Algoritmos y Estructuras de Datos
Tarea 4

Profesor: Sergio Rajsbaum
Ayudante: Jorge Figueroa

fecha de hoy: 18 de octubre 2006
fecha de entrega: 31 de octubre 2006 – No se aceptan tareas después de esta fecha
— explica en detalle y con claridad todas tus respuestas —
— Tus algoritmos deberán ser lo más eficiente posibles —
— explica el funcionamiento de tus algoritmos informalmente, luego escribe el código, y luego demuestra correctez y complejidad.—
Se permite trabajar en equipos de hasta *tres* personas.
Pero cada uno debe entregar la tarea resuelta por separado, e indicar el nombre de su compañero de equipo.

Tema: Lecturas, BFS, Calendarización, Caching.

1. Escribe un resumen de no más de una cuartilla de cada uno de los capítulos: Dijkstra, Rabin, Knuth, y Tarjan, del libro *Out of their Minds* de Shasha y Lazere.

2. Al ejecutar BFS sobre una gráfica conexa $G = (V, E)$, de $|V| = n$ vértices, la cola Q donde se van almacenando los vértices descubiertos que aun no se terminan de explorar, va cambiando de tamaño al ir ejecutando el algoritmo. Digamos que va teniendo tamaños $\ell_1, \ell_2, \cdots, \ell_n, \ell_{n+1}$, donde ℓ_i es igual al tamaño de Q al inicio de su i-ésima iteración. Por lo tanto, $\ell_1 = 1$ y $\ell_{n+1} = 0$. ¿Qué secuencias son posibles? Para cada secuencia posible describe una gráfica y una ejecución de BFS que genere esa secuencia. Y para las secuencias que no son posibles, demuestra que no lo son.

3. Considera el problema de calendarización en el cual cada pedido i tiene una fecha límite d_i y requiere de un tiempo de ejecución t_i.

   Vimos ejemplos en clase que muestran que la estrategia de *fecha límite más cercana primero* es mejor que tanto las estrategias de ordenar a las tareas en orden de sus longitudes, como la de ordenarlas por sus tiempos de sobra $d_i - t_i$. Describe ejemplos similares pero generales, para instancias de n tareas. Intenta presentar ejemplos en los que sea vea que es mucho mejor que las otras dos estrategias (explicando *cuanto* mejor lo es).

4. Caching.

   (a) Un algoritmo de caching es *reducido* si trae un elemento d a la memoria en un paso i solo si en ese paso hay un pedido para d. Demuestra en detalle que se puede suponer que si A es un algoritmo óptimo de caching, entonces A es reducido.

   (b) Considera el algoritmo *más lejano en el futuro* de Belady visto en clase para el problema de caching. Describe una secuencia de $n = 6$ pedidos para la cual el algoritmo incurra

en el mayor costo posible. Demuestra que es el mayor posible sobre todas las secuencias de $n = 6$.

5. Hemos visto tres técnicas para mostrar que un algoritmo "greedy" (avaro) es óptimo:

   (a) el algoritmo greedy siempre va adelante: mostrar que despues de cada paso del algoritmo greedy, su solución es al menos tan buena como la de cualquier otro algoritmo.

   (b) intercambios: ir transformando gradualmente cualquier solución a la solución obtenida por el greedy, sin afectar su calidad.

   (c) estructural: identifica alguna propiedad simple estructural que muestre que cualquier solución debe tener un cierto valor. Luego muestra que el algoritmo greedy óptimo logra esa cota.

   Describe *en detalle* un ejemplo para cada uno de estos casos. Mientras más original (distinta de la del libro) sea tu presentación, más puntos obtendrás.