Universidad de Talca
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Algoritmos y estructuras de datos

Tarea 1

Profesor: Rodrigo Paredes

Plazo: Viernes 20 de Abril de 2012. 0.5 puntos de descuento por día de atraso.

1. Objetivos

En esta tarea se pretende que el alumno (1) se familiarice con los lenguajes de programación C/C++ o Java, y (2) aprenda una metodología que le permita enfrentar un problema, construir un algoritmo que lo resuelva, definir un conjunto de casos de prueba apropiado para el problema en estudio y realizar los experimentos para validar el programa escrito.

2. Descripción de la tarea

Existen numerosas conjeturas sobre los números primos. En esta tarea trataremos de verificar experimentalmente si es que se cumplen o no tres de ellas, a saber:

- Conjetura Par de Goldbach: "Todo número natural par es la suma de a lo más dos primos".
- Conjetura Impar de Goldbach: "Todo número natural impar mayor que 1 es la suma de a lo más tres primos".
- Conjetura de cinco primos: "Todo número natural es la suma de a lo más cinco primos distintos".

El objetivo de la tarea es verificar experimentalmente las conjeturas. Para eso ustedes implementarán un programa en C/C++/Java que obedezca la siguiente línea de comandos:
`tarea1 --conjetura [1|2|3] -n N -v`

donde el primer par de argumentos (`--conjetura [1|2|3]`) indica que conjetura están verificando, el segundo par (`-n N`) para que valor de N lo están haciendo y `-v` es una ejecución con muchos comentarios con fines de depuración. Noten que se prueba cada conjetura por separado. Por ejemplo, para probar la conjetura impar de Goldbach, para el primer millón de números y sin activar la opción verbosa, la invocación en la línea de comandos sería:
`tarea1 --conjetura 2 -n 1000000`

Para esto ustedes deberán manejar una estructura que permita administrar a los números primos, y luego implementar algoritmos iterativos o recursivos para verificar la conjetura para cada valor de n entre 2 y N.

Si la conjetura se cumple, el programa imprime:

```
La conjetura XXX se cumple
```

En caso contrario, imprime:

```
La conjetura XXX no se cumple
Numeros que la fallan: N1, N2, ...
```

En que `XXX` es el número o texto de la conjetura y `N1, N2, ...` son los números que hacen fallar la conjetura, de haberlos.

Pruebas preliminares. Para poder verificar que sus algoritmos estén operando correctamente, ustedes implementarán una variante verbosa que les permitirá chequear manualmente (esta opción también será usada en el proceso de corrección de la presente tarea). En la variante verbosa su programa tendrá que mostrar la tabla de números primos desde 2 hasta N.

En la opción verbosa tienen que mostrar la tabla de primos. El formato de salida es de acuerdo al siguiente ejemplo:

```
tarea1 --conjetura 2 -n 20 -v
TABLA DE PRIMOS
2
3
5
7
11
13
17
19
La conjetura impar de Goldbach se cumple
```

Experimentación. Tendrán que ejecutar el programa con valores grandes de N. Por razones de uso de memoria, vuestro programa tendría que poder ejecutarse con valores de N que lleguen a 10^9 para cada una de las tres conjeturas.

En el informe indiquen como varían los tiempos de computo a medida que aumenta N en una decada de estudio (por ejemplo, de 10^8 hasta 10^9).

La tarea tendrá las siguientes penalizaciones que se aplican a la nota total de la tarea.

rango de N	$[10^9, \infty)$	$[10^8, 10^9)$	$[10^7, 10^8)$	$[10^6, 10^7)$	$[1, 10^6)$
penalización	sin penalización	-0.5 pts	-1 pt	-2 pts	-4 pts

3. Restricciones

- Los programas debe ser escritos en C, C++ o Java.
- Plazo de entrega: Viernes 20 de Abril de 2012. 0.5 puntos de descuento por día de atraso.
- Queda prohibido utilizar bibliotecas o clases preconstruidas en el lenguaje que implementen operaciones no estándar sobre arreglos. Tampoco se puede utilizar clases que administren arreglos (por ejemplo, se prohibe el uso de HashMaps u otras). USTED DEBE IMPLEMENTAR TODO.
- La tarea es individual.
- Para la tarea tienen que entregar un informe de resultados junto al código fuente (el programa que implementa la tarea). No se aceptan códigos sin informe ni informes sin código.
- Si la tarea no compila tiene nota 1.0 tanto en el código como en el informe.
- Cualquier falta a las restricciones anteriores invalida irrevocablemente la tarea.

4. Indicaciones

Si el informe de la tarea se escribe en LaTeX, tiene un punto extra en el informe.

5. Ponderación

El informe equivale a un 35 % de la nota de tarea.

El código equivale a un 65 % de la nota de tarea.

6. Entrega de la tarea

La entrega de la tarea debe incluir (1) el código del programa que resuelve la tarea y (2) un informe escrito donde se describa y documente el programa entregado, y se incluyan ejemplos de entradas y salidas. La entrega de la tarea se realizará a través de la plataforma `lms.educandus.cl` hasta las 23:55 del día del plazo final. Generen un único archivo que contenga tanto el informe como los códigos para compilar. También se solicita que se adjunte un archivo README que explique cómo compilar la tarea.

El contenido del informe debe seguir la siguiente pauta común para todas las tareas:

- Portada. El formato de la portada se muestra en la Figura 1.
- Introducción: Breve descripción del problema y de su solución.

Universidad de Talca
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Informe Tarea X

Nombre de la Tarea X

Fecha: <fecha entrega>
Autor: <nombre alumno>
e-mail: <correo del alumno>

Figura 1: Formato para la portada.

- Análisis del problema: El problema consiste basicamente en solucional la simulacion de una fila de un banco, utilizando sistemas de colas de prioridad, y colas normales, para luego establecer una estadistica con los resultados de tiempo de atencion por grupo etario. Esta simulacion esta regida por un numero finito de horas (valor entero entre 1.. 12), ademas como antecedente se tiene que llega una persona a la cola cada minuto a un ritmo constante, con edad aleatoria distribuida de forma uniforme entre 11.. 100; al igual que un tiempo de atencion variable entre 20.. 140 segundos, distribuidos de forma uniforme.

  Como estructuras necesarias, se plantea el uso de una cola de prioridad, una cola simple, y una estructura basica para almacenar la informacion de cada una de las personas. Abondar en como esta compuesta cada una de ellas, es trivial ya que es algo visto en la clase.

- Solución del problema:

  - Algoritmo de solución: Primero, dentro de el programa solicitado, comienza con la etapa de verificacion. Se validan los argumentos entregados de modo que estos no produzcan algun error propagable dentro de la ejecucion. De no ser valido algun argumento, el programa termina su ejecucion informandole al usuario del error.
    Luego comienza la etapa de la ejecucion propiamente tal. La clase "Simulador.java" se encarga de iniciar la ejecucion de la simulacion en el mismo constructor de la clase. Este configura las variables de la ejecucion, y luego define el tipo de cola a utilizar. Luego entra a un ciclo infinito, que dependiendo del retardo que tenga provoca que la simulacion tome mas o menos tiempo.
  - Diagrama de Estados: Muestra en forma global el programa.

- Diseño: Explicitar las Pre y Post condiciones consideradas, mostrar los invariantes empleados.
- Implementación: Se debe mostrar el pseudo-código del programa que soluciona el problema, explicando lo que hace. Omita cualquier detalle de implementación que sea irrelevante para entender la solución del problema. Se recomienda usar nombres representativos para las variables. NOTA: El código generado para resolver la tarea debe corresponder al diseño descrito, preocúpese de comentar el código donde sea necesario para facilitar su lectura.
- Modo de uso: Se debe explicar el modo de uso del programa y el modo de compilación. Nota la tarea debe ser compilable en los computadores de la Universidad.

- Pruebas: Se debe mostrar las pruebas realizadas y sus resultados. El número de pruebas puede variar dependiendo del problema En esta sección debe incluir las tablas y gráficos necesarios solicitados, y el análisis de los resultados. En este capítulo adjunten las conclusiones obtenidas de los resultados de la tarea.

- Anexos: De ser necesario, cualquier información adicional se debe agregar en los anexos y debe ser referenciada en alguna sección del informe de la tarea. Dentro de los anexos se puede incluir un listado con el programa completo que efectivamente fue compilado.

En este y los siguientes informes se va a valorar la ortografía y la redacción de acuerdo a las siguientes reglas de penalización.

Ortografía Se permiten hasta 10 errores ortográficos en todo el informe. Entre 11 y 20 errores descuenten 0.5 puntos al informe. Entre 21 y 30 descuentan 1.0 puntos al informe. Más de 30 descuenten 2.0 puntos al informe.

Redacción Se permiten hasta 5 ideas o párrafos mal redactados (es decir que no se entiendan). Entre 6 y 10 párrafos con problemas de redacción descuenten 0.5 puntos al informe. Entre 11 y 14 párrafos descuentan 1.0 puntos al informe. Más de 24 parrafos incomprensibles descuenten 2.0 puntos al informe.