



Informe Proyecto 3 Análisis de Algoritmos

Integrantes:

Linna Cao Merino

Bastián Gómez Jara

Jesús Gómez Zambrano

Profesora:

Cecilia Hernández Rivas

1a) Los subproblemas que se encuentran en este algoritmo se pueden dividir en 3, se utiliza la tabla para maximizar el valor actual de la casilla ocupada con 1 de las 3 posibles soluciones, para esto se escoje el puntaje mayor entre las casillas de la derecha menos el valor del gap, la de abajo menos el valor del gap o finalmente el valor de la casilla diagonal abajo-derecha a la cual se le suma o resta un valor dependiendo de si los caracteres de esa casilla son iguales o no.

Y su recurrencia seria:

$\max(\text{reglasrec}(i-1,j,s1,s2)-\text{gap}, \text{reglasrec}(i,j-1,s1,s2)-\text{gap}, \text{reglasrec}(i-1,j-1,s1,s2)+\text{match})$

si hay un match de las letras

o

$\max(\text{reglasrec}(i-1,j,s1,s2)-\text{gap}, \text{reglasrec}(i,j-1,s1,s2)-\text{gap}, \text{reglasrec}(i-1,j-1,s1,s2)-\text{notmatch})$

si no coinciden las letras

1d) Análisis de espacio del algoritmo bottom-up con tabulación:

Para nuestra implementación del código de comparación de secuencias, se necesita 3 estructuras "caras" en términos de espacio, ya que omitimos las funciones baratas como comparaciones, sumas y restas.

Las estructuras que utilizamos son 1 matriz bidimensionales de dimensiones $n*m$, siendo n el largo de la primera secuencia y m el largo de la segunda, la cual es una matriz de enteros y luego utilizamos 2 stack de caracteres con un tamaño máximo igual al largo de la secuencia más larga.

Si analizamos los espacios utilizando cotas superiores tenemos que la matriz de enteros utiliza $O(n*m)$ Y si asumimos que n es de largo mayor, la cota superior de los stack será de $O(n)$.

Por lo anterior podemos decir el análisis de espacio de este algoritmo esta dado por la cota superior de $O(n*m)$.

Análisis de tiempo del algoritmo bottom-up con tabulación:

Si analizamos nuestra implementación, tenemos que analizar 2 funciones, la función reglas que asigna los puntajes en una tabla, y optimo, que encuentra la secuencia óptima para el problema.



La función regla tiene 1 ciclo iterativo for, que itera de 1 hasta m, anidado en otro ciclo for, que itera desde 1 hasta n, por lo que podemos decir que la cota superior de esta función es $O(n*m)$. En cambio, la función optimo utiliza un ciclo while que itera mientras i y j, que representa n y m, no sea ninguno 0, por ende, itera hasta que 1 de los 2 sea 0, si sabemos que las secuencias en este punto tienen igual largo, decimos que la cota superior de esta función es de $O(n)$.

Por lo anterior, decimos que el análisis de tiempo para esta implementación esta dado por la cota superior $O(n*m)$.

Análisis de espacio del algoritmo top-down con memorization:

Para nuestra implementación del código usando top-down con memorization, al igual que con bottom up, solo se utiliza 1 matriz de enteros de dimensiones $n*m$ y 2 stacks de caracteres, que, si asumimos que n es de mayor largo, de tamaño n.

El análisis de espacio para esta implementación se aproxima bastante a la de bottom-up, solo difiriendo realmente en las operaciones de costos bajo y variables enteras.

Por lo anterior podemos decir el análisis de espacio de este algoritmo esta dado por la cota superior de $O(n*m)$.

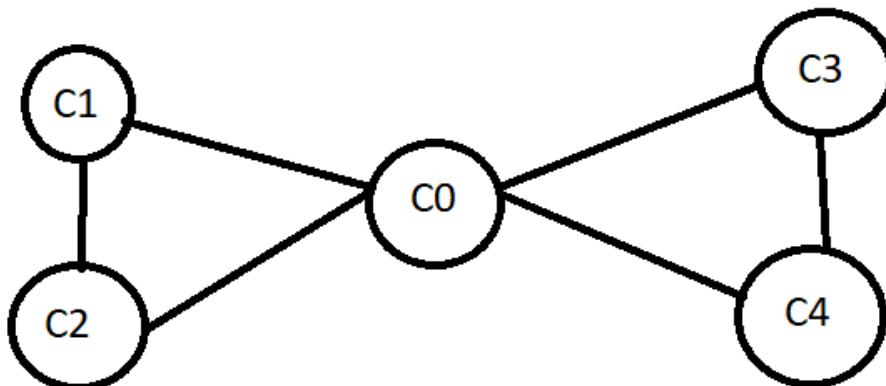
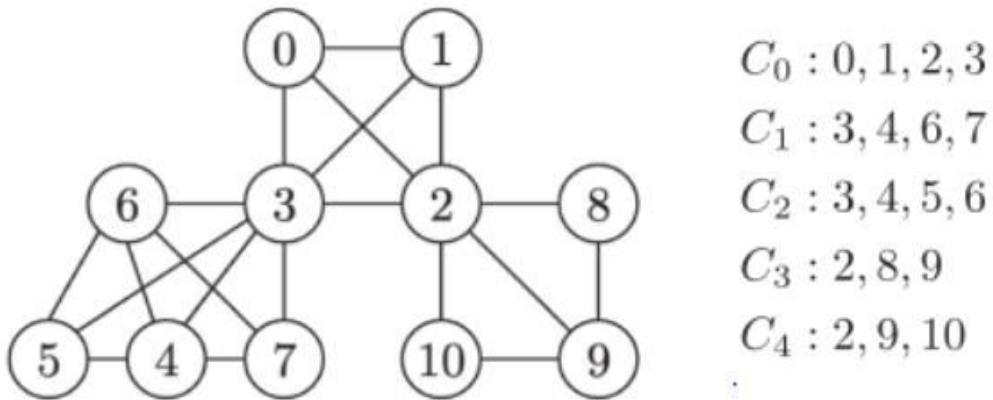
Análisis de tiempo del algoritmo top-down con memorization:

En la nuestra implementación del algoritmo top-down con memorization usamos recursividad, por lo que su análisis de tiempo está ligado a la cantidad de veces que se llama la función de forma recursiva en la función reglasrec, si observamos el caso base de la recursividad veremos que se deja de llamar recursivamente cuando ambos valores, i y j, son 0, por lo que podemos decir que, en el peor caso para este algoritmo, su cota superior será $O(n*m)$.

Ahora si quisiéramos analizar la función optimo, al igual que en el algoritmo de bottom-up, el ciclo de la función itera hasta que alguno de los 2 valore, i y j, sean 0, por lo que podemos decir que su cota superior será de $O(n)$.

Por lo anterior, decimos que el análisis de tiempo para esta implementación esta dado por la cota superior $O(n*m)$.

2.b) No, el algoritmo de vertex cover no se puede utilizar para solucionar el problema de encontrar el mínimo número de cliques que cubren el grafo un ejemplo de esto es que si realizamos el grafo de cliques correspondiente al grafo de ejemplo quedaría así:



Al aplicar nuestro algoritmo 2 aproximado daría como resultado todos los nodos menos C2 sin embargo C2 es necesario pues es el único clique que contiene C5.

A pesar de que no es posible utilizar este algoritmo para solucionar cualquier grafo dado sus cliques máximos que lo cubren, si aplicamos la restricción de que los cliques solo pueden ser de tamaño 2 estaríamos resolviendo el vertex cover como siempre por lo que sería posible aplicarlo si tomamos en cuenta esa restricción.