CC40A Diseño y Análisis de Algoritmos: Tareas 2 y 3

Profesor: Ricardo Baeza Yates
Prof. Auxiliar: Rodrigo Paredes

Plazo de entrega: fecha del examen y **no** habrá extensiones

P1 (2.5 ptos.)

Considere el problema de intersectar dos conjuntos de elementos enteros positivos que están almacenados como arreglos ordenados de tamaños m y n respectivamente (puede suponer $m \leq n$, que hay un tamaño máximo MAX y que se puede usar un 0 en la posición siguiente adonde acaba el conjunto).

Usted debe implementar los siguientes tres algoritmos:

- Calcular la intersección usando una mezcla (*merge*) de los arreglos.
- Calcular la intersección buscando cada elemento del arreglo más pequeño en el más grande con búsqueda binaria.
- Calcular la intersección buscando la mediana del arreglo pequeño con búsqueda binaria en el grande y luego resolver recursivamente los dos problemas restantes (intersectar los pedazos de la derecha e intersectar los pedazos de la izquierda).

Haga dos gráficos para m desde 1 hasta n, considerando $n = 100000$ para (a) el número de comparaciones entre elementos de los arreglos y (b) el tiempo que tardan los algoritmos. No es necesario probar todos los m, pero es importante tener mayor número de puntos en las zonas donde las curvas de los algoritmos se intersectan. Para cada caso use arreglos ordenados generados al azar (piense la mejor manera de hacer esto) y calcule el promedio de 10 casos.

El tercer algoritmo puede mejorarse si durante las llamadas recursivas usamos otro algoritmo cuando convenga (por ejemplo merge cuando m es parecido a n o búsqueda binaria cuando m es mucho más chico que n. A partir de su gráfico anterior y su apellido paterno, decida cuando llamar a otro algoritmo. Sin embargo este algoritmo será mejor que el original y entonces no necesariamente el mejor punto de corte estará dado por el gráfico anterior. Por ende, pruebe distintos puntos de corte alrededor del inicial y grafique el tiempo para cada caso para $m = 50000$ y $n = 100000$. Con esto, obtenga el punto de corte óptimo para su algoritmo y grafíquelo junto a los otros tres algoritmos anteriores para verificar la mejora obtenida (esto se llama ingeniería de algoritmos).

Primera Letra del Apellido Paterno	Algoritmo
A-L	Merge
M-Z	Búsqueda Binaria

Tabla 1: Algoritmo a usar dentro de la recursión cuando sea mejor.

P2 (1.5 pto.)

Considere el mismo problema de la pregunta anterior en el caso en línea. Es decir, no conocemos los conjuntos a priori y nos van entregando un elemento de cada conjunto a la vez. Si uno de los conjuntos acaba, se sigue recibiendo el otro conjunto. Como no sabemos cuán grande son los conjuntos no los podemos almacenar. Por otra parte, sólo nos interesan las primeras k respuestas.

Implemente un algoritmo eficiente que resuelva este problema usando espacio $O(1)$, leyendo la entrada de un archivo con el formato:

```
k
a1  b1
....
ai  bi
..  0
aj
0
```

Puede suponer que el conjunto más corto siempre está en la primera columna.

Indique cual es la competitividad de su algoritmo en función de n y m, los tamaños finales de los conjuntos.

P3 (2 ptos.)

El superstring más corto de un conjunto de strings (SCS), es aquel string de largo mínimo que contiene a todos los otros como substrings. Implemente un algoritmo eficiente que obtenga una solución subóptima para este problema cuando tenemos k strings de igual largo ℓ.

Para los experimentos use $k = 10$ y $\ell = 4$, generando los strings en forma aleatoria 10 veces para alfabetos de tamaño 2 y 4, sacando el promedio para cada alfabeto. Suponga que la entrada es de la forma

```
N k
s1
....
sk

s1
....
```

donde N es el número de problemas a resolver. Se proveerán datos para probar su programa y debe calcular para cada caso una cota superior al factor de aproximabilidad del mismo (para esto piense en una cota inferior del tamaño de la solución).