Universidad Nacional Autónoma de México
Facultad de Ciencias
Análisis de Algoritmos | 7083
Tarea 5 : | Tarea final
Sosa Romo Juan Mario | 320051926
13/11/2024

1. **Pruebe que el segundo elemento más chico de una lista de n elementos puede encontrarse con $n + \lceil \ln n \rceil - 2$ comparaciones.**

2. **Give an $O(n\ log\ k)$-time algorithm which merges k sorted lists with a total of n elements into one sorted list. (Hint: use a heap to speed up the elementary $O(kn)$-time algorithm).**

   Voy a asumir que el problema quiere juntar "k" listas ya ordenadas por separado en una lista ordenada y no se refiere a que las listas estan k-ordenadas y se quieren juntar en una sola lista ordenada.

   Usando un monticulo minimo

   La idea es utilizar un min heap para que, en cada paso siempre extraemos el elemento mas chico lo que evidentemente resultara en una lista ordenada de n elementos.

   Lo primero es crear el heap, como las listas ya estan ordenadas y buscamos solo el mas chico, es suficiente con tener el primer elemento de cada lista en nuestro heap, esto se puede hacer en $O(k\ log\ k)$ tiempo (insertar k nodos con cada inserción tomando a lo mas log k pues el monticulo tiene k nodos).

   No creo que sea necesario explicar como funciona un min heap porque eso es mas de EDD pero tiene como propiedad que ira acomodando los nodos de forma ascendente basandose en sus valores.

   Ahora para la creación de la lista, tomamos el primer elemento del heap, lo agregamos a la lista y lo eliminamos del heap, si el nodo que agregamos a la lista tenía un nodo siguiente en su lista de origen, habrá que agregarlo al heap, podremos acabar cuando el heap este vacío.

   Ahora, como ya mencionamos el heap en cualquier momento sera de tamaño a lo mas k (el elemento mas pequeño que estamos procesando de cada lista) por lo que la complejidad en espacio será de $O(k)$, ademas, insertar en este heap toma $O(log\ k)$ tiempo y vamos a insertar en total n nodos por lo que la complejidad en tiempo sera de $O(n\ log\ k)$ (notese que $n \geq k$ si una lista es vacia ni la consideramos).

3. **Consider an $n \times n$ board of checkerboard B of alternating black and white squares. Assume that n is even. We seek to cover this checkerboard with rectangular dominos of size 2×1.**

(a) **Show how to cover the board with $\frac{n \times n}{2}$ dominos.**

(b) **Remove the upper left and lower right corners from B. Show that you cannot cover the remaining board with $\frac{n \times n}{2} - 1$ dominos.**

(c) **Remove one arbitrary black square and one arbitrary white square from B. Show that the rest of the board can be covered with $\frac{n \times n}{2} - 1$ dominos.**

4. **Suppose we are given the minimum spanning tree T of a given graph G (with n vertices and m edges) and a new edge $e = (u - v)$ of weight w that we will add to G. Give an efficient algorithm to find the minimum spanning tree of the graph $G+e$. Your algorithm should run in $O(n)$ time.**

5. **Usted tiene que ordenar una serie Σ_n de n números, tales que todos son 0 ó 1. La única operación que puede hacer es comparar dos números cualesquiera x y y, y cada que los compara recibe la respuesta $x < y$, $x = y$, or $x > y$.**

(a) **De un algoritmo que con a lo más $n - 1$ comparaciones ordena Σ_n. Pruebe que su algoritmo es óptimo.**

(b) **Encuentre un algoritmo que utilizando en promedio $\frac{2n}{3}$ comparaciones ordena Σ_n (esto bajo la suposición que los elementos de Σ_n tienen la misma probabilidad de ser 0 ó 1). Demuestre que su algoritmo es óptimo.**

6. **Consider the numerical game "20 questions". In this game, player 1 thinks of a number in the range 1 to n. Player 2 has to figure out this number by asking the fewest number of true/false questions. Assume that nobody cheats.**

(a) **What is an optimal strategy if n in known?**

(b) **What is a good strategy is n is not known?**

7. **Sea T un árbol con raíz con n vértices. Cada nodo v tiene asociado un peso $w(v)$. Utilizando programación dinámica, encuentre un algoritmo de tiempo lineal para encontrar el conjunto independiente de T de peso máximo.**

Este problema se asemeja bastante al problema 3 de la tarea 3 asi que lo vamos a abordar de manera similar. (reutilice algunos de los diagramas que hice porque me quedaron bonitos)

Aclaraciones:

- Solo vamos a considerar pesos positivos.
- El arból puede o no ser binario.
- Solo vamos a considerar arrboles con mas de 2 nodos pues si no el problema es trivial. (aunque el algoritmo que vamos a presentar funciona para arboles para todo n)

La idea es utilizar el arbol que ya existe, agregarle informacion y recorerlo de manear eficiente.

Usando DP en el arból

Lo que vamos a intentar es considerar a un nodo y sus hijos para ver si nos conviene tomarlo o no. Comenzamos por los nodos mas faciles de analizar, las hojas. Si un nodo es hoja, entonces el conjunto independiente de peso maximo que lo contiene es el mismo nodo.

Vamos a utilizar un recorrido postorden paa recorrer el arbol. De manera que procesamos primero a los hijos de un nodo antes de procesar al nodo en si, este recorrido tiene complejidad $O(n)$.

Ahora, en cada nodo vamos a agregarle información, para ello vamos a usar 2 funciones que se pueden describir asi:

- **Incluir:** Si incluimos al nodo en el conjunto, entonces sus hijos no pueden ir. Esto se ve asi:

$$\text{Incluir}(nodo) = peso(nodo) + \sum_{h \in hijos} \text{excluir}(h)$$

- **Excluir:** Si excluimos al nodo del conjunto, entonces podemos o no incluir a sus hijos. Esto se ve asi:

$$\text{Excluir}(nodo) = \sum_{h \in hijos} \max\{\text{incluir}(h), \text{excluir}(h)\}$$

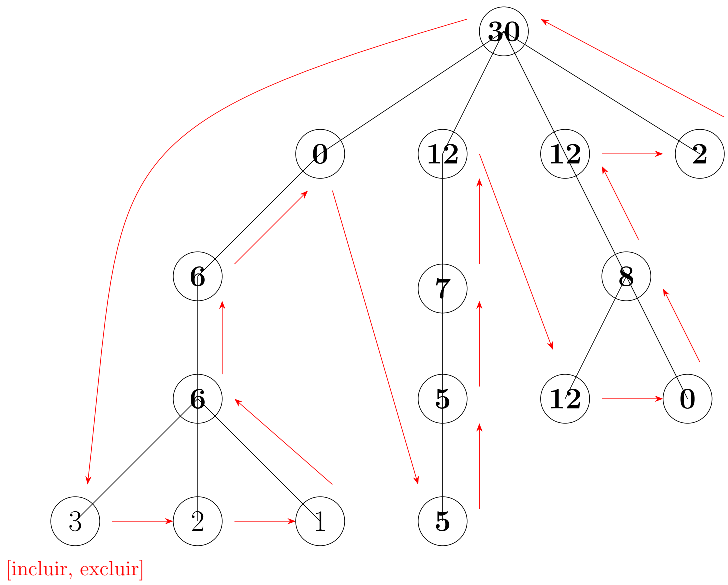
Ademas de estas funciones, vamos a agregar la base de la recurrencia, como ya dijimos cuando un nodo es hoja no tiene restricciones, por lo que podemos definir las funciones asi:

$$\text{incluir}(nodo) = peso(nodo)$$
$$\text{excluir}(nodo) = 0$$

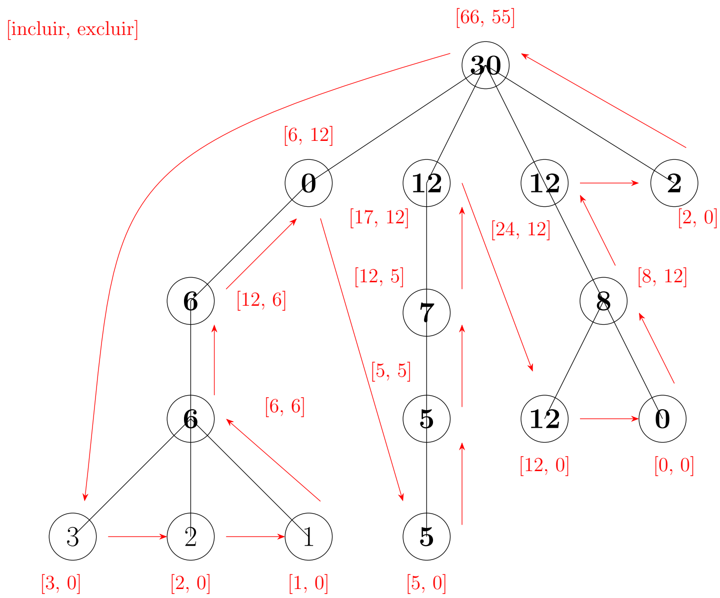
Adicionalmente, si queremos considerar negativos (aunque ya lo resuelve) podemos decir que si un valor es negativo simplemente no lo agregamos.

Entonces para cada nodo, empezando por las hojas agregamos esta información y recuperamos la información de sus hijos para poder calcular la información del nodo padre.

Entonces el arbol se va a ir llenando algo asi en el recorrido:

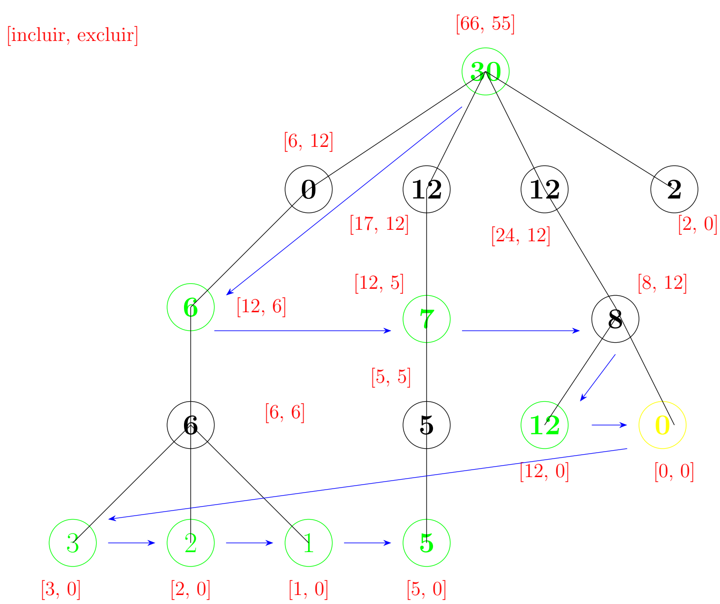


Seguimos el camino rojo y para cada nodo vamos calculando sus 2 valores, esto sucede en a lo mas O(n) pues cada nodo puede tener a lo mas n-1 hijos. (las funciones de incluir y excluir tienen que checar a todos sus hijos) pero en el caso de las hojas y en general cuando tienen una cantidad razonable de hijos (hijos "constantes" relativo a n) se hace en O(1).



De aqui ya podremos decir cual sera el peso maximo, pero el problema nos pide el conjunto independiente, asi que vamos a crearlo, para ello vamos a hacer un recorrido bfs para checar por nivel si un nodo es incluido o no, si es incluido no tenemos que cehcar a sus hijos, solo habria que checar a sus nietos, si un valor da igual podemos decidir si incluirlo o no.

Esto nos puede quedar algo asi:



El camino azul nos dice cuales tenemos que verificar pero hay que recalcar que todos los nodos entran a la fila en algun momento aunque sea para meter a sus hijos, por lo que la complejidad de este paso es O(n) (bfs usualmente es la suma de aristas y vertices pero en arboles tenemos n-1 aristas).

Los nodos que agregamos los pinte de verde pero podemos agregarlos a una lista y los que excluyo son o hijos de incluidos o nodos cuyo valor de excluir es mayor que el de incluir y no quedan coloreados (o incluidos).

En total entonces el algoritmo tiene complejidad de recorrer el arbol con un recorrido postorden, agregar informacion por cada nodo mas el recorrerlo con bfs, o lo que es lo mismo O(n)*O(1) + O(n) = O(n) y el espacio adicional es O(n) para guardar la lista de nodos que vamos a incluir. (notemos que en el caso de que un nodo tiene muchos hijos estos a su vez ya no pueden tener muchos hijos).

8. **Dado un arreglo A con n enteros positivos y negativos.**

   (a) **Diseña un algoritmo de tiempo cuadrático que encuentre la pareja de números que maximizan el producto.**

   (b) **Diseña un algoritmo de tiempo $O(n \log n)$ que encuentre la pareja de números que maximizan el producto.**

   (c) **¿Se puede mejorar el inciso anterior? Si es el caso, muestre el algoritmo, sino explique por qué.**

9. **Suppose that we are given a sequence of n values $x_1, x_2, \ldots, x_n$ and seek to quickly answer repeated queries of the form: given i and j, find the smallest value in $x_i, \ldots, x_j$.**

   (a) **Design a data structure that uses $O(n^2)$ space and answers queries in $O(1)$ time.**

   (b) **Design a data structure that uses $O(n)$ space and answers queries in $O(logn)$ time.**

10. **El intervalo común más largo de dos sucesiones X y Y , es un conjunto de elementos consecutivos de X y Y más largo que aparece en ambas sucesiones (ojo, no estamos hablando de la subsucesión común más larga). Por ejemplo, el intervalo común más largo de las sucesiones *fotografía* y *tomografía* es *ografía*. Sean $n = |X|$ y $m = |Y|$. Encuentre un algoritmo que en $\Theta(nm)$ encuentre el intervalo común más largo de X y Y.**

11. **Let $G = (V, E)$ be a bipartite graph with vertex partition $V = L \cup R$, and let G' be its corresponding flow network. Give a good upper bound on the length of any augmenting path found in G' during the execution of FORD-FULKERSON.**

12. **Professor Adams has two children who, unfortunately, dislike each other. The problem is so severe that not only do they refuse to walk to school together, but in fact each one refuses to walk on any block that the other child has stepped on that day. The children have no problem with their paths crossing at a corner. Fortunately both the professor's house and the school are on corners, but beyond that he is not sure if it is going to be possible to send both of his children to the same school. The professor has a map of his town. Show how to formulate the problem of determining if both his children can go to the same school as a maximum-flow problem.**

13. **Mientras caminas por la playa encuentras un cofre de tesoros. El cofre contiene n tesoros con pesos $w_1, \ldots, w_n$ y valores $v_1, \ldots, v_n$. Desafortunadamente sólo tienes una mochila que solo tiene capacidad de carga M. Afortunadamente los tesoros se pueden romper si es necesario. Por ejemplo, la tercera parte de un tesoro i tiene peso $\frac{w_i}{3}$ y valor $\frac{v_i}{3}$.**

    (a) **Describe un algoritmo voraz de tiempo $\Theta(nlogn)$ que resuelve este problema.**

    (b) **Prueba que tu algoritmo obtiene la solución correcta.**

    (c) **Mejora el tiempo de ejecución de tu algoritmo a $\Theta(n)$**

14. **Encuentre la parentización óptima para multiplicar seis matrices de dimensiones 4×9, 9×4, 4×10, 10×2, 2×5, 5×6.**

15. **Sea P_n una familia de n puntos en el plano, en posición general. Encuentre un algoritmo que encuentre el triángulo de mayor área cuyos vértices estén en P_n. Su algoritmo tiene que trabajar en menos de $O(n^3)$. Hint: Pruebe primero que los vértices de dicho triángulo están en el cierre convexo de P_n, esto reduce el problema al caso en que los elementos de P_n son los vértices de un polígono convexo.**

16. **Supongamos que usted es el dueño de la compañía Anuncios Espectaculares y que recibe un contrato para colocar anuncios espectaculares en la autopista México Querétaro . Los lugares donde puede colocar sus anuncios están localizados en los kilómetros $x_1, \ldots, x_n$ de dicha autopista. Si usted coloca un anuncio en el lugar localizado en el kilómetro x_i, recibirá una ganancia r_i. Por razones de seguridad, los anuncios no pueden estar muy cerca uno del otro, y la distancia entre dos anuncios consecutivos tiene que ser al menos 10 kilómetros. Observe que la distancia entre dos sitios cualquiera donde puede localizar sus anuncios no es necesariamente mayor que 10 kilómetros.**

    Encuentre un algoritmo que resuelva el problema de colocar anuncios de tal forma que las ganancias de su compañía se maximicen. Por ejemplo si:

    $$\{x_1, x_2, x_3, x_4\} = \{12, 14, 23, 28\}$$

    y

    $$\{r_1, r_2, r_3, r_4\} = \{5, 6, 5, 1\}$$

    la solución óptima sería colocar anuncios en los kilómetros 12 y 24 para una ganancia de 10.

17. **Consider the following data compression technique. We have a table of m text strings, each at most k in length. We want to encode a data string D of length n using as few text strings as possible. For example, if our table contains $(a, ba, abab, b)$ and the data string is $bababbaababa$, the best way to encode it is $(b, abab, ba, abab, a)$ a total of five code words. Give an $O(nmk)$ algorithm to find the length of the best encoding. You may assume that every string has at least one encoding in terms of the table.**

18. **La compañía *Monsters, Inc* preparó para el día de hoy una fila de $n \geq 2$ puertas para abrir y recolectar los gritos de los niños con el fin de abastecer de energía a la Monstropolis. Con el propósito de salir temprano los monstruos determinaron abrir todas las puertas de jalón y así terminar lo más rápido posible, por lo que asignaron a Sullivan a recorrer toda la fila y abrir todas las puertas. Al ver el desastre que esto causaría, Mike Wazowski recorrió la fila y cerró todas las puertas de manera alternada iniciando en la puerta 2. Sin embargo, ¡Quedaron $\frac{n}{2}$ puertas abiertas!. Por lo que el resto de los monstruos decidieron ayudar de la siguiente manera: el monstruo con el recorrido i-ésimo, cambiaría el estado de cada i-ésima puerta iniciando desde la puerta**

i. Después de hacer este proceso n veces, ¿Quedan puertas abiertas? ¿Cuántas y cuáles son si es el caso?