Algoritmos y Estructuras de Datos – Primer Parcial
Respuestas del examen. 7 de febrero de 2.006

1. (2,5 puntos)

a)

En el caso de la dispersión cerrada, el máximo número teórico de muestras a tomar sería de 2000. Sin embargo, conforme el número de muestras se acerca a ese tamaño, el tiempo de ejecución de las operaciones crece muy rápidamente, en concreto de la forma O(1/(1-**n**/**B**)), siendo **n** el número de muestras y **B** el tamaño de la tabla. Evidentemente, no se puede fijar un tope que marque el límite eficiente/ineficiente. Pero es posible fijar un tamaño promedio de las secuencias de búsqueda deseado, y calcular la proporción de llenado máximo. Por ejemplo, si fijamos tamaño 3, resolviendo la ecuación: 1/(1-**n**/**B**) = 3, tenemos una proporción de llenado de 2/3, siendo el número de muestras **n**= 2/3***B**= 1333.

En el caso de la dispersión abierta, no existe un máximo teórico de muestras, y el tiempo de las operaciones crece de forma lineal respecto del número de muestras, con un O(1+**n**/**B**). Si fijamos también tamaño 3 para la longitud promedio de las secuencias de búsqueda, el número de muestras se calcula con: 1+**n**/**B** = 3, luego **n**/**B**= 2, y el número de muestras resultante sería **n**= 2***B**= 4000.

b)

Una buena función de dispersión debe ser fácil de calcular, debe evitar los sinónimos entre las celdas adyacentes, y debe repartir los elementos "aleatoriamente" en la tabla. Es obvio que la clave son los pares (x, y). El producto (x*y) no cumple la tercera propiedad (por ejemplo, no generaría números primos) y la suma (x+y) no cumple la segunda ni la tercera. Una buena opción sería usar un método de multiplicación aplicado a ambas claves, usando factores que sean coprimos con 2000:

$$h(x, y) = (x*287 + y*911) \bmod 2000$$

c)

Suponiendo que tenemos una buena función de dispersión, los 40375 elementos se repartirían de manera más o menos uniforme entre las 2000 posiciones de la tabla. Por lo tanto, el número promedio de colisiones al insertar un nuevo elemento sería: 40375/2000 = 20,19.

d)

En general, la redispersión doble es una buena estrategia:

$$h_i(x, y) = (h(x, y) + i*C(x,y)) \bmod 2000$$

Pero, ojo, es imprescindible garantizar que los valores de C(x,y) sean distintos de 0 y coprimos con 2000. En otro caso no se recorre toda la tabla. Una posible elección sería la siguiente:

$$C(x, y) = (x*171 + y*319)*10 + 1 \rightarrow \text{valor no divisible por 2 ni por 5}$$

2. (2,5 puntos)

La operación se puede implementar fácilmente usando recursividad. Sólo hay que tener cuidado con que si la marca de fin apunta al mismo nodo, no caer en una recursividad infinita. La implementación podría ser como la siguiente:

```
operación UnionTries (A: trie; var B: trie)
    para cada carácter c hijo del nodo A hacer
        si Consulta(B, c)==NULO entonces Inserta(B, c, NuevoTrie)
        si c≠$ entonces UnionTries(Consulta(A, c), Consulta(B, c))
    finpara
```

3. (2,5 puntos)

El problema es una variación del problema de caminos mínimos con dos diferencias: buscamos máximos en lugar de mínimos, los costes se combinan con el producto en lugar de con la suma. Por lo tanto, podemos solucionarlo con una simple modificación del algoritmo de Dijkstra en ambos puntos.

```
operación Mundial (P: array [1..n, 1..n] de real)
var
    D: array [2..n] de real          // Máximas probabilidades
    S: array [2..n] de booleano      // Nodos escogidos o no
    R: array [2..n] de entero        // Nodos de paso

    // 1. Inicialización
    para v:= 2..n hacer
        D[v]:= P[1, v]
        S[v]:= false
        R[v]:= 1
    finpara
```

```
    // 2. Cuerpo del algoritmo
    para i:= 1, ..., n-1 hacer
        // Ojo, error frecuente: los nodos no se tratan de forma secuencial, sino
        // por el máximo D[v] entre los candidatos, es decir:
        v:= nodo con S[v]==FALSE y máximo D[v]
        S[v]:= TRUE
        para cada nodo w adyacente a v hacer
            si (NOT S[w]) AND (D[v]*P[v,w]>D[w]) entonces
                D[w]:= D[v]*P[v, w]
                R[w]:= v
            finsi
        finpara
    finpara
```

La solución final, la máxima probabilidad, estaría en **D[n]** (en nuestro caso **n**=12) y la estrategia óptima sería: 1 → ... → **R[R[n]]** → **R[n]** → **n**. La aplicación al ejemplo se deja como ejercicio.

4. (2 puntos)

NOMBRE Matriz_de_naturales

CONJUNTOS

M – conjunto de matrices de naturales
N – conjunto de naturales

SINTAXIS

crear: → M
asignar: M x N x N x N → M
obtener: M x N x N → N
sumar: M x M → M
productoEscalar: M x N → M
máximoHistórico: M → N

SEMÁNTICA, ∀ n, n2, i, j, i2, j2 ∈ N, ∀ m, m2 ∈ M

obtener(crear, i, j) = cero
obtener(asignar(m, i, j, n), i2, j2) = SI esIgualN(i, i2) Y esIgual(j, j2) ⇒ n | obtener(m, i2, j2)
sumar(crear, m) = m
sumar(asignar(m, i, j, n), m2) = asignar(sumar(m, m2), i, j, sumarN(n, obtener(m2, i, j)))
productoEscalar(crear, n)= crear
productoEscalar(asignar(m, i, j, n), n2)= asignar(productoEscalar(m, n2), i, j, MultiplicarN(n, n2))
máximoHistórico(crear) = cero
máximoHistórico(asignar(m, i, j, n)) = máximoN(n, máximoHistórico(m))