INF-221 Diseño y Análisis de algoritmos, 2019-2

Tarea 2

Profesor: Diego Arroyuelo
Ayudantes: Tomás Berrios, Alejandro Vilches
`tomas.berrios@sansano.usm.cl`,
`alejandro.vilches@sansano.usm.cl`

Fecha de Entrega: 18 de noviembre, 2019
Plazo máximo de entrega: 5 días.

Reglas del Juego

La presente tarea debe hacerse en grupos de 3 personas. Toda excepción a esta regla debe ser conversada con los ayudantes **ANTES** de comenzar la tarea. No se permiten de ninguna manera grupos de más de 3 personas. Las tareas deben compilar en los computadores que se encuentran en el laboratorio B-032 (LDS). Deben usarse los lenguajes de programación C o C++. Se recomienda compilar en el terminal usando `gcc archivo.c -o output -Wall` (en el caso de lenguaje C) o `g++ archivo.cpp -o output -Wall` (en el caso de C++). Si usa C++, está permitido emplear estructuras de datos de la `stl`, en caso de ser necesarias.

Problema 1: Dividir y Conquistar

Implemente el algoritmo basado en dividir y conquistar estudiado en clases para la multiplicación de números enteros, representados como números binarios de $n \leq 10.000$ bits. Asuma que los números son representados como strings, para permitir una mejor manipulación de los mismos. Implemente además el algoritmo clásico $\Theta(n^2)$ (de fuerza bruta), y muestre en un informe gráficas de comparación de los tiempos de ejecución para distintos valores de n.

Escriba un informe de a lo más 1 página usando LaTeX, que muestre los resultados experimentales mencionados anteriormente, comparando con la solución de fuerza bruta para resolver el problema. Por ejemplo, puede mostrar una gráfica comparativa que muestre cómo crece el tiempo de ejecución, a medida que crece el valor de n.

Formato de Entrada

Los datos serán leídos desde la entrada standard, en donde cada línea tendrá el siguiente formato:

```
N número1 número2
```

en donde `N` indica la cantidad de bits que tienen los números a multiplicar (`número1` y `número2`). Los valores en cada línea están separados por un único espacio. La entrada es finalizada con `EOF`.

Un ejemplo particular de entrada es el siguiente:

```
9 101101100 100011101
12 110111011001 111011110000
3 100 101
1 1 0
```

Formato de Salida

La salida del programa debe mostrarse a través de la salida standard, mostrando una cadena binaria por cada línea de la entrada. Esa cadena corresponde a la multiplicación de las cadenas correspondientes en la entrada.

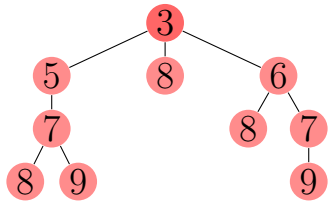
La salida correspondiente a los datos de entrada mostrados anteriormente es la siguiente:

```
11001010100111100
110011101101100101110000
10100
0
```

Problema 2: Decrecer y Conquistar

La entrada para este problema es un árbol general, es decir, un árbol en el que cada nodo puede tener una cantidad arbitraria y no limitada (*a priori*) de hijos. En este caso particular, los nodos del árbol almacenan valores enteros que son crecientes desde la raíz a una hoja del árbol. La cantidad de nodos del árbol es $0 < n \leq 1.000.000$. Además, la entrada consiste de $Q \leq 20.000$ consultas del siguiente tipo: dado un nodo cuyo número en una enumeración preorden es v, y dado un valor entero P, encontrar el ancestro del nodo que está más cerca de la raíz y que tiene un valor almacenado que es $\geq P$. Se asume que el número preorden de la raíz del árbol es 0.

Suponga que el árbol general ha sido representado de la siguiente manera: Se realiza un recorrido preorden del árbol, escribiendo un parentésis '`(`' cada vez que se llega a un nodo x, luego se visita recursivamente en preorden cada uno de los subárboles de los hijos de x, y posteriormente escribiendo un paréntesis '`)`' antes de abandonar el nodo (después de haber procesado todo su subárbol). Por ejemplo, dado el siguiente árbol general:



el mismo será representado mediante la secuencia:

```
((((()))()(()(())))
```

Note que por cada nodo del árbol se tiene un par de paréntesis `(` y `)`. Por ejemplo, a la raíz del árbol le corresponden el primer `(` y el último `)`. Por lo tanto, en total tenemos una secuencia de $2n$ paréntesis para representar el árbol. Notar que la secuencia está totalmente balanceada. Además, los valores de cada nodo serán almacenados de acuerdo a un recorrido en preorden, siendo la secuencia correspondiente para el ejemplo anterior la siguiente:

```
3 5 7 8 9 8 6 8 7 9.
```

Se pide resolver el problema mediante un algoritmo de tiempo total $O(n + Q \log n)$.

Formato de Entrada

Los datos se ingresarán a través de la entrada standard. Cada entrada consiste de varios casos de prueba, siendo la entrada terminada por `EOF`. Para cada caso de prueba, la primera línea del mismo contiene un valor `N` indicando la cantidad de nodos del árbol. La siguiente línea contiene una secuencia de `2N` paréntesis balanceados, representando el árbol tal como se explicó anteriormente. Le sigue una línea que contiene `N`

valores, correspondientes a los nodos del árbol en un recorrido preorden del mismo. Luego le sigue una línea que contiene un valor `Q`. Finalmente le siguen `Q` líneas, cada una conteniendo una consulta sobre el árbol que consiste de un número preorden $0 \leq v < \texttt{N}$, y un peso P. Asuma que el peso P siempre será menor o igual que el almacenado en el nodo con número preorden v.

Un ejemplo de entrada válida es el siguiente:

```
10
((((()))()(()(()))))
3 5 7 8 9 8 6 8 7 9
3
4 4
4 2
4 6
```

Formato de Salida

La salida debe ser mostrada en la salida standard. Por cada caso de prueba se debe imprimir el valor `Q`, seguido por `Q` líneas con las respuestas a cada una de las consultas. Cada respuesta es un único entero indicando el número preorden del nodo del árbol más cercano a la raíz, el cual sea ancestro del nodo v de consulta y cuyo valor sea al menos P.

La salida correspondiente a la entrada mostrada anteriormente es:

```
3
1
0
2
```

Entrega de la Tarea

La entrega de la tarea debe realizarse enviando un archivo comprimido llamado

`tarea1-apellido1-apellido2-apellido3.tar.gz`

(reemplazando sus apellidos según corresponda) en el sitio Moodle (`https://moodle.inf.utfsm.cl`) del curso, a más tardar el día 18 de noviembre, 2019, a las 23:55:00 hrs (Chile Continental), el cual contenga:

- Un archivo pdf con el informe de la resolución del problema 1.
- Los archivos con el código fuente necesarios para el funcionamiento de la tarea.
- `NOMBRES.txt`, Nombre y ROL de cada integrante del grupo.
- `README.txt`, Instrucciones de compilación en caso de ser necesarias.
- `Makefile`, Instrucciones para compilación automática.

Rúbrica de Evaluación

Para la corrección de la tarea se evaluarán los aspectos y descuentos estipulados en la Tabla 1. Cada ítem tendrá su puntaje del 1 al 100 y la nota final será el promedio de las ponderaciones de estos, que serán evaluados de acuerdo a los porcentajes indicados. Los descuentos son aplicados sobre la nota final.

Manejo de Memoria (Pregunta 1 y 2)	**20%**
Uso correcto de memoria (Ej. No acceder a regiones no inicializadas)	30%
Uso eficiente de memoria	30%
Administración correcta de memoria (`malloc`, `realloc`, `free`)	40%

Formato I/O (Pregunta 1 y 2)	**10%**
Uso correcto de la entrada y salida estándar	50%
Lectura e impresión en el formato solicitado	50%

Técnica de Programación (Pregunta 1)	**70%**
Uso de la estrategia de programación solicitada en el enunciado	20%
Eficiencia del algoritmo implementado	50%
Uso de las estructuras de datos adecuadas para resolver el problema	20%
Informe (formato, comparación de algoritmos y conclusiones, redacción y ortografía)	10%

Técnica de Programación (Pregunta 2)	**70%**
Uso de la estrategia de programación solicitada en el enunciado	25%
Eficiencia del algoritmo implementado	50%
Uso de las estructuras de datos adecuadas para resolver el problema	25%

Ítem	Descuento
No entregar en `.tar.gz`	-15
Código fuente está desordenado	-15
No incluir los archivos de documentación solicitados	-15
Copiar (desde algún sitio web o compañero)[1]	-100
Tarea no compila	-100
Tarea produce *segmentation fault* para cualquier caso	-100
Atraso	-20 por día
Warnings de compilación	-5 por *warning*

Table 1: Rúbrica de Evaluación con respecto al segundo problema.

[1] El incidente será reportado con el jefe de carrera.