Similitud de Secuencias: de un Par a Todas contra Todas

Ricardo Baeza-Yates
Centro de Investigación de la Web
Depto. de Ciencias de la Computación
Universidad de Chile
ricardo@baeza.cl

Resumen

- Similitud y alineamiento de secuencias
- Programación Dinámica
- Heurísticas de Filtrado (e.g. Blast)
- Árboles y arreglos de sufijos
- Conclusiones

Metas Contrapuestas

- Biólogo: 'Encuentre un algoritmo eficiente que resuelva mi problema.'
- Informático: 'Déme un problema que pueda resolver en forma eficiente.'
- Choque de culturas: ¿Qué pasa cuando ninguno de los dos casos es posible?

Estructura Similar $\rightarrow$ Secuencia Similar

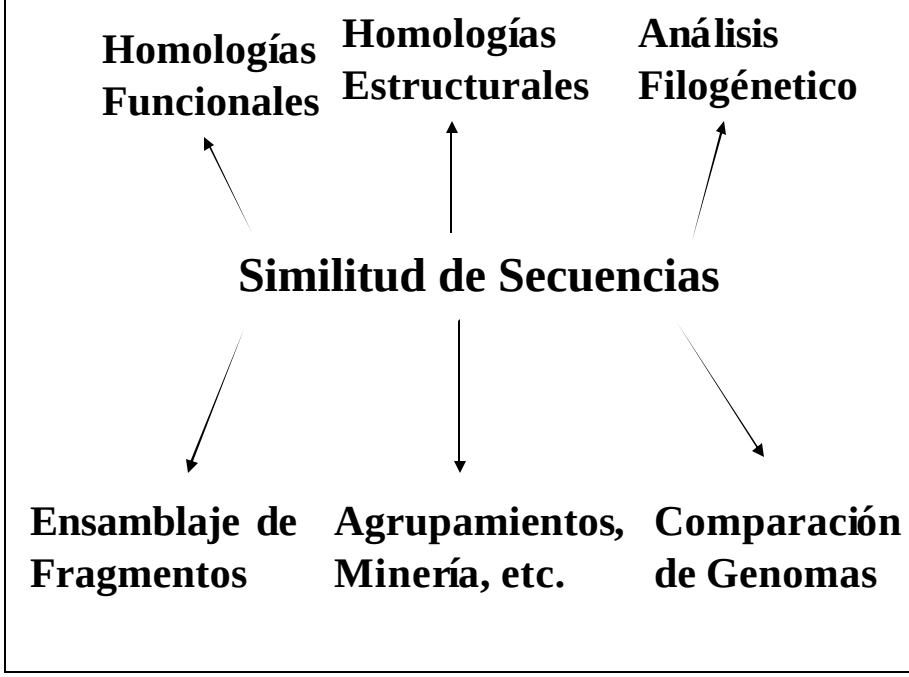
- Lo inverso no es cierto
- Evolución convergente: soluciones similares a problemas similares pueden ser internamente muy diferentes.
- Ejemplos: oso y oso panda, corvina y delfín, un murciélago y un pájaro, etc.
- Lo mismo es cierto entre especies moleculares y anatomías

Secuencia → Función

- Secuencias similares a veces tienen funciones similares
- 'Los mismos genes que funcionan en moscas son los mismos que funcionan en humanos.'

  -- Eric Wieshaus

  Nobel por su trabajo en drosófilas (1995)



¿Qué es un Alineamiento?

- *Alinear* estas dos secuencias en forma óptima

```
GACGGATT
GATCGGTT
```

- Definir en forma *precisa* que es un alineamiento

Definición de Alineamiento

- Insertar espacios tal que las letras iguales calcen o se alineen con espacios:

```
GA-CGGATT
GATCGG-TT
```

- No permitir que los espacios se alineen
- Aceptar espacios incluso al comienzo o el final

```
GCAT-
-CATG
```

Definición de Similitud

- Dado un alineamiento, se calcula una *medida de similitud*
- Tres posibilidades en cada columna:
  - calce letra-letra
  - error letra-letra
  - error letra-espacio (o viceversa)

Alineamiento Óptimo

- Crear función de similitud
- Convencionalmente:

  +1 punto por cada calce

  -1 castigo por error letra-letra

  -2 castigo por error letra-espacio

Soluciones de Alineamiento

- Búsqueda secuencial exhaustiva usando programación dinámica
- Filtro eficiente más búsqueda secuencial
  - Blast, Fasta, etc.
- Soluciones en base a índices especiales
  - Arboles y arreglos de sufijos
  - Estructuras basadas en q-gramas
- Métodos híbridos

Solución usando Programación Dinámica

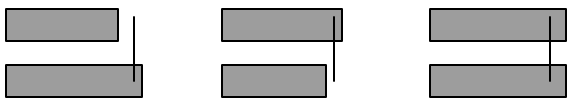
- Programación dinámica: usar la solución de subproblemas más pequeños para calcular problemas más grandes
- Dadas dos secuencias *s,t* de largo *m,n (m£n)*
- Construir los alineamientos óptimos de los prefijos (inducción)
- ¿Caso base?
- ¿Relación de recurrencia?

Recurrencia

- Dado un alineamiento óptimo de prefijos de *s*,*t* más cortos que *i*,*j*, encontrar el óptimo de *s*[*1..i*], *t*[*1..j*]
- Tres posibilidades:
  - extender *s* en una letra, *t* en un espacio
  - extender *s* y *t* en una letra
  - extender *s* en un espacio, *t* en una letra

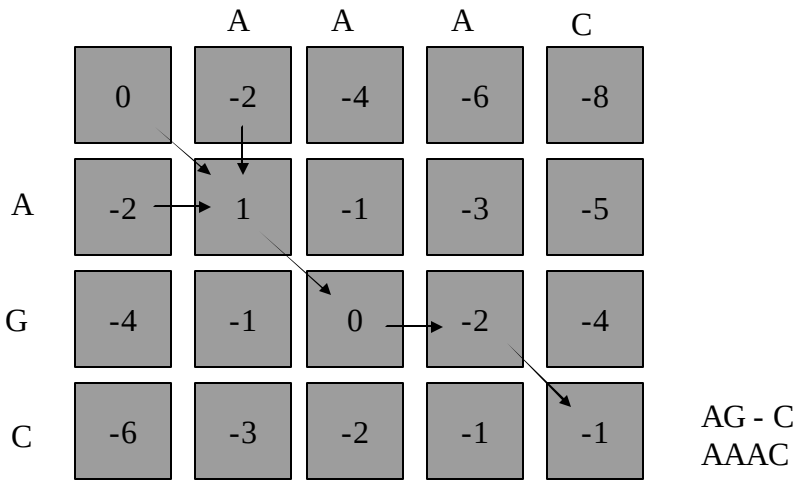
Recurrencia (cont.)

- $Sim(i,j) = max(\ Sim(i-1,j)-2,\ Sim(i,j-1)-2,\ Sim(i-1,j-1) + Score(s_i,t_j)\ )$



- $Score(si,tj) = 1$ si $s_i = t_j$, -1 si no
- En general, *Score* puede ser distinto para cada par de letras y también el costo de un espacio puede variar

Ejemplo Simple



Detalles

- ¿Cuál es el mejor orden para llenar la matriz?
- ¿Cómo encontramos el alineamiento?
- ¿Qué hacemos si hay empates?
- ¿Cuánto espacio se necesita?

  min(m,n)
- ¿Cuánto tiempo se ocupa?

  O(nm)

¿Cuando es la Similitud Interesante?

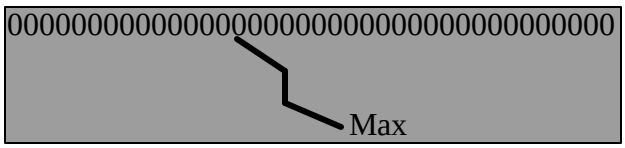
- LCS: Subsecuencia común más larga de dos secuencias
- LCS Promedio de dos secuencias aleatorias

Alfabeto binario: 81%
ADN: 65%

- Por esto las similitudes interesantes son altas

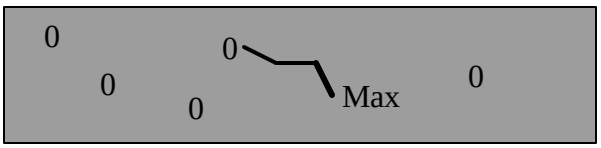
Alineamiento Semiglobal

- El alineamiento no siempre tiene que ser global, por ejemplo si *m* es mucho menor que *n*
- Alineamiento semiglobal: los gaps en los bordes valen 0
  - La primera fila es todo ceros
  - Basta calcular el valor máximo de la última fila
  - No consideramos los espacios a la derecha



Alineamiento Local

- Buscamos el subsegmento de *t* más parecido a un subsegmento de *s*
  - Le agregamos un cuarto caso a la recurrencia, un calce vacío (score igual a 0)
  - Luego buscamos el valor máximo en toda la matriz
  - Es decir, podemos comenzar y terminar en cualquier parte



Ejemplos

- *s* = CAGCACTTGGATTCTCGG (*n=18*),
  t = CAGCGTGG (*m=8*)

```
CAGCACTTGGATTCTCGG
CAGC-----G-T----GG
```
Global

```
CAGCA-CTTGGATTCTCGG
---CAGCGTGG--------
```
Semiglobal

```
CAGCACTTGGATTCTCGG
     CGTGG
```
Local

El Castigo de los Espacios (gaps)

- El modelo supone que dos espacios de tamaño 1 son equivalentes a un espacio de tamaño 2
- Es un modelo lineal: $gap(k) = bk$
- ¿Es realista? ¿Por qué no?

Funciones Genéricas para Gaps

- Alineamientos ya no pueden calcularse como la suma de sus partes
- Aún son la suma de *bloques* con una letra alineada o un gap en cada una
- Los bloques son: letras alineadas, s-gap, t-gap

```
A|A|C|---|A|GAT|A|A|C
A|C|T|CGG|T|---|A|A|T
```

PD para Gaps Genéricos

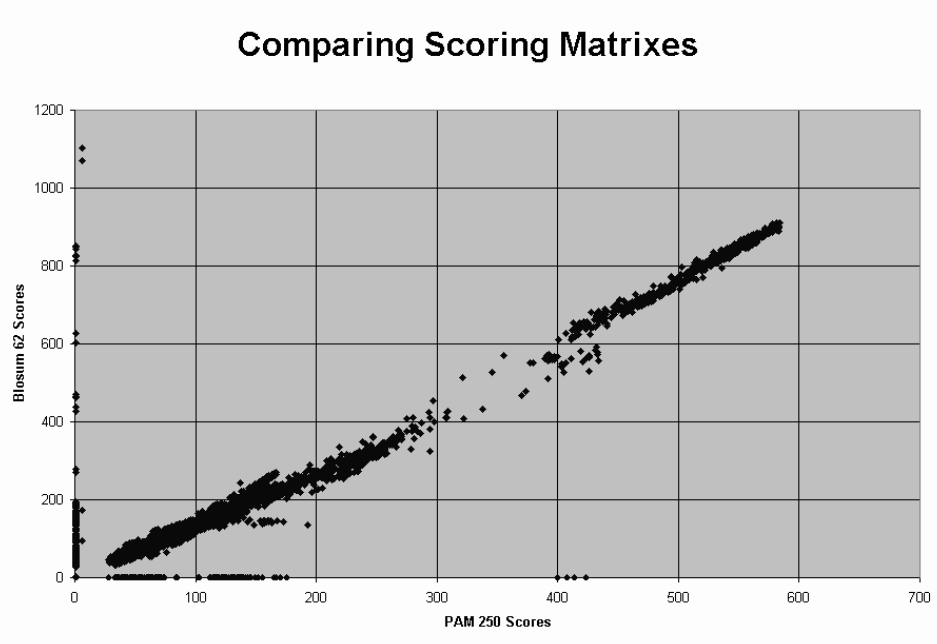
- Se necesitan tres matrices, una para cada tipo de bloques
- La complejidad es $O(n^2m)$
- Es prohibitivo para problemas grandes
- Funciones afines permiten mantener el tiempo cuadrático: $gap(k) = bk + c$

¿Es Alineamiento Múltiple una Generalización?

- Desde el punto de vista computacional, si. Pasamos de dos secuencias a muchas secuencias
- Si, pues ayuda a detectar similitudes débiles o falsas biológicamente
- No, ya que vamos de similitud biológica conocida a similitud probable de secuencias
- Comparación múltiple de secuencias (MSC)

Medidas de Similitud

- **Distancia de Hamming (1951):** número de posiciones donde las secuencias son diferentes
- **Distancia de Levenshtein (1961):** el número mínimo de inserciones, cambios y borrados de caracteres para transformar una secuencia en la otra
  (**Damerau:** trasposiciones valen lo mismo)
- **"Matrices de Substitución (PAM)" (Dayhoff, 1978):** probabilidades de substituir un caracter por otro
- **Alineamiento con "Gaps" (Varios autores, 1980+)**
- **Modelos Estadísticos (Varios autores, 1982+)**



Heurísticas de Filtrado

- Detectar posibles candidatos
- Heurísticas pueden ser con pérdida o sin pérdida
- Ejemplo: BLAST es con pérdida
- Pero sigue siendo *O(nm)* con una constante menor

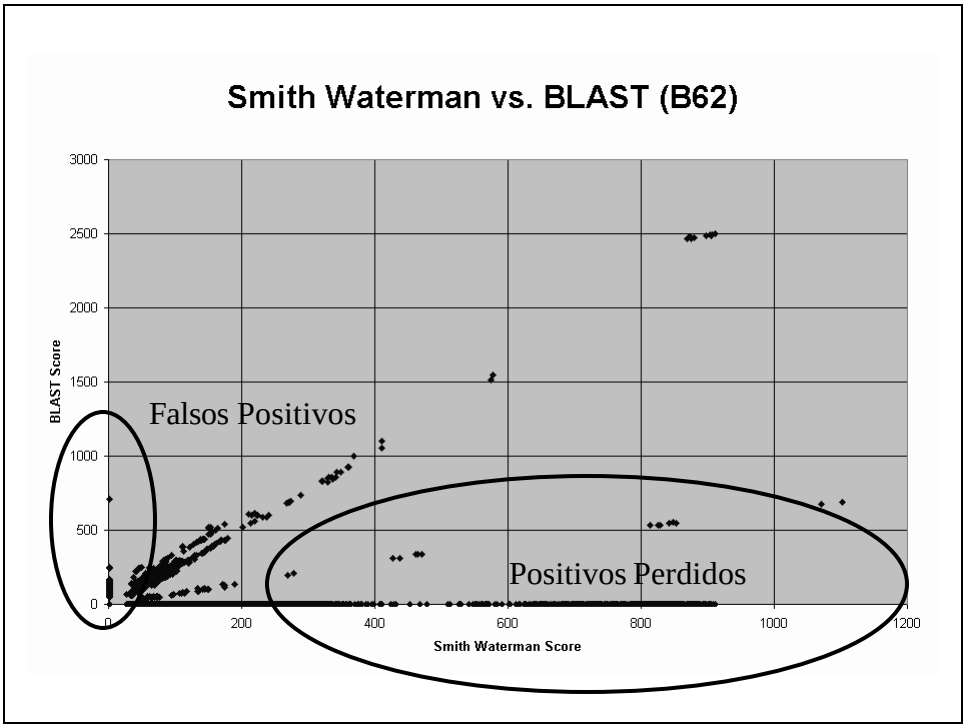
Heurística de BLAST

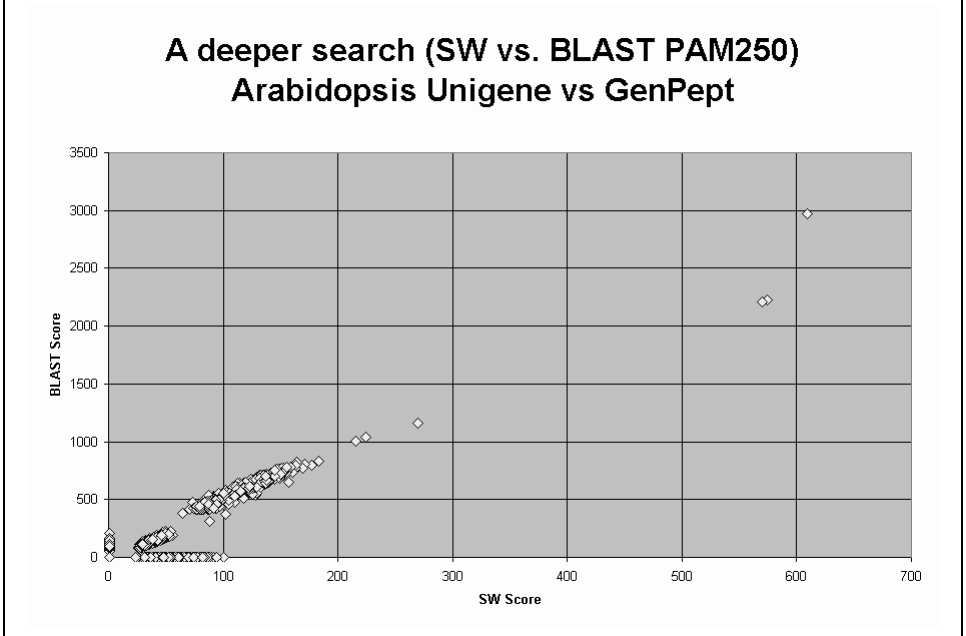
- Un candidato potencial debe contener un segmento (MSP) sobre un cierto umbral de similitud

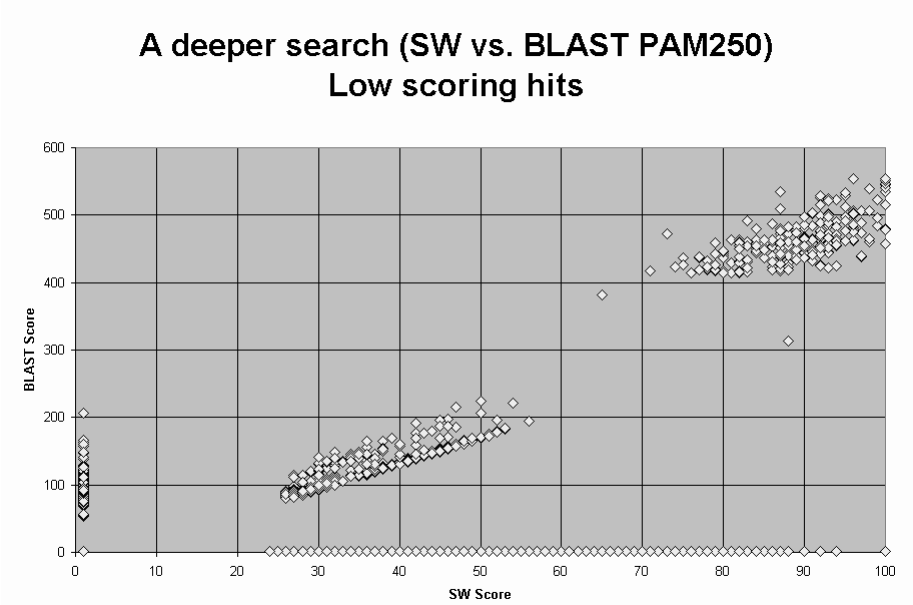
```
ca AACTGCTG aacgttgtcgtgagttctggctgcta--
--AACTGCTG ggctctc-----ccgatcggctggcaaa
```

"BLAST's great utility is for finding high scoring MSPs quickly." Altschul et. Al.; 1990

- Experimentos de C. Dwan (2002)


Smith Waterman vs. BLAST (B62)
BLAST Score
Smith Waterman Score
Falsos Positivos
Positivos Perdidos
0
500
1000
1500
2000
2500
3000
0
200
400
600
800
1000
1200

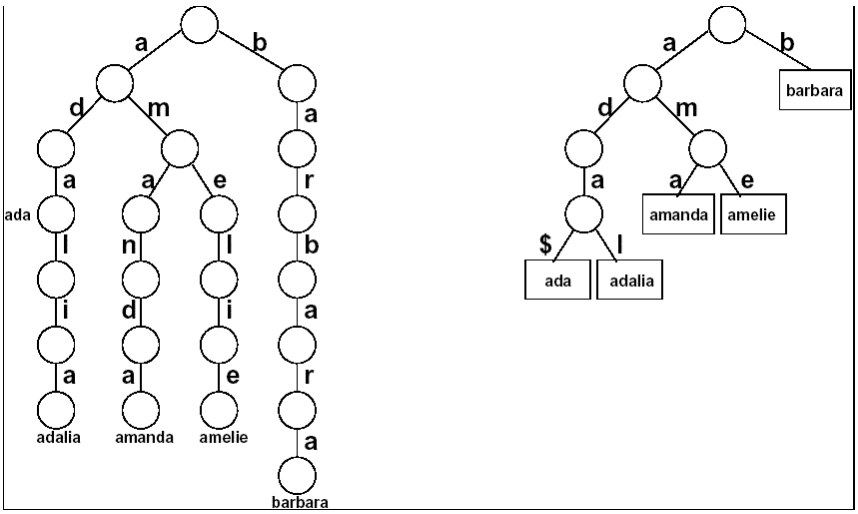

A deeper search (SW vs. BLAST PAM250)
Arabidopsis Unigene vs GenPept
BLAST Score
SW Score
0
500
1000
1500
2000
2500
3000
3500
0
100
200
300
400
500
600
700



Algoritmos de Alineamiento

- **(Needleman & Wunsch, 1970):** PD global
- **SW (Smith & Waterman, 1981):** PD local para matrices PAM
- **FASTA (Pearson et al. 1988):**
- **BLAST (Altschul et. al. 1990):** Heurística aproximada de SW
- **Gapped BLAST (Altschul et al. 1997)**: Mejoras
- ¿Se puede hacer mejor?

Árboles Digitales



Búsqueda

- Seguir las ramas que tienen los caracteres de la palabra buscada
- Búsqueda es ***independiente*** del tamaño del texto (*O(m)*)

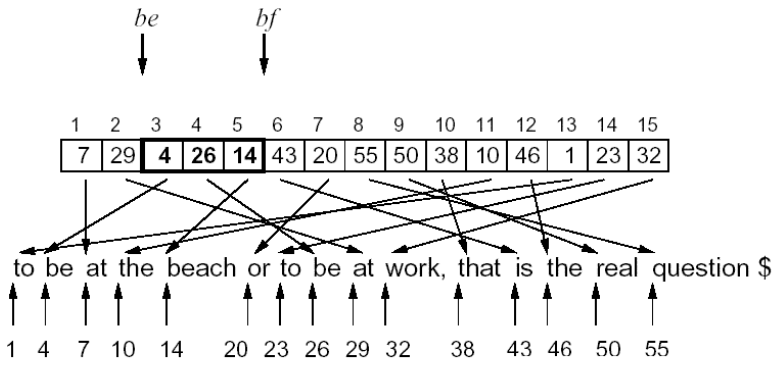
- Problema: espacio (> 10 veces el texto)

Árboles y Arreglos de Sufijos

- Árbol digital de todos los sufijos de un texto
- Las hojas son los sufijos

- Arreglo de sufijos: mantener sólo las hojas
- Espacio es 4 veces el texto

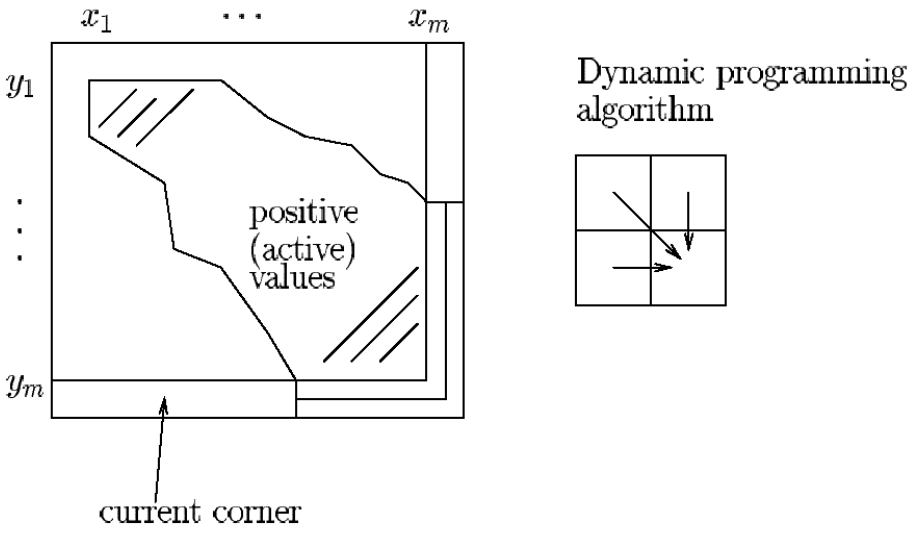
Búsqueda

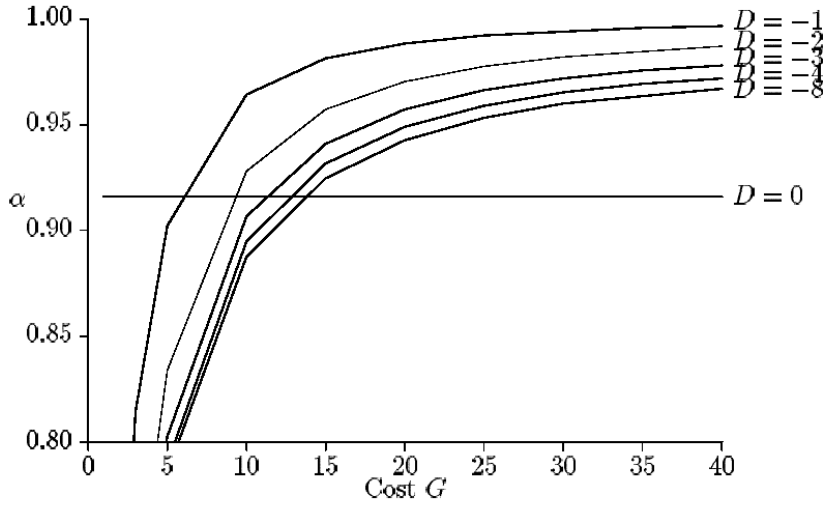


Factor extra de *log(n)*
Número de resultados es gratis

Búsqueda indexada

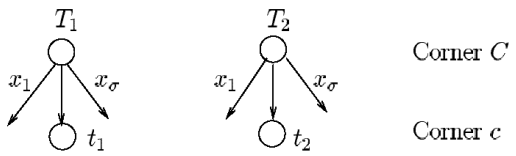
- Construir arreglo de sufijos de la BD (*O(n)*)
- Buscar la secuencia patrón en el arreglo simulando la programación dinámica
- Por cada nodo se mantiene una matriz de valores activos
- Tiempo promedio es $O(n^a m \log(n))$
- El exponente depende del umbral G y el costo de un gap D





Todos contra Todos

- Se simula la PD del árbol de sufijos consigo mismo
- Hay pares de nodos activos (de cada copia del árbol)



- El tiempo es $O(n^{1+a} m \log(n))$
- Fuerza bruta sería $O(n^2 m^3)$

```
Push( ActiveSet, [Root, Root, ZeroCorner] );
while ActiveSet ≠ Empty do
{
    [T1, T2, C] ← Pop( ActiveSet );
    for every non null subtree t1 ∈ T1 do
        for every non null subtree t2 ∈ T2 do % OPT
        {
            c ← ExpandCorner( C, Symbol(t1), Symbol(t2) );
            if Cost(c) ≥ G then      % goal reached
            {   % if is not the same sequence
                if Size(t1) > 1 or t1 ≠ t2 then
                Sols ← Sols ∪ {[t1, t2, c]};
            }
            else if Cost(c) > 0 then
                Push( ActiveSet, [t1, t2, c] );
            % otherwise the node pair is discarded
        }
}
return( Sols );
```

Conclusiones

- Existen algoritmos mejores que se usan poco (Darwin, Gonnet 1990)
- Software vs. Hardware
- BLAST no encuentra todo, podemos perder soluciones
- Todos con todos: la cantidad de resultados potenciales es cuadrático
- Minería de soluciones

¿Preguntas?

Invitaciones:

- Minería de Consultas en la Web
  Mañana sábado 10AM
  Auditorio Depto. Matemática, UnB

- ACM SIGIR 2005 Information Retrieval Conf.
  Salvador, Brasil, Agosto
  UFMG/U. de Chile