

UNIVERSIDAD DE CHILE
**Departamento de Ciencias de
la Computación**
Control 1 CC40A
Semestre: Agosto 2009
Solution Usage FileNames

Problem 1 (1.5 marks)**File:** `./desperateStudent02.pbtex`**Usage:**

- *Spring 2004, University of Waterloo, SCS, Final of CS234*
- *Examen de Calificación de Doctorado, Algoritmos, Julio 2008.*

Un alumno está tomando tres cursos, y es muy importante que no repruebe ninguno. Él es muy bueno en **dynamic programming**, y como tiene sólo cuatro horas para estudiar, él quiere optimizar el tiempo a dedicar a cada curso de manera de minimizar la probabilidad de reprobárselos todos. La siguiente tabla muestra la probabilidad de reprobárselos en función del número de horas que el alumno le dedique a ese tópico.

Horas $M(h,c)$	Algoritmos	Bases de Datos	Ingeniería de Software
0	0.9	0.75	0.8
1	0.7	0.7	0.7
2	0.6	0.67	0.65
3	0.55	0.65	0.62
4	0.5	0.62	0.6

- Si la probabilidad de reprobárselos **Algoritmos** es p_1 , de reprobárselos **Bases de Datos** es p_2 y de reprobárselos **Ingeniería de Software** es p_3 , ¿cuál es la probabilidad de reprobárselos todos?

—————begin solution—————

The probability of failing all of them is $p_1 \times p_2 \times p_3$.

—————end solution—————

- Sea $C(c, h)$ la probabilidad de reprobárselos todos los tópicos en $\{1, \dots, c\}$ (c tópicos en total) si el estudiante le dedica una combinación óptima de tiempo, usando h horas en total. ¿Cómo se puede escribir $C(c, h)$ recursivamente?

—————begin solution—————

The topic c can be studied for any time t between 0 and h hours. Once decided how much time to study topic c , the optimal way to study the topics $\{1, \dots, c-1\}$ in the remaining time $h - t$ is define inductively:

$$C(c, h) = \min_{0 \leq t \leq h} \{C(c-1, h-t) \times M[t, c]\}$$

—————end solution—————

- Complete los tres espacios vacíos en la tabla siguiente, calculando $C(c, h)$.

-----begin solution-----

$C(i,j)$	0	1	...
Algoritmos	.9	.7	...
Bases de Datos	$.9 * .75 = .68$	$\min\{.9 \times .7, .7 \times .75\} = \min\{.63, .525\} = .525$	...
Ingeniería de Software	$.68 * .8 = .54$	$\min\{.68 \times .7, .525 \times .8\} = \min\{.476, .42\} = .42$	...

$C(i,j)$	0	1	2	3	4
Algoritmos	0.9	0.7	0.6	0.55	.5
Bases de Datos	0.68	0.525	0.45	0.41	.375
Ingeniería de Software	0.54	0.42	0.36	0.314	287

-----end solution-----

4. La tabla siguiente contiene los valores correspondientes a las elecciones para cada aplicación de la recurrencia.
 ¿Cómo se puede utilizar esta información para deducir la distribución de tiempo óptima?
 ¿Cuál es esta distribución?

-----begin solution-----

$t(i,j)$	0	1	2	3	4
Algoritmos	0	1	2	3	4
Bases de Datos	0	0	0	0	0
Ingeniería de Software	0	0	0	1	1

Based on this table, the optimal distribution of study time involves 3 hours of **Ingeniería de Software**, 1 hour of **Algoritmos** and 0 hours of **Bases de Datos**.

The optimal solution is always represented by the cell at the bottom right. The value in this cell indicates how many hours the topic corresponding to this row must be studied, and how many columns left is the next value to consider in the row above, from which one can iterate.

-----end solution-----

Problem 2 (1.5 marks)**File:** ./contamination06.pbtex**Usage:**

- *CS240, University of Waterloo, Midterm Spring 2006:*

maxpoints 5.00**max** 5.00**median** 2.50**average** 2.65

- *Examen de Calificación de Doctorado, Algoritmos, Julio 2008.*

La asociación local para la protección de la naturaleza organizó la recolección de 16 muestras de agua de 16 diferentes lagos, uno de los cuales está contaminado. Es posible comprobar en dos semanas, utilizando una incubadora, si una pequeña cantidad de agua está contaminada. Desafortunadamente, la asociación puede utilizar solamente cuatro incubadoras disponibles en la universidad.

Ayude la asociación y explique a ellos un procedimiento para analizar las 16 muestras en dos semanas, suponiendo que sólo una muestra contiene la bacteria, y utilizando el hecho que una incubadora es capaz de detectar una cantidad muy pequeña de bacterias, como por ejemplo en una mezcla de muestras. Su explicación puede ser muy corta, pero necesita aclarar los siguientes puntos:

1. qué pruebas son realizadas;
2. cómo analizar los resultados de dichas pruebas;
3. pruebe que el problema puede ser resuelto en dos semanas;
4. ilustre su solución con un diagrama.

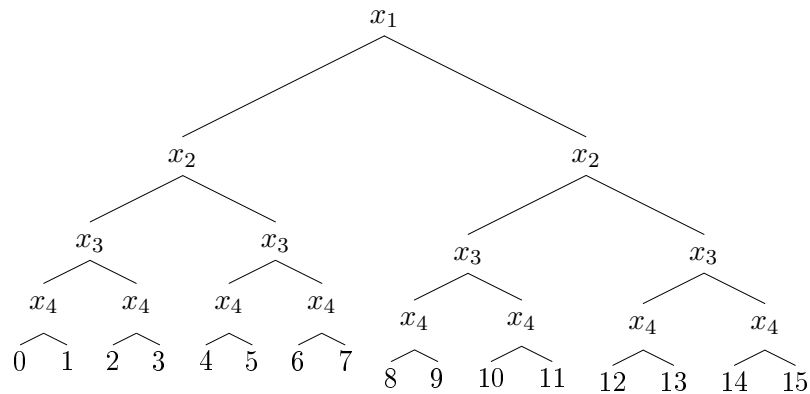
—————begin solution—————

The solution is based on putting in the incubator a combination of several samples at once. We give it for general N , just take $N = 16$ for the solution to this particular problem.

1. Prepare some mixes, each containing water from a number of samples as follows: Number the samples from 0 to $N - 1$; write these numbers in binary, and look at the bits of the binary encodings from left to right; Potion p contains some water from sample b if and only if the p th bit in the binary encoding for sample b is 1. This set of mixes is the set of tests to be run.
2. Note that with $\lceil \lg N \rceil$ bits, you can encode $2^{\lceil \lg N \rceil} \geq N$ different numbers in binary. The number of mixes needed is the number of bits needed, so the number of tests is $n = \lceil \lg N \rceil$.
3. To determine which sample contains the germ from the test results, define $(x_i)_{i \leq n}$ such that $x_i = 1$ if the mix i has a positive test result, and $x_i = 0$ otherwise. The values of $(x_i)_{i \leq n}$ form a number x on n bits, and this number (between 0 and $N - 1$) is the number of the contaminated sample.

4. For example, in the case where one sample contains the germ among $N = 16$ samples, the analysis of only $n = 4$ mixes composed as described in the array below permits us to find the sample, as shown in the tree below. Note that the bits are arranged from the heavier 1 to the lighter n (from the left to the right): this is not important as long as the same convention is applied for encoding and decoding.

Mix 1 is a mix of the water samples 8 to 15;
 mix 2 is a mix of the water samples 4 to 7 and 12 to 15;
 mix 3 is a mix of the water samples {2, 3, 6, 7, 10, 11, 14, 15};
 mix 4 is a mix of the water samples corresponding to odd numbers.



-----end solution-----

Problem 3 (1.5 marks)

File: `./suffixTree01.tex`

Usage:

- *Fall 2004, Assignment 4w*
- *Winter 2006, Assignment 5*
- *Spring 2006, University of Waterloo, SCS, Assignment 5.*

maxpoints 8.00

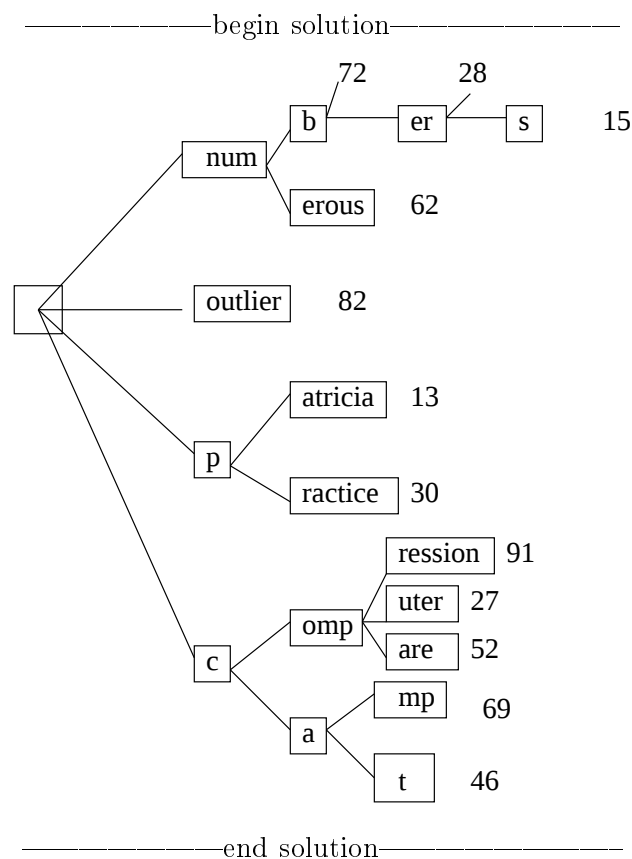
max 8.00

median 8.00

average 6.78

El index siguiendo es associando un numero a cada caldena de simbolos. Dibuja el arbol de sufijo correspondiendo, con cual se puede encontrar el numero associado a una caldena w o a todas las caldenas de prefijo w .

- | | | |
|--------------------|-----------------|-----------------|
| ■ numbers - 15 | ■ numerous - 62 | ■ camp - 69 |
| ■ compression - 91 | ■ cat - 46 | ■ practice - 30 |
| ■ patricia - 13 | ■ computer - 27 | ■ compare - 52 |
| ■ number - 28 | ■ numb - 72 | ■ outlier - 82 |



Problem 4 (1.5 marks)**File:** ./preguntas02.tex**Usage:**

- 2009 Septiembre, Universidad de Chile, DCC, Fake control 1. Average: X/X

1. Simplifica las formulas siguientes: (justifica su respuesta.)

■ $\sum_{i=1}^n i$

-----begin solution-----

Suponga que n es par, y $k = n/2$:

$$\begin{aligned}
 \sum_{i=1}^n i &= \sum_{i=1}^k i + \sum_{i=k+1}^n i \\
 &= \sum_{i=1}^k i + \sum_{i=1}^k (n - i + 1) \\
 &= \sum_{i=1}^k (i + n - i + 1) \\
 &= k(n + 1) \\
 &= \frac{n(n + 1)}{2}
 \end{aligned}$$

Suponga que n es impar, y $n = 2k + 1$:

$$\begin{aligned}
 \sum_{i=1}^n i &= \sum_{i=1}^{2k} i + 2k + 1 \\
 &= \frac{2k(2k + 1)}{2} + 2k + 1 \\
 &= \frac{(n - 1)n}{2} + n \\
 &= \frac{(n - 1)n + 2n}{2} \\
 &= \frac{(n + 1)n}{2}
 \end{aligned}$$

-----end solution-----

■ $\sum_{i=1}^n \frac{1}{2^i}$

-----begin solution-----

Considera que $\sum_{i=1}^n 1/2^i + 1/2^n = 1$. Entonces:

$$\sum_{i=1}^n 1/2^i = 1 - \frac{1}{2^n}$$

-----end solution-----

■ $\sum_{i=1}^n \frac{i}{2^i}$

-----begin solution-----

Reescriba la suma como un triangulo, donde cada linea es una suma de la forma $\sum_{i=1}^n 1/2^i$. Resulta que:

$$\sum_{i=1}^n i/2^i = 2 - \frac{1}{2^n}$$

-----end solution-----

2. Da el algoritmo para “Bucket Sort”.

-----begin solution-----

```
1. for j=1 to \sigma do C[j] <- 0
2. for i=1 to n do C[A[i]]++
3. P[1] <- 1
4. for j<- 2 to \sigma do
  - P[j] <- P[j-1] + C[j-1]
5. for i<-1 to n
  - B[P[A[i]]++] <- A[i]
```

Este algoritmo es particularmente practica para ordenar llaves asociadas con objetos, donde dos llaves pueden ser asociadas con algunas valores distintas. Nota que el ordemaniemto es *stable*.

-----end solution-----

3. Describa la estructura de datos de SkipList, y las complejidad de su operaciones.

-----begin solution-----

- una skip-list de altura h para un diccionario D de n elementos es una familia de lists $S_0, \dots, S_h$ y un puntero al primero elemento de S_h , tal que
- S_0 contiene D ;
- cada S_i contiene un subconjunto aleatorio de S_{i-1} (en promedio la mitad)
- se puede navegar de la izquierda a la derecha, y de la cima hast abajo.
- se puede ver como n tores de altura aleatorizada, conectadas horizontalmente con punteros de la izquierda a la derecha.
- la informacion del diccionario estan solamente en S_0 (no se duplica)
- Las operaciones “Search”, “Insert” y “Delete” estan suportadas en tiempo $O(\lg n)$ en promedio **sobre el aleatorio de la estructura de dato**.

-----end solution-----