4to Concurso Interno de Programación

Grupo de Algoritmia de la UNAM - FES Acatlán

March 2014

Problema A - Annie la Hija de la Obscuridad

Dadas las posicines de Annie y de un campeón en el plano, se nos pide saber si Annie es capaz de encadenar los tres hechizos que conoce al campeón enemigo. El problema nos dice que no habrá obstáculos entre ellos y que el lanzamiento de los hechizos es de manera instantanea.
Los tres ataques de Annie son los siguientes:

1. "Desintegrar". Alcance $600u$.
2. "Incinerar". Alcance $500u$.
3. "Incovar". Alcance $775u$.

Por lo tanto lo que necesitamos para poder encadenar los tres hechizos es que la distancia entre Annie y el campeón sea menor o igual a $500u$ ya que de éste modo los tres ataques pueden alcanzar al campeón

Data: (A_x, A_y) := coordenadas de Annie, $(C[i]_x, C[i]_y)$:= coordenadas del i-ésimo campeón, N := número de campeones
for $i \leftarrow 1$ **to** N **do**
  if $(A_x - C[i]_x)^2 + (A_y - C[i]_y)^2 \leq 500^2$ **then**
    `print("FULL COMBO")`
  else
    `print("OUTPLAYED")`
  end
end

Problema B - Colores

Nos dan un diccionario de colores cuyo código está en binario. En el problema se te da el código alterado (intercambiando unos y ceros) de un color y se te pide imprimir el color correspondiente del código original.
Dado que el tamaño del diccionario es muy pequeño el problema puede ser resuelto usando ocho **if**, uno para cada caso.

Un ejemplo de uno de éstos **if** es:

Data: d_1, d_2, d_3, el código alterado del color
if $d_1 == 0$ **and** $d_2 == 1$ **and** $d_3 == 0$ **then**
  | `print("Verde")`
end

Problema C - CandyLand

Se da un arreglo A de tamaño N de enteros $-100 \leq S_i \leq 100$ y se nos pide encontrar un subarreglo B consecutivo de tamaño $K \leq N$ en A tal que la suma de los elementos en B sea la máxima posible.

Si definimos $acum[x] = \sum_{i=1}^{x} S_i$. Podemos ver que para cualesquiera índices a y b tenemos

$$\sum_{i=a}^{b} S_i = acum[b] - acum[a-1]$$

En especial si $b - a + 1 = K$ tenemos la suma de un subarreglo de tamaño K

Usando el hecho anterior podemos construir el siguiente algoritmo

Data: A[ ], acum[ ], K, best := suma máxima del subarreglo, ind := índice del primer elemento de la izquierda del subarreglo

```
acum[0] ← 0
for i ← 1 to N do
    acum[i] ← acum[i − 1] + A[i]
end
for i ← K to N do
    if acum[i] − acum[K] > best then
        best ← acum[i] - acum[K]
        ind ← i − K + 1
    end
end
print(ind best)
```

Problema D - Raíces digitales

En el problema se define $S(n)$ como la suma de los dígitos de n y

$$rd(n) = \begin{cases} S(n) & \text{si } S(n) < 10 \\ rd(S(n)) & \text{si } S(n) \geq 10 \end{cases}$$

Se nos da $1 \leq k \leq 100$ y $1 \leq d \leq 9$, y se nos pide encontrar el número más pequeño *min* y el número más grande *max* tales que $rd(min) = rd(max) = d$ y tanto *min* como *max* contengan k dígitos. Éstos dos números se construyen de la siguiente manera:

$$min = (1)(\underbrace{0 \ \ldots \ 0}_{k-2})(d-1)$$

$$max = (\underbrace{9 \ \ldots \ 9}_{k-1})(d)$$

Problema D - Raíces digitales

Ejemplo: $k = 7$, $d = 5$

$$min = 1000004$$
$$max = 9999995$$

Que min cumple con $rd(min) = d$ es sencillo de ver. Comprobarlo para max es un poco más complicado.

Supongamos que n lo podemos escribir como la concatenación de dos números a y b. Por ejemplo, si $n = 1427$, entonces $a = 1$ y $b = 427$.
Es claro que $S(n) = S(a) + S(b)$.
Aplicando el operador S a ambos lados, tenemos:

$$S\Big(S(n)\Big) = S\Big(S(a) + S(b)\Big)$$

Problema D - Raíces digitales

Como $rd(n)$ se define en términos de $S(n)$, entonces

$$\begin{aligned} rd\Big(rd(n)\Big) &= rd(n) \\ &= rd\Big(rd(a) + rd(b)\Big) \end{aligned}$$

Ahora necesitamos ver que $rd(9 \times n) = 9$, $1 \leq n \leq 100$. Para ésto comprobamos a "mano" que $S(9 * m) = 9$, $1 \leq m \leq 9$ y a partir de aquí generalizamos.

Usando éstos dos resultados tenemos:

$$\begin{aligned} rd(max) &= rd\Big(rd(\underbrace{9 \ldots 9}_{k-1}) + rd(d)\Big) \\ &= rd(9 + d) \\ &= d \end{aligned}$$

Problema E - Buscando asiento

El problema nos dice que hay A niños que se quieren sentar en B asientos, sin embargo cada uno de los niños quiere sentarse en un asiento tal que sus dos vecinos estén desocupados. Los niños van sentándose uno por uno escogiendo cualquier asiento que cumpla con los requerimientos. Debemos ver si siempre es posible que estén agusto independientemente de cómo se vayan sentado (caso "Si"), si depende de los asientos que escojan (caso "Tal vez"), o sin importar que asientos escojan no es posible que estén agustos (caso "No").

Problema E - Buscando asiento

1. Caso "No". Si $A > \left\lceil \frac{B}{2} \right\rceil$. Pues la mejor forma en que se pueden ir sentando es dejando un espacio de separación, si así no es posible sentarlos, entonces no hay forma de que estén agusto.
2. Caso "Si". Si $A \leq \left\lceil \frac{B}{3} \right\rceil$. La "peor" forma en la que se pueden acomodar los niños es si cada uno se coloca a dos asientos de separación de sus vecinos, ya que si se sientan a un asiento de separación sería la forma más "compacta", y si se sientan a más de dos, entonces un tercer niño puede sentarse en medio de ellos. Ésto se vería de la siguiente forma:

$$...010\underbrace{010}_{3}010...$$

   Por lo tanto se puede apreciar que cada niño "cubre" tres asientos.
3. Caso "Tal vez". Si $\left\lceil \frac{B}{3} \right\rceil < A \leq \left\lceil \frac{B}{2} \right\rceil$

Problema F - Fibonacci

En este caso se nos da una cadena de dígitos F cuya construcción se parece a la sucesión de Fibonacci. La forma de construirla es sumar los dos últimos dígitos de la cadena y concatenar el resultado. Los primeros dígitos de la cadena son los siguientes:

$$\underbrace{1123581347}_{10}112...$$

Y se nos pide saber cuál es el n-ésimo dígito de la cadena F.

Los dígitos señalados siempre se van a repetir, debido a que se toman los dos últimos dígitos para construir los siguientes. Por lo tanto sólo debemos guardar los diez primeros dígitos y el n-ésimo dígito de F será $F[n\%10]$.

Problema G - Código

En el problema se nos da una S cadena de tamaño $N \times M$ codificada. La forma en que se codifico la cadena fue metiéndola en una matriz de tamaño $N \times M$ y rotarla 90 grados en sentido del reloj. Después de ésto se agarran los caracteres dentro de la matriz rotada en orden de arriba hacia abajo y de izquierda a derecha.

La tarea consiste en decodificar el mensaje. Para hacerlo hay que meter la cadena en una matriz de $M \times N$, rotarla 90 grados en sentido contrario al reloj e imprimir los caracteres en el mismo orden. Esto se puede lograr con puro manejo de índices:

```
for i ← M to 1 do
    for j ← 1 to N do
        print(S[i + M × j])
    end
end
```