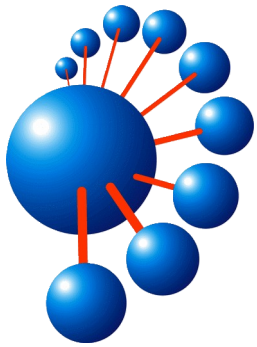


Tarea 2
Análisis de Algoritmos

Profesora: Andrea Rodríguez.
Integrantes: Diego Varas.
Jeremías Torres.

Mayo 26, 2017

Introducción

Un ladrón intenta escapar de la policía, para ello viajará por varios días de ciudad en ciudad para perder su rastro. Cuando esté razonablemente seguro que perdió el rastro de la policía, decidirá viajar a Atlanta donde venderá lo robado. El ladrón quiere gastar la menor cantidad posible en vuelos. El valor del pasaje depende de la ciudad de origen y destino, además del día del viaje. Cada ciudad tiene un horario de vuelos que se repite cada día.

Se pide poder encontrar el coste mínimo que gastará en pasajes para poder llegar desde París a Atlanta.

Desarrollo

Para resolver el problema utilizamos el algoritmo de Bellman-Ford el cual consiste en encontrar el menor valor de un nodo a otro.

```
Begin
for (d=1) to  k
for (i=1) to n
for (j=1) to n
if (i != j)
q <- (d-1) % dia[j][i]+1
if (costo_vuelo[j][i][q] && costo_min[j][d-1] != INF)
costo_min[i][d] <- min(costo_min[i][d],
costo_min[j][d-1]+costo_vuelo[j][i][q])
end if
end if
end for
end for
end for
end
```

- costo_vuelo[j][i][q] -> Costo de viajar desde una ciudad **j** a una ciudad **i** un día **q**.
- costo_min[j][d] - > Costo mínimo de viajar a una ciudad **j** en un vuelo **d**.

Demostración:

Para demostrar que este algoritmo es correcto utilizamos el método del bucle invariante:

```
Begin
for (d=1) to  k
for (i=1) to n
for (j=1) to n
if (i != j)
q <- (d-1) % dia[j][i]+1

if (costo_vuelo[j][i][q] && costo_min[j][d-1] != INF)
then
if(costo_min[i][d] > costo_min[j][d-1] + costo_vuelo[j][i][q])
then
costo_min[i][d] <- costo_min[j][d-1] + costo_vuelo[j][i][q])
end if
end if
end if
end for
end for
end for
end
```

- Para ver de manera más sencilla la demostración tomaremos las siguientes igualdades:

- costo_min[i][d] = d(v)
- costo_min[j][d-1] = d(u)
- costo_vuelo[j][i][q] = t(u,v)

- Entonces la invariante sería d(v) >= min(s,v) (mínima distancia de un vértice inicial s y un vértice final v).

- Para demostrar que la invariante se mantiene sobre cualquier paso del cálculo del costo mínimo, usaremos contradicción.

- Entonces

  d(u) + t(u,v) = d(v)
  d(u) + t(u,v) <= min(s,u) + t(u,v)

Nos da que d(u) < min(s,u). Al calcular el costo mínimo entre u y v no produce cambios en d(u) y esta desigualdad debe ser verdad para poder calcular el costo mínimo, lo cual contradice el cambio de v por el cual d(v) < min(s,v). Con esto concluimos que la invariante d(v) >= min(s,v) se mantiene para todo v en V.

Además cuando el valor de d(v) alcanza la cota inferior nunca cambiará, ya que está demostrado que d(v) >= min(s,v). Siendo el cálculo del costo mínimo una función que solo decrece los valores de d(v).

Costo Asintótico:

La complejidad en el peor caso del algoritmo de Bellman-Ford es O(V*E), ya que hace E cálculos de costo mínimo para V-1 iteraciones. Así el peor escenario de Bellman-Ford corre en O(V*E).

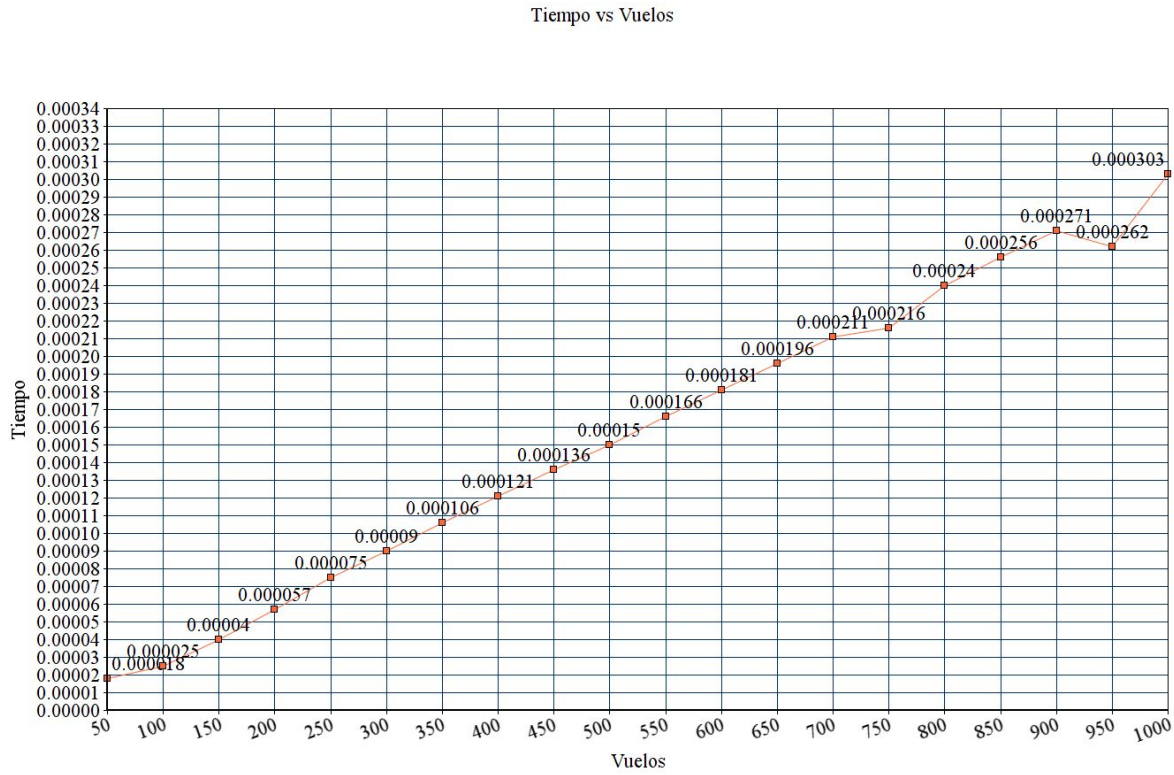
En el mejor escenario el número de iteraciones es mucho menor para ciertos grafos solo se necesita 1 iteración, por lo que en el mejor de los casos es O(E) el tiempo necesario. Un ejemplo que solo necesitaría una ronda de cálculos de costo mínimo es el grafo donde cada vértice se conecta al siguiente de forma lineal.

Ahora en nuestro caso el algoritmo de Bellman-Ford realiza n iteraciones para n ciudades para k vuelos. Ya que analizamos cada vértice con sus adyacentes dependiendo del día que se realiza el vuelo, añadiendo que puede ir y volver a una ciudad. Por lo tanto, la complejidad en nuestro problema es de O(n^2*k).

Evaluación Experimental:

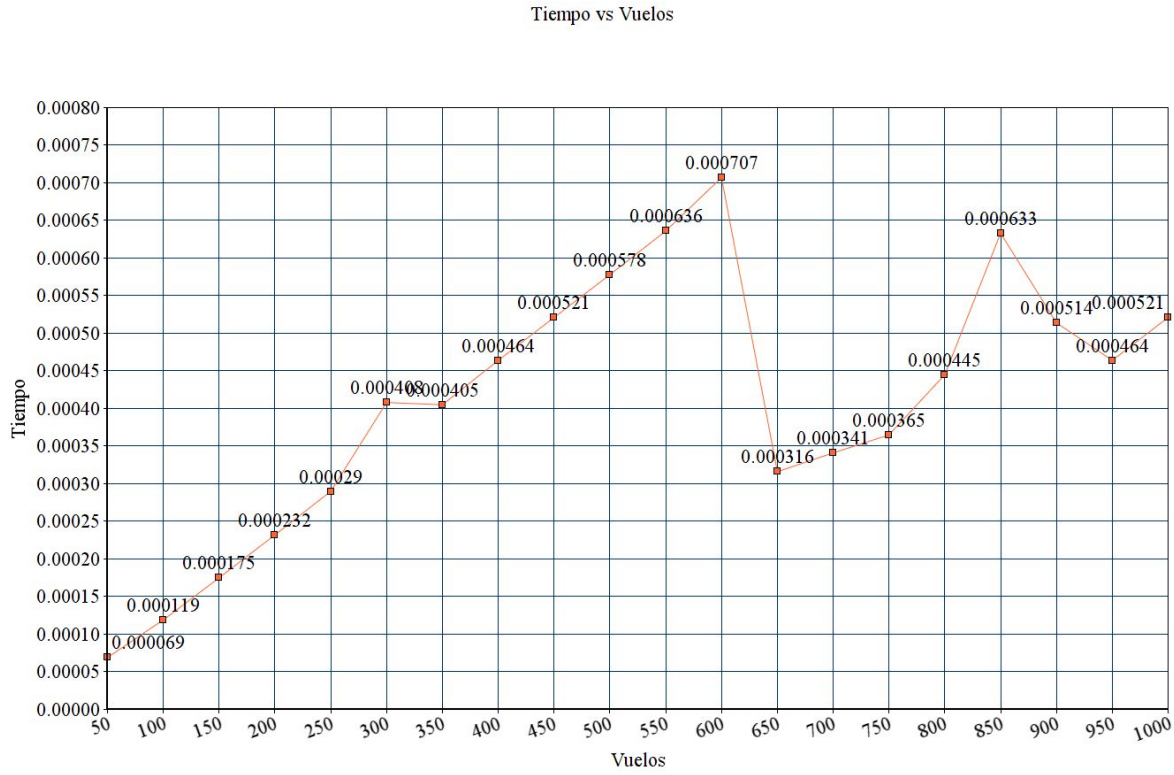
En este apartado presentaremos 3 gráficos en donde se probó variando el número de ciudades y el número de vuelos.

Grafico 1



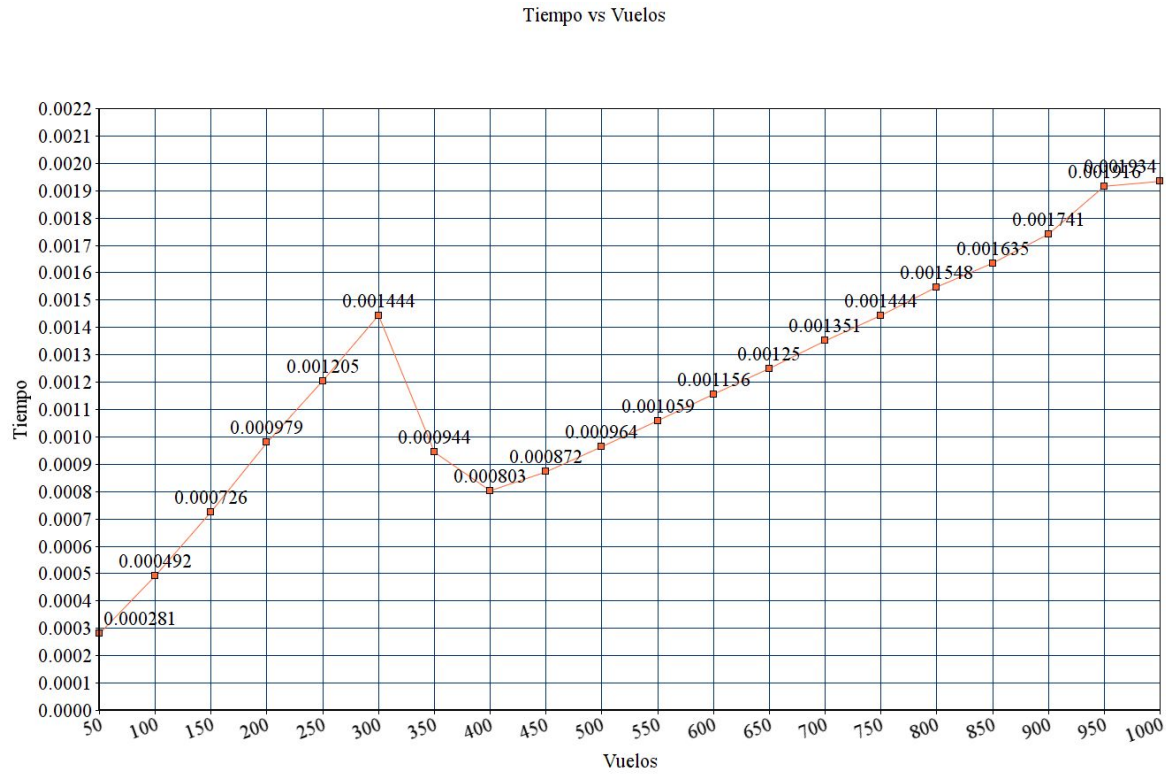
Nuestro análisis experimental se realizó en un caso de 3 ciudades fijas (Caso más básico)en donde el número de vuelos va variando de 50 en 50 hasta llegar al máximo número de vuelos que es 1000, ademas solo variamos la cantidad de días entre 1<= d <= 3.

Grafico 2



Se realizó en un caso de 5 ciudades fijas (Caso Promedio) en donde el número de vuelos va variando de 50 en 50 hasta llegar al máximo número de vuelos que es 1000, ademas solo variamos la cantidad de días entre 1<= d <= 3.

Grafico 3



Se realizó en un caso de 10 ciudades fijas (Casi peor caso)en donde el número de vuelos va variando de 50 en 50 hasta llegar al máximo número de vuelos que es 1000.
No es el peor caso, ya que en los datos ingresados no variamos mucho la cantidad de días en los cuales se podían realizar los vuelos, este valor lo hicimos variar entre 1 <= d <=3 mientras que el problema permitía variarlo entre 1 <= d <= 30.

Analizando los datos entregados por los 3 casos es posible observar que a medida que nos acercamos al peor caso la curva de los gráficos se asemeja a O(k*n^2) (Donde k es el número de vuelos y n el número de ciudades), lo cual fue lo obtenido por el análisis teórico del algoritmo.

Conclusión

Nuestro trabajo se podía resolver con otros algoritmos de búsqueda de camino mínimo. Nos decidimos por Bellman Ford, ya que encontramos que su implementación era más sencilla en comparación a los demás.

De todas maneras la creación del input se hizo un poco difícil, ya que para ciertos casos este era bastante grande dada la condición de tener n(n-1) horarios de vuelo, por ejemplo en el caso de 10 ciudades debíamos generar 90 horarios diferentes.