*)

Ejemplo:





~~B~~	C	D	~~C~~	E	~~E~~	~~D~~	D
~~2~~	12	10	~~8~~	9	~~3~~	~~6~~	7

Cola Priorizada

Grafos

Algoritmos

Algoritmo de extensión mínima

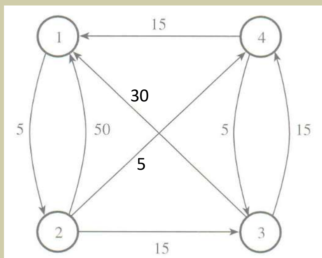
- Se desea calcular la longitud del camino más corto entre cada par de nodos.
- Para establecer la matriz de adyacencias se utiliza:

  $L[i,i]=0 \quad 1 \leq i \leq n$

  $L[i,j]\geq 0 \quad \forall\ i,j \in A$ (hay arista de i a j)

  $L[i,j]=\infty \quad$ Si $i,j \notin A$ (no hay arista de i a j)

Ejemplo:



L[4][4]={{ 0, 5, ∞, ∞},
{ 50, 0, 15, 5},
{ 30, ∞, 0, 15},
{ 15, ∞, 5, 0}};

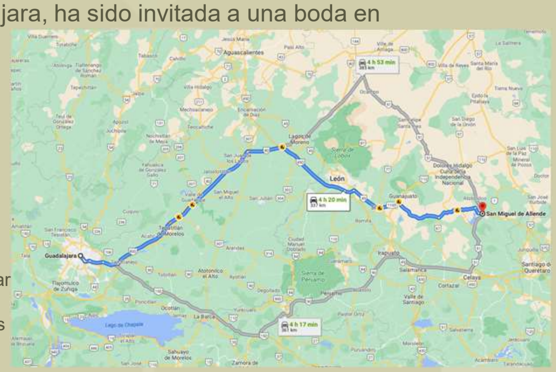
Grafos

Árbol del camino más corto

Aplicaciones

Mary, quién vive en Guadalajara, ha sido invitada a una boda en San Miguel de Allende.

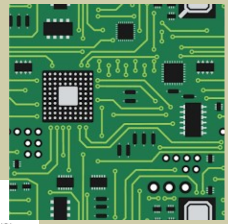
Dado un mapa de carreteras de México en el que está marcada la distancia entre cada par de intersecciones adyacentes, ¿cómo puede determinar la ruta más corta?



Posible solución:

- Una forma posible sería enumerar todas las rutas de Guadalajara a San Miguel de Allende, sumar las distancias en cada ruta y seleccionar la más corta.
  - Sin embargo, es fácil ver que incluso si rechazamos las rutas que contienen ciclos, Mary tendría que examinar una enorme cantidad de posibilidades, la mayoría de las cuales simplemente no vale la pena considerarlas.
    - Por ejemplo, una ruta de Guadalajara a San Miguel de Allende que pasa por Culiacán es obviamente una mala elección, porque Culiacán está a varios cientos de millas de distancia.

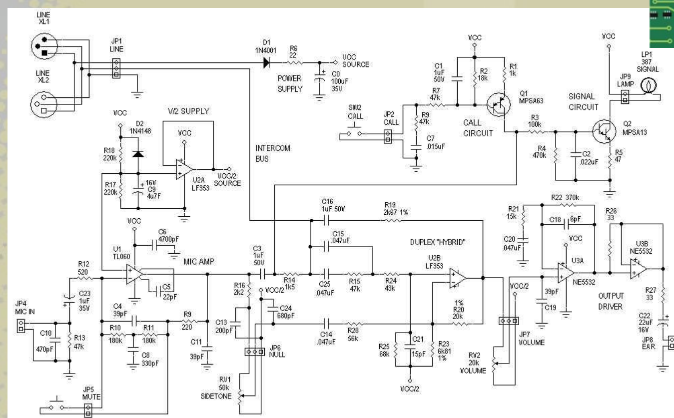
Grafos



Árbol del camino más corto

Otras aplicaciones

Diseño de circuitos.



Grafos

Árbol del camino más corto

- Es un Subgrafo Conectado que contiene cuál es la trayectoria óptima que puede seguirse de un Nodo para llegar a cualquier otro Nodo del Grafo.
- Utiliza el mismo algoritmo que el Árbol de Extensión Mínima, sólo que ahora la prioridad está dada por la suma de los costos desde el Nodo origen hasta el Nodo que actualmente se analiza.

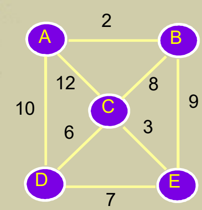
¿Qué se modela?

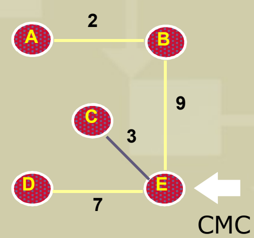
Los pesos de las aristas pueden representar otras métricas distintas a las distancias, por ejemplo: tiempo, costo, penalizaciones, pérdidas o cualquier otra cantidad que se acumule linealmente a lo largo de un camino o ruta y que se desee minimizar.

Grafos

Árbol del camino más corto

Ejemplo:





Cola Priorizada

~~EC 3~~ ~~ED 7~~ ~~EB 9~~ ~~ECA 15~~ ~~ECB 11~~ ~~ECD 9~~ ~~EDA 17~~ ~~EBA 11~~

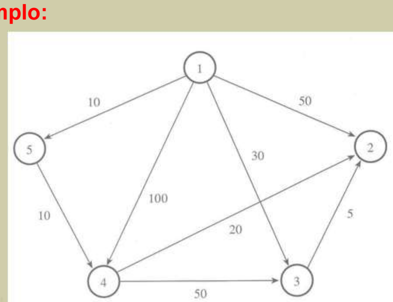
Grafos

Algoritmos

Algoritmo de caminos mínimos

- Considere un grafo dirigido G=<N,A> (N=nodos, A=aristas dirigidas).
- Cada arista posee una longitud no negativa.
- Uno de los nodos se considera un nodo origen.
- Por sencillez, asume que los nodos están numerados (N={1,2,..,n}) y que existe una matriz L que guarda la longitud de todas las aristas dirigidas: $L[i,j] \geq 0$ si la arista $(i,j) \in A$ y $L[i,j]=\infty$ en caso contrario.

Ejemplo:



L[5][5]=
{{∞, 50, 30,100, 10},
{∞, ∞, ∞, ∞, ∞},
{∞, 5, ∞, ∞, ∞},
{∞, 20, 50, ∞, ∞},
{∞, ∞, ∞, 10, ∞}};

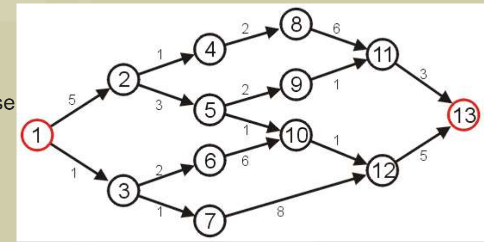
Grafos

Algoritmos

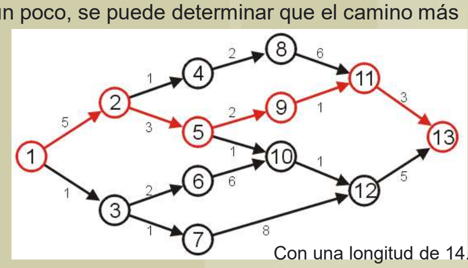
Algoritmo de camino más corto

Ejemplo:

Dado el siguiente grafo, suponer que se desea encontrar el camino más corto desde el vértice 1 al vértice 13.



Después de analizar un poco, se puede determinar que el camino más corto es el siguiente:



Con una longitud de 14.

Existen otros caminos, pero son más largos.

Grafos

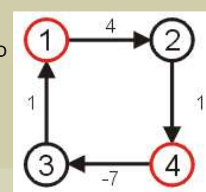
Algoritmos

Algoritmo de camino más corto

Ciclos negativos

Si se tienen vértices negativos, es posible que se termine en un ciclo en el que cada paso a través del ciclo se disminuye la longitud total.

Por lo tanto, la longitud más corta es indefinida para un grafo así.



Ejemplo:

- Considera el camino más corto desde el vértice 1 al 4.

Solo se considerarán ponderaciones no negativas.