**Parcial Final Algoritmos II-2006**
**Nicolás Ramírez Calderón 257084**

**1.** An array *A*[1 _ n] contains all the integers from 0 to n except one. It would be easy to determine the missing integer in *O*(n) time by using an auxiliary array *B*[0…n] to record which numbers appear in *A*. In this problem, however, we cannot access an entire integer in *A* with a single operation. The elements of *A* are represented in binary, and the only operation we can use to access them is "fetch the *j*th bit of *A*[i]," which takes constant time. Show that if we use only this operation, we can still determinate the missing integer in *O*(n) time. (Hint: bucket sort binary).

**R//**

Se asume que *n* es al menos una potencia de 2.

- ✓ Iterar sobre todos los elementos de *n*, examinando el bit menos significativo. Colocar esos números donde 1 es el bit menos significante dentro de un arreglo, y esos en que 0 es el bit menos significativo en un segundo arreglo. Un arreglo tendrá un elemento más que el otro. Descartar el arreglo más grande. (Si el arreglo más grande es de todos pares, entonces sabemos que el número perdido es impar; así mismo, si el arreglo más grande es de impares, entonces sabemos que el número perdido es par.) Esto toma tiempo proporcional a O(n).

- ✓ Ahora se iteramos sobre los restantes n/2 elementos, examinando el segundo bit menos significativo. Ubicar esos números en que 1 es el segundo bit menos significativo. Dentro de un arreglo, y los que tienen 0 como segundo bit menos significativo dentro de otro. Nuevamente uno de los arreglos va a tener más elementos que el otro. Descartar el más grande. Cada uno de los números del arreglo restante tiene el mismo segundo bit menos significativo. El número perdido también tiene éste como su segundo bit menos significativo. Este paso toma tiempo proporcional a *O*(n/2).

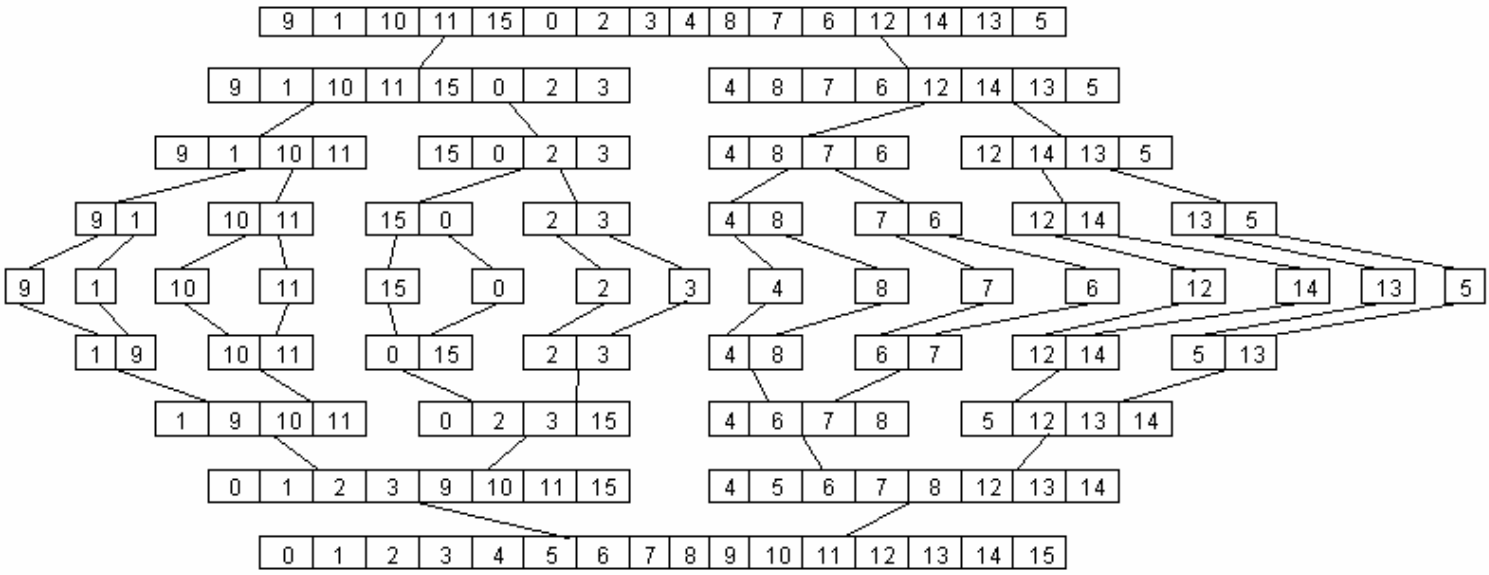
- ✓ Ahora iterar sobre los restantes n/4 elementos. Partiendo en 2 subarreglos de tamaño n/8 examinando el tercer bit menos significativo. Descartado el arreglo más grande. Esto toma tiempo proporcional a *O*(n/4).

- ✓ Continuar con la estrategia reiterativa de dividir y descartar hasta que se encuentre en número perdido.

Un acercamiento iterativo toma tiempo proporcional a $n+\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+...+1$, que es limitado por 2n. Por lo tanto el algoritmo anterior gasta un tiempo proporcional a *O*(n).

**2.** Draw the recursion tree for the MERGE-SORT on an array of 16 elements. Explain why memorization is ineffective in speeding up a good divide-and conquer algorithm such as MERGE-SORT.

**R//**




La memorización es inefectiva en un buen problema del estilo de “divide y vencerás” ya que ésta es usada cuando se tratan subproblemas que se sobrelapan. En el caso de este algoritmo no existen problemas que se sobrelapen entonces se puede decir que NO es necesaria la memorización y es inefectiva.

**3.** Professor Midas drives an automobile from Newark to Reno along Interstate 80. His car’s gas tank, when full, holds enough gas to travel n miles, and his map gives the distances between gas stations on his route. The professor wishes to make as few gas stops as possible along the way. Give an efficient method by which Professor Midas can determine at which gas stations he should stop, and prove that your strategy yields an optimal solution.

**R//**

Posible solución:

<u>Estrategia</u>

Se desarrolla una estrategia para resolver este problema. En esta estrategia siempre se irá a la parada más lejana posible para llenar el tanque. Se para en la ciudad donde no se pueda ir más lejos, se llena el tanque y se continúa hasta que se llegue a la ciudad de destino. Dado $W = \{W_1, W_2, ..., W_k\}$ que es la secuencia de paradas de el acercamiento. Dado $R = \{R_1, R_2, ..., R_i\}$ que es la solución óptima. S tiene costo(W) = k y costo(R) = i respectivamente. Desde que R sea óptima entonces $i \leq k$.

Prueba de la primera parada

Primero se prueba que $W_1$ una de las entradas de la solución óptima. Se denota dist(s1, s2) como la distancia entre las dos paradas. Se observa que la distancia entre $dist(Newark, W_1) \geq dist(Newark, R_1)$ de todas maneras se debe escoger $R_1$ en ves de $W_1$.

Si $R_1 = W_1$ esta bien. Ahora se supone $dist(Newark, W_1) \rangle dist(Newark, R_1)$ , se crea otra solución $R' = \{W_1, R_2, ..., R_i\}$. Se observa que $dist(W_1, R_2) \leq dist(R_1, R_2)$, si se tiene suficiente gasolina para manejar de $R_1$ a $R_2$, esto es también factible de $W_1$ a $R_2$. También costo(R')=costo(R)=i, que da una óptima solución. Así R' es factible y óptimo.

Prueba de subestructura óptima

Ahora se encuentra un pequeño subproblema, con parada de inicio en $R_1$ a Reno. Se necesita mostrar que $R'' = R - R_1 = R' - W_1$ es la solución óptima al subproblema. Se tiene $\cos to(R'') = i - 1$. Suponer por lo contrario que R'' no es una solución óptima para este subproblema, se tiene otra solución óptima $T = \{t_1, t_2, ..., t_m\}$. Como P es mejor solución que R'', $\cos to(T) = m < \cos to(R'') = i - 1$. Ahora se considera $W_1$+T, que es una óptima solución para el problema inicial, teniendo costo($W_1$+T)=m+1<i, esto contradice que R es una solución óptima del problema original. Por esto se ha probado que $\cos to(W) = k = i$.