7203 - At most twice

Autor: Pablo Ariel Heiber, Argentina
Solución: Nicolás Javier Guimar, Argentina
Catergoría: Adhoc

13 de marzo de 2016

Solución

El problema nos pide, dado un número entero $U(1 \leq U \leq 10^{18})$, encontrar un número L tal que $L \leq U$, donde L no puede repetir más de dos veces cualquiera de sus dígitos.

Para esto utilizamos tres vectores:

- `entrada`: donde almacenaremos el número ingresado.
- `resultado`: donde iremos generando el número L que tenemos que imprimir por pantalla.
- `frecuencia`: donde almacenaremos la cantidad de ocurrencias de cada dígito presente en nuestro vector `resultado`.

La estrategia consiste en recorrer el vector `entrada`, de izquierda a derecha, e ir copiándolo, dígito a dígito, a nuestro vector `resultado`[30−38] hasta que:

- nos encontremos con un dígito que ya está dos veces en dicho vector (frecuencia=2). En este caso nos quedamos con el índice `i` de esta "posición de conflicto" y procedemos de la siguiente forma. Primero buscamos si existe algún número `num` menor que el dígito en la "posición de conflicto" que tenga una `frecuencia[num] != 2` [42]. Luego:
  - Si encontramos dicho número, actualizamos su frecuencia, lo insertamos en la posición `i` de `resultado`, y llamamos a función `rellenar()` con la posición `i+1` y la logitud del número ingresado `len` [58−60]. Esta función[9−19] lo que hará es rellenar el vector `resultado` en el rango `[inicio,fin)` con los dígitos disponibles (según el vector `frecuencia`) de mayor a menor. Por último se imprime `resultado` [64].
  - Si no lo encontramos[44], procedemos a iterar sobre el vector `entrada` de derecha a izquierda, desde la posición actual `i` hasta su inicio (`i=0`) [46]. En cada posición decrementaremos la frecuencia del dígito que tenemos en `resultado`, puesto que ya la hemos contabilizado[30−38], y buscaremos nuevamente si existe algún número `num` menor que el dígito de la posición actual, que tenga una frecuencia < 2. Si no existe, seguimos con la posición siguiente[50]. Si existe, actualizamos su frecuencia, lo insertamos en `resultado` y terminamos de rellenar el vector[51−54].
    Un caso particular sucede cuando nos encontramos en la primer posición del vector (`i=0`, última iteración) y encontramos que `num=0`, si rellenamos de posición siguiente a la actual como lo veníamos haciendo, el resultado sería un número con la forma `0xxx...`, lo cual es incorrecto, es por eso que agregamos este caso particular[49] para poder llamar a `rellenar` en el rango `[0,len-1)` puesto que el resultado disminuyó un orden de magnitud. Finalmente imprimimos `resultado`.
- ó terminemos de procesar U, en cuyo caso sólo nos resta imprimir el mismo número[64] (recordar que puede darse que $L = U$).

Código

Código 1: at_most_twice.cpp

```cpp
#include <cstdio>
#include <cstring>

using namespace std;

char entrada[19], resultado[19], frecuencia[10];
int i, num, len;

inline void rellenar(int inicio, int fin)
{
    int f = 9;
    while(inicio < fin)
    {
        if(frecuencia[f] < 2)
            resultado[inicio] = f+'0', ++frecuencia[f], ++inicio;
        else --f;
    }
    resultado[fin] = '\0';
}

int main(int argc, char const *argv[])
{
    while(scanf("%s",entrada)!=EOF)
    {
        memset(frecuencia,0,sizeof frecuencia);
        memset(resultado,'\0',sizeof resultado);

        len = strlen(entrada);

        for(i=0;i<len;++i)
        {
            num = entrada[i]-'0';
            if(frecuencia[num] < 2)
            {
                ++frecuencia[num];
                resultado[i] = num + '0';
            }else break;
        }

        if(i != len)
        {
            for(num = entrada[i]-'0'-1; num>=0 && frecuencia[num]==2; --num);

            if(num == -1)
            {
                for(--i, --frecuencia[entrada[i]-'0']; i>=0; --i, --frecuencia[entrada[i]-'0'])
                {
                    for(num = entrada[i]-'0'-1; num>=0 && frecuencia[num]==2; --num);
                    if(i == 0 && num == 0) {rellenar(0, len-1); break;}
                    if(num == -1) continue;
                    ++frecuencia[num];
                    resultado[i] = num + '0';
                    rellenar(i+1, len);
                    break;
                }
            }else
            {
                ++frecuencia[num];
                resultado[i] = num + '0';
                rellenar(i+1, len);
            }
        }

        printf("%s\n", resultado);
    }
    return 0;
}
```

Código 1: at_most_twice.cpp