Tomás Maturana
19.186.973-7

Control 3

P1:

1) Ya que se tiene tiempo ilimitado para crear las estructuras, se puede crear una lista de largo n (con posiciones del 1 al n para simplificar la respuesta), que tenga un 0 en las posiciones donde hay un X, un 1 en las posiciones donde hay un Y, y -1 en las demás posiciones. Luego, por cada i en indiceInvertido(x) o indiceInvertido(y), eligiendo el arreglo de menor largo, se va a la posición i y se verifica si existe un 0 en la posición i-1 (en caso de elegirse indiceInvertido(y)) o si existe un 1 en la posición i+1 (en caso de elegirse indiceInvertido(x)), retornando todos los pares {i-1, i} o {i, i+1}. El tiempo de este algoritmo será de O(min(n_x, n_y)) exactamente.

2) Se crean dos árboles vEB que indiquen las posiciones en las que aparecen X e Y, uno para cada uno. Los llamaremos XvEB e YvEB. Luego, en el top de cada vEB quedarán sólo los id de los subuniversos q contienen X o Y, entonces podemos tomar en cuenta sólo aquellos sub universos en los que existen X y también Y (intersectarlos). En efecto, se puede buscar en dichos subuniversos, para cada vEB si existe un X que precede a un Y en tiempo O(log log n), buscando el predecesor para una posición de un Y en XvEB y comprobando si es una posición contigua, y en su defecto, se busca el sucesor en YeVB. Esto bastará con hacerlo como máximo A_x,y veces, ya que al buscar por una posición en XvEB que no arroja una posición contigua y luego buscar en YvEB, se eliminan todas las posibilidades de que un par de elementos entremedio de las posiciones buscadas sean uno de los pares que buscamos. Es decir, el algoritmo en el peor caso es de tiempo O(2*A_x,y*log log(n)) = O(A_x,y*log log(n))

P2:

1) Ya que no habrán pares de caracteres repetidos, la probabilidad de que el número de bloques sea S/2, es decir, el mínimo posible, será de (½)^len(s). Por otro lado, el máximo sería tener S bloques, con la misma probabilidad. La probabilidad de que se arme un bloque con dos letras, es de 1/3, ya que quedan descartadas las posibilidades de que aparezcan letras repetidas contiguas. Sea X la v.a. que dice la cantidad de bloques de S:
   E(X)= sum[i= len(s)/2 -> i=len(s)] de (i*1/3^i)
   =

2)

P3:

1) Consideremos este algoritmo recursivo que recibe dos árboles a unir, T1 y T2.
   Si alguno de los dos árboles es nulo, se retorna el que no es nulo.
   Se comparan las prioridades de cada raíz y se deja el de mayor prioridad como la raíz del nuevo árbol.
   Supongamos que el árbol que no se ocupó es T2, y que T1 tiene hijos h1 y h2, con h1.clave<h2.clave. Luego, se compara la clave de T2 con las claves de h1 y h2 y se llama nuevamente al algoritmo, pero uniendo T2 con h1 si es que T2.clave<h1.clave; o T2 con h2 si T2.clave>h2.clave.
   De esta forma queda un árbol “grande”, que es casi la unión de T1 y T2, más otros árboles de menor altura que T1 y T2 y con raíz de prioridad mucho menor, por lo tanto basta con unir este “árbol grande” con los árboles que fueron quedando afuera.
   Hay que recalcar que se mantiene la estructura de árbol aleatorizado y que el costo es inferior a recolectar todos los elementos e insertarlos en un árbol vacío.