ceros para lo cual hay cuatro posibilidaes, luego $\Pr(e_2) = 1/4$. Finalmente, resolvamos el problema originalmente planteado. e_3 puede verse como "tres libres o cuatro libres", si e_4 ="cuatro libres", tendremos $e_3 = e_2 \cup e_4$ además e_2 y e_4 son excluyentes ($e_2 \cap e_4 = \emptyset$), de este modo $\Pr(e_3) = \Pr(e_2) + \Pr(e_4) = 1/4 + 1/16$.
Supongamos ahora que deseamos conocer la probabilidad de que e_5 ="p_1 y p_4 estén libres". Trataremos de descomponer el problema para simplificarlo. Por un lado, que p_1 esté libre tiene probabilidad $1/2$, ya que en las cadenas que representan los estados fijamos una componente a 0 (la primera), las restantes, entonces, tienen ocho posibilidades. Igual podríamos hacer con p_4. Si nuestros procesadores no siguen una política de maestro-esclavo o algo semejante, podemos decir que el evento dado por $\{0x_2x_3x_4\}$ es independiente del evento $\{x_1x_2x_30\}$: ninguno de ellos de forma alguna implica al otro. Se tiene, también, que A y B son eventos *independientes* sii $\Pr(A \cap B) = \Pr(A)\Pr(B)$. Por tanto $\Pr(e_5) = \Pr(\{0x_2x_3x_4\} \cap \{x_1x_2x_30\}) = \Pr(\{0x_2x_3x_4\})\Pr(\{x_1x_2x_30\}) = \frac{1}{2} \cdot \frac{1}{2}$.

Ejercicio 12

1. Demostrar las igualdades de la obs. 4.
2. Para el problema de sistema de cuatro procesadores, calcular la probabilidad de los siguientes eventos:
   a) Dos procesadores libres.
   b) p_1, p_3 libres y p_4 ocupado.
   c) Dos procesadores libres y uno ocupado.
3. Diga por qué dos eventos excluyentes no son independientes (excepto si alguno tiene probabilidad cero).

Para simplificar la descripción de eventos de interés, en un espacio $\mathcal{U}$, se emplea una función que tome como un valor relacionado con los eventos considerados. Esta función, con dominio $\mathcal{U}$, se llama *variable aleatoria*. Por ejemplo, si nos interesa el número de procesadores libres, X="número de procesadores libres". Recordando los ejemplo anteriores, e_1 ="al menos un procesador libre", $\Pr(e_1) = \Pr(X \geq 1)$. Observemos que los posibles valores de X son $\{0, 1, 2, 3, 4\}$, y que $\Pr(X = 0) + \Pr(X = 1) + \Pr(X = 2) + \Pr(X = 3) + \Pr(X = 4) = 1/16 + 4/16 + 6/16 + 4/16 + 1/16 = 1$. Como lo muestra la anterior suma, la probabilidad más alta es para $X = 2$. Esta idea está relacionada con la pregunta ¿Cuál es el valor más probable de X? El valor aludido también se llama *valor esperado* o *esperanza de* X y está definida para una variable aleatoria X como

$$E[X] = \sum_i x_i \Pr(X = x_i).$$

Es importante notar que hemos supuesto, en nuestro ejemplo de los procesadores, que todos los estados son igualmente probables, es decir la probabilidad del estado 0000 es la misma que la de 0101, etc. se dice que los estado tienen una *distribución uniforme*. Esta es una suposición pero no tiene que cumplirse siempre, de hecho nos parecería más adecuado que la probabilidad de que todos los procesadores estuvieran libres fuera menor que la de los demás estados. Será común trabajar con la distribución uniforme. Por ejemplo: sea S, un arreglo de $i - 1$ fichas ordenadas por su valor. Dada una nueva ficha C_x, la probabilidad de que C_x ocupe el lugar k en el arreglo S, es $\frac{1}{i}$, esto es puede tenerse $C_x \leq C_1$ o $C_1 < C_x \leq C_2$ o $\ldots$ $C_{i-2} < C_x \leq C_{i-1}$ o $C_{i-1} < C_x$, que son todos los posibles estados y son igualmente probables.

Ejercicio 13

1. Calcule la esperanza de X cuando X representa el número de procesadores ocupados.
2. Para el ejemplo de las fichas antes descrito, sea Y la variable aleatoria que representa el número fichas que deben observarse en orden antes de elegir el lugar que le corresponde a C_x. Calcule $E[Y]$.

2. Introducción a la algoritmia

En este capítulo empezaremos nuestro estudio detallado de algoritmos. Primero definiremos algunos términos: veremos que un *problema*, tal como multiplicar dos enteros positivos, normalmente tendrá infinitos *ejemplares*, tal como multiplicar los enteros 981 y 1234. Un algoritmo debe trabajar correctamente con todo ejemplar del problema que dice resolver.
Después explicamos lo que entendemos por la *eficiencia* de un algoritmo y discutimos formas diferentes para elegir el algoritmo más eficiente en caso de que existan varios algoritmos para solucionar un mismo problema. Veremos que es crucial saber cómo cambia la eficiencia de un algoritmo en la medida que los ejemplares de un problema se hacen más grandes y son (usualmente) por tanto más dificiles de resolver. También distinguiremos entre la eficiencia promedio de un algoritmo cuando es usado sobre muchos ejemplares de un problema y su eficiencia en el *peor caso* posible. El estimado pesimista del peor caso frecuentmente es apropiado cuando tenemos que estar seguros de resolver un problema en una cantidad limitada de tiempo.
Una vez que hayamos definido lo que entendemos por eficiencia, podemos empezar a investigar los métodos usados para analizar algoritmos. Nuestra línea de ataque es tratar de contar el número de operaciones elementales, tales como las sumas y productos que ejecuta un algoritmo. No obstante, veremos que los cálculos no son tan directos ya que aún estas operaciones tan comunes se hacen más lentas conforme el tamaño de sus operandos se hace más grande. También tratamos de convenir alguna noción sobre la diferencia práctica entre un algoritmo bueno y uno malo en términos de tiempo de cálculo.

2.1. Problemas y ejemplares

Existen varios algoritmos para resolver el problema de multiplicar dos enteros positivos, un ejemplar de este problema es multiplicar 981 por 1234. No obstante, dichos algoritmos no sólo proveen una forma de multiplicar estos dos números en particular. En efecto, ellos dan una solución general al *problema* de multiplicar dos enteros positivos. Decimos que (981, 1234) es un *ejemplar* de este problema. Multiplicar 789 por 9742 es otro ejemplar del mismo problema y lo podemos expresar como (789, 9742). Pero multiplicar -12 por 83.7 no es un ejemplar del problema, por dos razones: la primera es que -12 no es positivo y la segunda es que 83.7 no es un entero. (Por supuesto, (-12, 83.7) es un ejemplar de otro problema de multiplicación más general) La mayoría de los problemas interesantes tienen una colección infinita de ejemplares. No obstante, hay excepciones. Hablando estrictamente, el problema de jugar un juego perfecto de ajedrez tiene sólo un ejemplar, ya que existe solamente una única posición inicial. Más aún existe sólo un número finito de subejemplares (las posiciones intermedias legales). Sin embargo esto no significa que el problema carezca de interés algorítmico.
Un algoritmo debe trabajar correctamente sobre todo ejemplar del problema que resuelve. Para mostrar que un algoritmo es incorrecto, sólo necesitamos encontrar un ejemplar del problema para el cual es incapaz de encontrar una respuesta correcta. Tal como un teorema propuesto se puede desaprobar con un simple contraejemplo, un algoritmo se puede rechazar sobre la base de un solo resultado incorrecto. Por otro lado, así como puede ser difícil probar un teorema es usualmente difícil probar la correctitud de un algoritmo. Para hacer esto posible, cuando se especifica un problema es importante definir su *dominio de definición*, esto es, el conjunto de ejemplares a ser considerados. Así, los algoritmos mencionados para multiplicar enteros positivos no funcionarán con operandos negativos o fraccionales, al menos no sin alguna modificación. Por supuesto, esto no significa que los algoritmos sean inválidos: porque ejemplares de multiplicación involucrando números negativos o fracciones no están en el dominio de definición que elegimos al enunciar el problema.
Cualquier dispositivo real de cálculo tiene un límite en el tamaño de los ejemplares que puede manejar, ya sea porque los números involucrados son muy grandes o porque desbordamos el espacio de almacenamiento. No obstante, este límite no se le puede atribuir al algoritmo que elegimos usar. Máquinas distintas tienen límites distintos