Laboratorio No. 5
Algoritmos voraces o codiciosos (Greedy algorithms)

Daniel Rendón Montaño
Juan Daniel Morales Arias
Brayam Cristyam Sepúlveda Vásquez

Preguntas y Respuestas

1. Para resolver el problema del agente viaje, usando un algoritmo voraz, aun cuando no arroje la solución óptima, ¿qué debe cumplir el grafo para que el algoritmo, al menos, arroje una solución, así no sea óptima?

   R/ Para resolver el problema del agente viajero usando algoritmos voraces, tenemos que garantizar que va a ser posible visitar todos los vértices y llegar al punto de partida, si pudiésemos saber si un grafo tiene un camino Hamiltoniano, esto nos lo garantizaría. Pero la forma más fácil de saber es que estén todos los vértices conectados, lo cual nos garantiza que va a ser posible finalizar en el vértice del cual partimos.

2. Codigo UVA
   Tomado de internet, lo explicamos dentro del codigo.

```
#include
<iostream>
#include <algorithm>

using namespace std;
 //Tomado de https://codingrush.wordpress.com/2012/07/14/uva-11389-the-bus-driver-problem/


int main(){
    int n,d,r,i,overtime,temp,morning[105],evening[105];
    //Se ejecuta por siempre hasta que alguno de los valores no sea valido.
    while (true){
        cin>>n>>d>>r;
        if (!n || !d || !r)return 0;
        //Almaceno el peso de las rutas del dia y de la noche
        for (i = 0 ; i < n ; i ++)
            cin>>morning[i];
        for (i = 0 ; i < n ; i ++)
            cin>>evening[i];
```

```
        //Se ordenan los pesos de menor a mayor
        sort(morning,morning+n);
        sort(evening,evening+n);

        overtime = 0;
        //Se iteran ambos arreglos cogiendo el peso mas pequeno de los de la
manana
        //Y el mayor del de la noche, se hace la combinacion de las horas y se
mira
        //Si a alguno se le debe pagar horas extra. Si hay horas extra, se
suman y se
        //Multiplican por el coste de la hora adicional.


        for (i = 0 ; i < n ; i ++){
            temp = morning[i]+evening[n-i-1];
            if (temp > d)
                overtime+=r*(temp-d);

        }
        //Imprimo el tiempo extra
        cout<<overtime<<endl;

    }

    return 0;
}
```