Diseño y análisis de algoritmos

Tarea 8

Profesores: Armando Castañeda y Sergio Rajsbaum

Ayudantes: Luis Gómez y Diego Velázquez

Martes 19 de Noviembre de 2019

Fecha de entrega: Martes 3 de Diciembre de 2019

INSTRUCCIONES:

- Se puede hacer en equipos de 2 o 3 personas, pero hay que entregarla individualmente. La resuelven entre todos, pero cada quien la escribe en sus palabras. Anotar en cada tarea el nombre de todos los miembros del equipo.
- Las respuestas deben estar escritas con claridad, todos los enunciados demostrados.
- No se aceptan tareas después de la fecha límite.
- No escribas la implementación de tus algoritmos.
- Si el ejercicio dice "prueba", "demuestra" o "muestra", no debes dejar ningún hecho sin justificar; esto significa que debes decir por què lo que escribes es verdad. Si lo que se pide es una explicación, es suficiente que enuncies los hechos que explican lo que se pide sin decir por qué son verdaderos pero tienes que manifestarlos completamente.

PROBLEMAS:

1. Considera el siguiente problema de *formato de textos*. Como entrada tenemos un arreglo $W[1, \ldots, n]$ tal que cada $W[i]$ es una palabra; supondremos que el último símbolo en $W[i]$ es un espacio en blanco. También recibimos un entero $\mathsf{long\_linea} > 0$ como parte de la entrada, tal que $\mathsf{long\_linea}$ es mayor o igual a la longitud de cada $W[i]$, denotada $|W[i]|$. Lo que buscamos es dar *formato* al texto en W y para esto hay que decidir cuales palabras van juntas en la misma línea. Si decidimos poner en una misma línea las palabras $W[i], \ldots, W[i']$, $i \leq i'$, lo que se denota $\ell[i, i']$, entonces la *penalización* de $\ell[i, i']$ es:

   - ∞ si $\mathsf{long\_linea} < |W[i]| + \ldots + |W[i']|$ (es decir, las palabras no caben en una sola línea);
   - de otra forma, $(\mathsf{long\_linea} - (|W[i]| + \ldots + |W[i']|))^3$.

   Entonces, la *penalización de un formato* $\ell_1[1, i_1], \ell_2[i_i + 1, i_2], \ldots, \ell_j[i_{j-1}, n]$ de W es la suma de las penalizaciones de sus líneas.

   a) Demuestra que la siguiente estrategia *greedy* produce formatos de W arbitrariamente malos, es decir, con penalizaciones arbitrariamente grandes: procesamos las palabras de $W[1]$ a $W[n]$; si aún hay espacio suficiente para meter $W[i]$ en la línea actual, entonces la metemos; de otra forma, abrimos una nueva línea en la que ponemos $W[i]$.

   b) Usa la técnica de programación dinámica para diseñar un algoritmo que encuentre un formato con penalización mínima. Demuestra la correción de tu algoritmo y haz un análisis de tiempo y espacio.

   c) Modifica tu algoritmo para que funcione con una función de penalización de línea dada; la penalización de un formato sigue siendo la suma de las penalizaciones de sus líneas.

2. Considera un arreglo $A[1, \ldots, n]$ con enteros positivos en sus entradas. Decimos que una pareja (i, j) es un *declive* si $i \leq j$ y $A[i] \geq A[j]$; la *longitud* de (i, j) es $A[i] - A[j]$. Diseña un algoritmo de tiempo y espacio $o(n^2)$ que calcule un declive de A de longitud máxima (es o pequeña; investiga ese concepto). Demuestra que tu algoritmo es correcto y haz el análisis de tiempo y espacio.

3. Considera el problema de selección de centros visto en clase. Demuestra que el siguiente algoritmo devuelve un conjunto C con a lo más k centros tal que $rc(C) \leq 2\, rc(C^*)$, donde C^* es un conjunto con a lo más k centros óptimo, es decir, con radio de covertura mínimo. Puedes suponer como correctos todos los algoritmos y afirmaciones vistas en clase.

```
Alg. SelecCentros(S, k)
   If |S| <= k then
      return S
   Else
      Inicializa C con cualquier elemento s de S
      While |C| < k do
         Elegir cualquier elemento s en S que maximice dist(s,C)
         Agregar s a C
      endwhile
      return C
```

4. Escribe una versión recursiva del algoritmo probabilístico de corte mínimo visto en clase (basado en contracciones de aristas). Demuestra que el conjunto de aristas devuelto por el algoritmo es efectivamente un corte de la gráfica inicial.

5. Tenemos n servidores que buscan coordinarse para ejecutar localmente la misma acción. De forma abstracta, los servidores llegan a un *consenso* sobre un bit, y ejecutan localmente la acción asociada. Por ejemplo, si el bit consensuado es 0, cada servidor hace *rollback* en su base de datos local, pero si el bit consensuado es 1, cada servidor hace *commit* localmente. Considera el siguiente algoritmo probabilístico para este problema:

```
Alg. ConsensoBinario
   r = 0
   While TRUE do
      1. r = r+1
      2. Cada servidor obtiene un bit aleatorio b_r con probabilidad uniforme
      3. Cada servidor comunica su bit b_r a todos los servidores
      4. Si todos los bits b_r de los n servidores son iguales, cada servidor
         ejecuta localmente la tarea correspondiente y termina
      5. De otra forma, continua
   endwhile
```

Responde a lo siguiente:

a) Demuestra que el número esperado de iteraciones para ejecutar la acción es $O(2^n)$. Tip: modela el problema con una variable aleatoria con distribución geométrica.

b) ¿Cúal es el número esperado de iteraciones si en cada una de ellas los servidores obtienen su r-ésimo bit, b_r, llamando una función shared_random_bit(r) que devuelve el mismo r-ésimo bit a todos los servidores con probabilidad p, para alguna constante $1 < p < 0$?