Universidad de Buenos Aires
Facultad de
Ciencias Exactas y Naturales
Departamento de Computación

Algoritmos y Estucturas de datos III

Segundo Cuatrimestre de 2011

Trabajo Práctico 1

Problema1: Campeonato de Tennis.
Problema2: Pizza entre amigos.
Problema3: Conjetura de Goldbach.

Grupo: '1'

Integrante	LU	Correo electrónico
Cammi, Martín	676/02	martincammi@gmail.com
Garbi, Sebastián	179/05	garbyseba@gmail.com
Kretschmayer, Daniel	310/99	daniak@gmail.com

Índice

1. Ejecución del TP

1.1. Lenguaje utilizado

El lenguaje utilizado para el trabajo práctico ha sido *Java*, compilando con la versión 1.5 de la Virtual Machine.

Este trabajo se acompaña con los fuentes de la solución que puede importarse en cualquier Eclipse IDE.

Para la ejecución de los tests se ha utilizado JUnit version 4.

1.2. Como ejecutar el TP

Desde cualquier Eclipse:

- Seleccionar File ⇒ Import.
- Sseleccionar General ⇒ Existing Projects into Workspace ⇒ Next.
- Seleccionar el directorio llamado Algo3Tp1.
- Finish.

Desde la vista de **Package Explorer** bajo el paquete **src** aparecerán tres paquetes màs y dentro de cada uno de ellos los siguientes archivos de java:

- problema1

  Tennis.java

  TestTennis.java

- problema2

  Pizza.java

  TestPizza.java

- problema3

  Goldbach.java

  TestGoldbach.java

Los archivos Tennis.java, Pizza.java y Goldbach.java contienen los algoritmos solución. El resto de los archivos que comienzan con **Test** son los tests asociados. Para correr los tests basta con hacer botón derecho en cualquiera de ellos y luego Run As ⇒ JUnit Test.

2. Problema1: Campeonato de Tennis

2.1. Introducción

2.2. Explicación de la solución

2.3. Pseudo-código

2.4. Complejidad

2.5. Tests

2.6. Gráficos

2.7. Conclusiones

3. Problema2: Pizza entre amigos

3.1. Introducción

3.2. Explicación de la solución

3.3. Pseudo-código

3.4. Complejidad

3.5. Tests

3.6. Gráficos

3.7. Conclusiones

4. Problema3: Conjetura de Goldbach.

4.1. Introducción

En matemática una conjetura es una afirmación que no ha sido demostrada, pero basado en pruebas empíricas parecería ser cierta. En este contexto la Conjetura de Goldbach o Conjetura fuerte de Goldbach intenta expresar una verdad una relación entre pares y primos que no ha sido aún verificada en su totalidad, dicha conjetura enuncia que:

Todo número par mayor a dos puede ser escrito como suma de dos números primos.

En el presente trabajo implentaremos un algoritmo que, basado en la Conjetura de Goldbach y dado un número par mayor a dos retorne dos primos que sumados den dicho número.

Utilizaremos la técnica de backtraking para generar posibles soluciones e ir al mismo tiempo realizando podas para no recorrer las que no los sean.

4.2. Explicación de la solución

Como precondición para el algoritmo el número n ingresado deberá ser par. A continuación el algoritmo irá generando pares de números que sumen n y que sean candidatos a solución del siguiente modo:

1 + (n-1), 2 + (n-2), ... (n/2) + (n/2), ... (n-2) + 2, (n-1) + 1

Poda1: Lo primero que notamos es que al llegar a la mitad de pares generados la siguiente mitad será similar a la anterior ya que serán los mismos sumandos pero invertidos con respecto a la suma. Entonces con verificar que sumandos son primos en la primera mitad basta para también cubir los casos de la segunda mitad.

La primera poda entonces es solo recorrer la primer mitad:

1 + (n-1), 2 + (n-2), ... (n/2) + (n/2)

Poda2: También puede verse que el número 1 no es de por si primo así que cualquier suma en que participe no será una solución válida. Podamos entonces la solución del 1.

2 + (n-2), ... (n/2) + (n/2)

Poda3: Del mismo modo, también podemos observar que cualquier número par, salvo el 2 no será primo ya que al ser par será divisible por 2. Excluimos entonces todas las sumas con pares en ellas salvo la que contenga al 2.

2 + (n-2), 3 + (n-3) ... 2m+1 (n-(2m+1)) ... (n/2)-1 + (n/2)+1

Poda4: El caso del 2 podríamos tratarlo a parte ya que la única suma en la que participa es en la del 4 donde 4 = 2 + 2. No existe otra suma en la que el primo 2 aparezca ya que cualquier par que tenga como un sumando al 2 deberia tener como segundo sumando a otro número par y además primo y el único que cumple esa condición es el 2 caso que acabamos de excluir.

3 + (n-3) ... 2m+1 (n-(2m+1)) ... (n/2)-1 + (n/2)+1

De esta forma generamos pares de números que sumados dan el par inicial y que han sido previamente filtrados por las podas mencionadas. A continuación se verificará si alguno de ellos es solución. La definición de solución en este problema es para cada par si ambos números son primos.

Cabe aclarar que para optimizar el algoritmo la generación de candidatos y la verificación de solución se realizarán simultáneamente. No generaremos todas las soluciones y luego las verificaremos una a una, sino que verificaremos una a una a medida que las vayamos generando.

4.3. Pseudo-código

para cada n1,n2 candidatos podados **hacer**
  si esPrimo(n1) y esPrimo(n2) **entonces**
    retornar n1, n2
  fin si
fin para

4.4. Complejidad

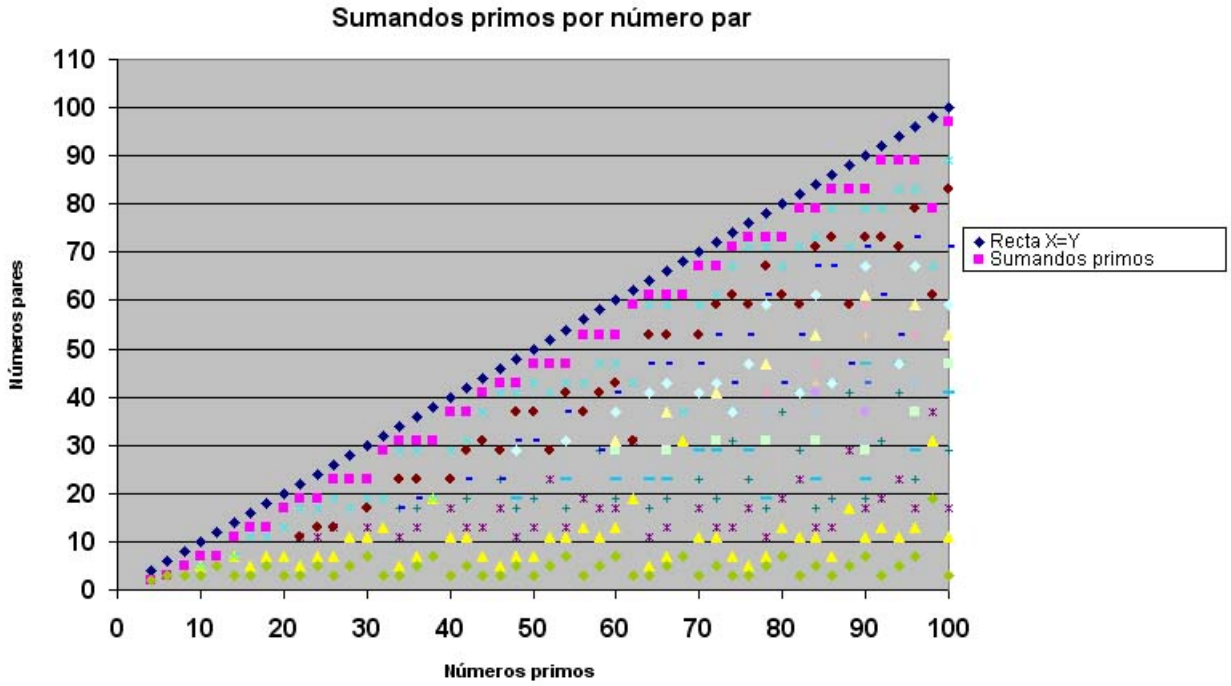
```
Paso1:    Para cada n1,n2 candidatos O(n/2)
Paso2:       Si esPrimo(n1) y esPrimo(n2) entonces O(\sqrt{n1}) + O(\sqrt{n2})
Paso3:         retornar n1, n2
Paso4:       Fin Si
Paso5:    Fin Para
```

El costo del algoritmo es de O(n/2) * O(verificarPrimos(n)) donde
Sea ci = el número impar i;
Sea `c_i` = el candidato i-ésimo.
Si n/2 es par el costo es n+4/4
Si n/2 es impar el costo es n+2/4
Si n/2 es par:
verificarPrimos(n) = $\sum_{i=1}^{n+4/4} esPrimo(2*i+1) + esPrimo(n-2*i+1)$

Como esPrimo(n) = O($\sqrt{(n)}$)
verificarPrimos(n) = $\sum_{i=1}^{n+4/4} \sqrt{(2*i+1)} + \sqrt{(n-(2*i+1))}$
$\sum_{i=1}^{n+4/4} \sqrt{(2*i+1)} + \sqrt{(n-(2*i+1))} < \sum_{i=1}^{n+4/4} (2*i+1) + n - (2*i+1)$

4.5. Tests

4.6. Gráficos



4.7. Conclusiones