Programación III

Trabajo Práctico Especial

Parte II

Integrantes: Carpinetti, Heroel - Careri, Nicolas

Github: https://github.com/HeroelC/tpeProgramacionIII

Fecha de entrega: Viernes 14 de junio de 2019

Lugar de entrega: Laboratorio de ISISTAN

Introducción

Un organismo de seguridad aéreo desea supervisar el correcto funcionamiento de todos los aeropuertos habilitados. Para esto, partiendo de un aeropuerto cualquiera debe visitar todos los aeropuertos restantes y retornar al aeropuerto de origen.
Por una cuestión de costos, este recorrido debe realizarse visitando cada aeropuerto una única vez y viajando la menor cantidad posible de kilómetros totales.

Alternativa 1: Solución utilizando algoritmo greedy

Dificultades a nivel implementación

Al ser un algoritmo greedy una de las características que tiene es que son fáciles de implementar, este toma la mejor decisión que tiene disponible y no vuelve a replantearse la decisiones que ya fueron tomadas.

Limitaciones

La búsqueda de un óptimo local no implica encontrar un óptimo global.

Si el grafo no está muy conectado puede que nunca llegue a una solución.

Ventajas

Fáciles de implementar.

Producen soluciones eficientes.

A veces encuentran la solución óptima.

Contexto de aplicación

Normalmente se aplica a los problemas de optimización. Por lo tanto se aplicaría cuando se necesite realizar una búsqueda por la cual se sigue una heurística que consistente en elegir la opción óptima en cada paso local con la esperanza de llegar a una solución general óptima.

Calidad de la solución

En este caso la estrategia greedy no da una solución óptima siempre.

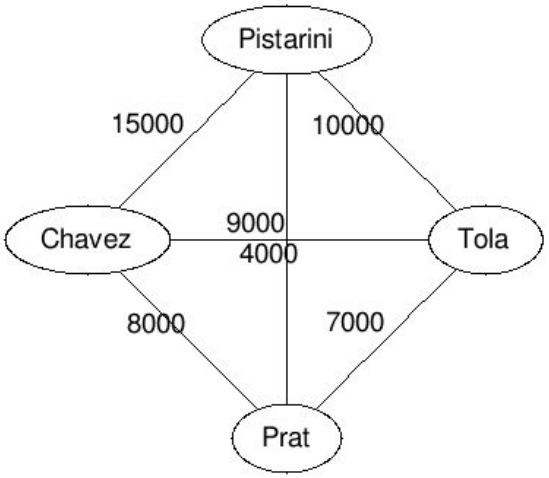
Explicación de las decisiones de implementación realizadas y las estructuras utilizadas

A la hora de implementar este algoritmo , se decidió iterar la lista de adyacencia del aeropuerto origen, optando por la de menor longitud y desplazandonos al aeropuerto destino de la ruta seleccionada repitiendo secuencialmente estas decisiones en cada paso. Cada aeropuerto visitado, se lo agrega a una lista denominada "caminoActual" y se acumulan las distancias entre dichos aeropuertos. Para ir guardando el recorrido de aeropuertos que va siguiendo el algoritmo se decidió usar una LinkedList que nos provee dinamismo a la hora de insertar y remover nodos frecuentemente. Una vez visitados todos los aeropuertos el algoritmo finaliza. A continuación se podrá un observar un pseudocódigo de la parte iterativa del algoritmo.

```
MIENTRAS (hayan aeropuertos por visitar){
        Ruta = seleccion(aeropuerto); //aeropuerto principalmente es el origen
        kmTotales.sumarDistancia(Ruta)
        caminoActual.agregar(ruta.getDestino());
        Aeropuerto = ruta.getDestino(); //desplazamiento
        Aeropuerto.MarcarVisitado();
}
solucion.add(kmTotales, caminoActual);
```

Análisis del seguimiento

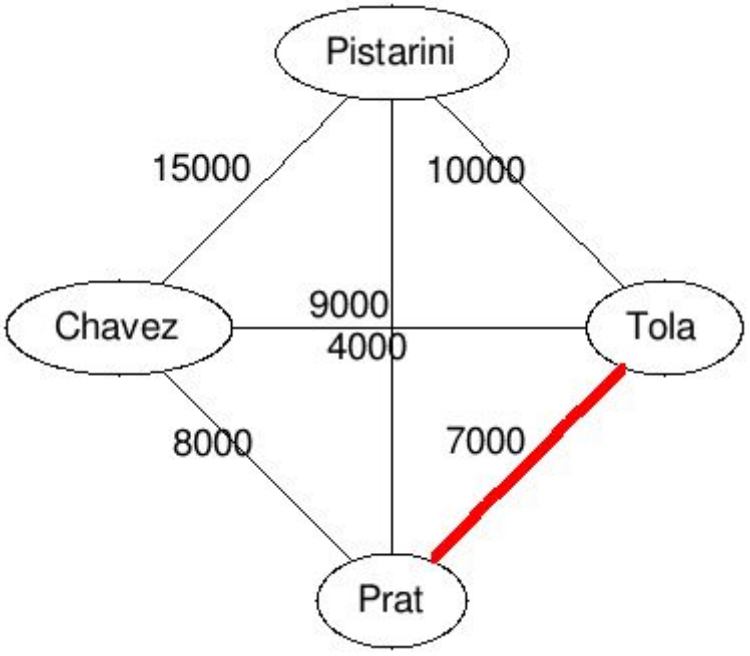
Grafo de ejemplo para el seguimiento del greedy (Ver figura 1)



Inicio (Figura 1)

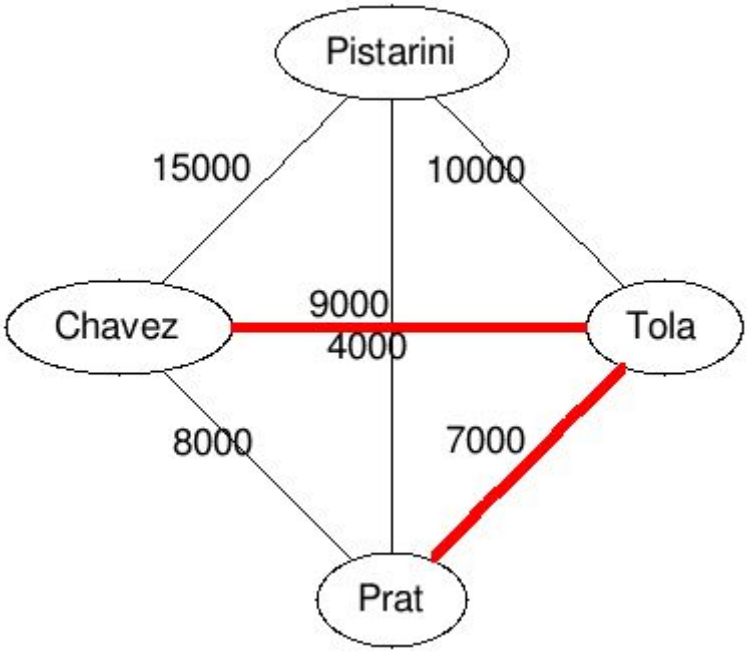
Origen de donde se desea partir: Prat

Se procede a marcar como no visitado todo los aeropuertos. Luego se procede a la selección del candidato (éste será la ruta que posea menor distancia) y se marca el origen como visitado.
La selección va a dar como resultado ir a Tola (Ver figura 2)



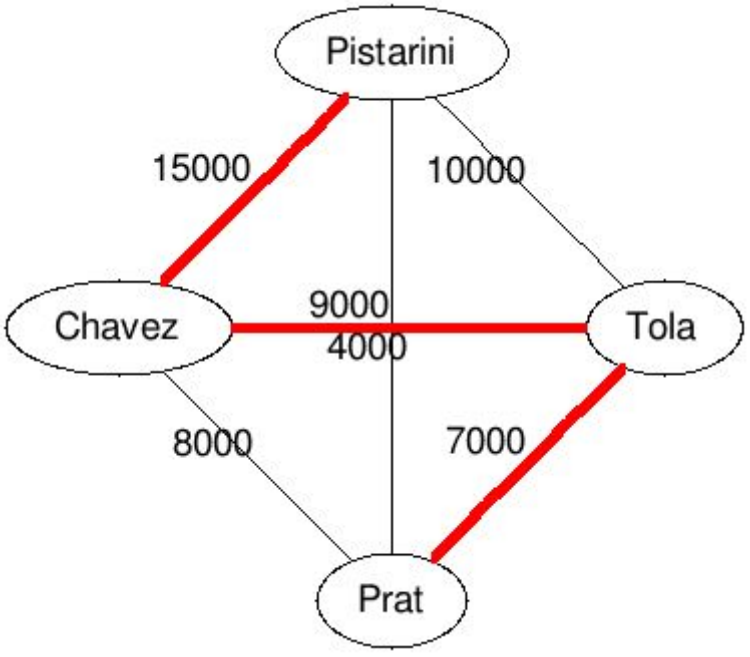
Selección de la ruta desde el Prat (Figura 2)

Una vez que se encuentra en la ciudad de Tola, este debería realizar la selección de la ruta más conveniente (la que conlleve menor recorrido en kilómetros) y no esté visitada. La selección será Chavez (Ver figura 3)



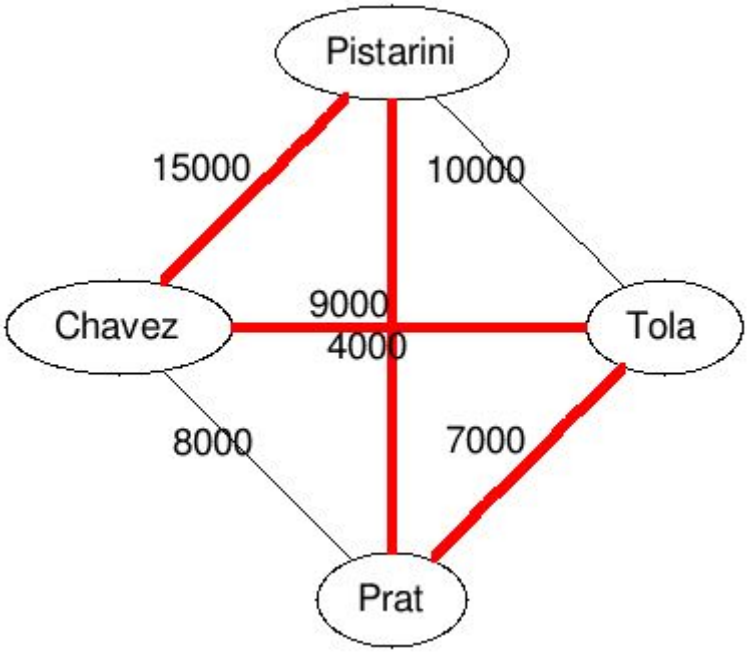
Selección de la ruta desde Tola (Figura 3)

Desde Chávez se procederá a seleccionar la ruta con menor distancia y se marcara a este como vistado (Ver figura 4)



Selección de la ruta desde Chávez (Figura 4)

En este caso la ruta con mejor distancia es Prat, pero el algoritmo reconoce que ya fue visitado, así que sigue con la que mejor le conviene.



Selección de la ruta desde Pistarini (Figura 5)

Dando como resultados el siguiente recorrido:

Prat - Tola - Chávez - Pistarini - Prat

7000 km + 4000 km + 15000 km + 9000 km = 35000 km

Cuando en realidad el recorrido más óptimo hubiera sido:

Prat - Pistarini - Tola - Chávez - Prat

9000 km + 10000 km + 4000 km + 8000 km = 22000 km.

Alternativa 2: solución utilizando backtracking

Dificultades a nivel implementación

Limitaciones

Consume mucha memoria para tener que almacenar los ciclos de búsqueda e ineficiencia al encontrar todos los caminos posibles dentro del grafo.

Ventajas

Halla la mejor solución disponible.

Es relativamente sencillo de implementar en los problemas a resolver.

Contexto de aplicación

Normalmente se aplica cuando se necesita hallar todas las soluciones a un problema, encontrar la solución más óptima de un problema, o encontrar al menos una solución al problema.

Calidad de la solución

En este caso la estrategia backtracking nos brinda una solución (si es que la hay) óptima siempre.

Explicación de las decisiones de implementación realizadas y las estructuras utilizadas

Para implementar el backtracking , se tuvo que buscar una alternativa que encuentre todos los caminos posibles dentro del grafo para así poder quedarse con la más grande, para ello por cada estado del algoritmo se pregunta si se visitaron todos los nodos y en caso de ser así, se pregunta si en el que se está parado se puede llegar al principio y si se cumple la condición nos quedamos con el camino de mayor kilometraje acumulado hasta el momento.

En caso de no llegar a un estado final, es decir en que no se hayan visitado todos los nodos, el algoritmo se mete recursivamente en las adyacencias del aeropuerto agregando este a la lista del recorrido que va haciendo el algoritmo y acumulando la distancia de las rutas entre los aeropuertos que va recorriendo el mismo.

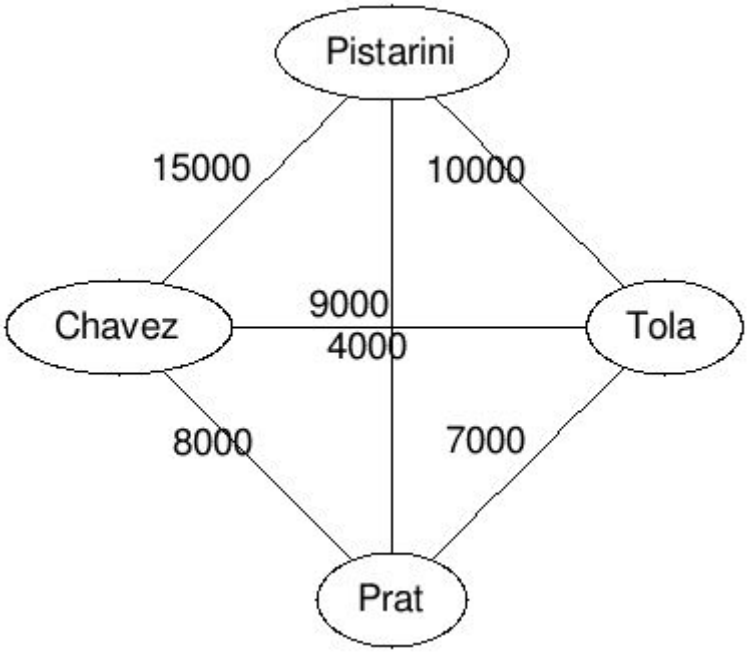
Una vez saliendo de la recursión retrocedemos desmarcando como visitado el aeropuerto y sacándolo de la lista del recorrido.

A continuación se puede observar un pseudocódigo del backtracking:

```
back(origen, caminoActual, principio, km) //en el estado inicial el principio es igual al origen
origen.MarcarVisitado();
caminoActual.agregar(origen);
PARA (cada ruta adyacente al origen)
        km.sumarDistancia(ruta);
        SI (ruta.getDestino() es igual al principio del recorrido)
                SI (se visitaron todos los aeropuertos)
                        SI (el nuevo camino es más corto que el anterior)
                                solución.agregar(km, caminoActual);
        SÍ NO
                        SI (el siguiente adyacente no está visitado)
                                back(ruta.getDestino(), caminoActual, principio, km);
        km.restarDistancia(ruta);
origen.marcarNoVisitado();
caminoActual.remover(origen);
```

Análisis del seguimiento

Grafo de ejemplo para el seguimiento del backtracking (Ver figura 6)



Grafo de ejemplo (Figura 6)

Origen de donde se desea partir: Prat

Supongamos que el backtracking se encuentra en el final del recorrido del greedy (Ver figura 7)

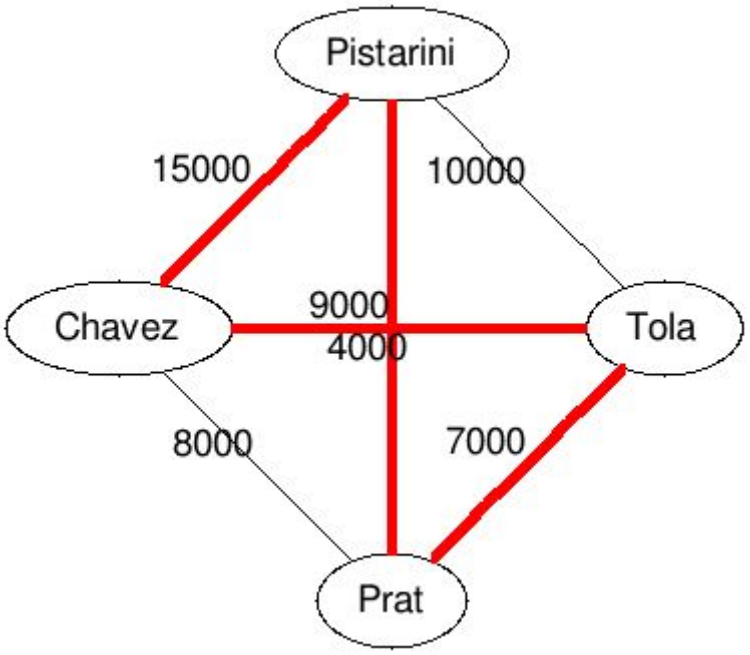
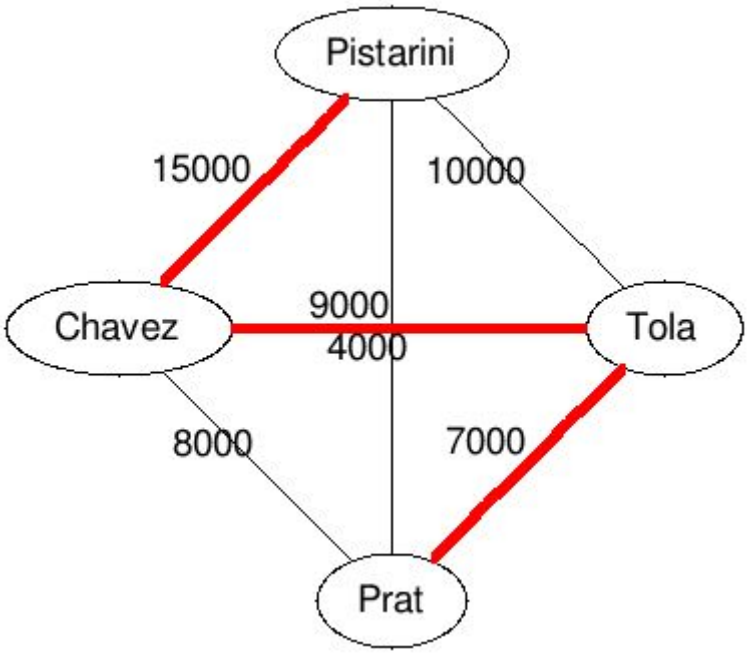
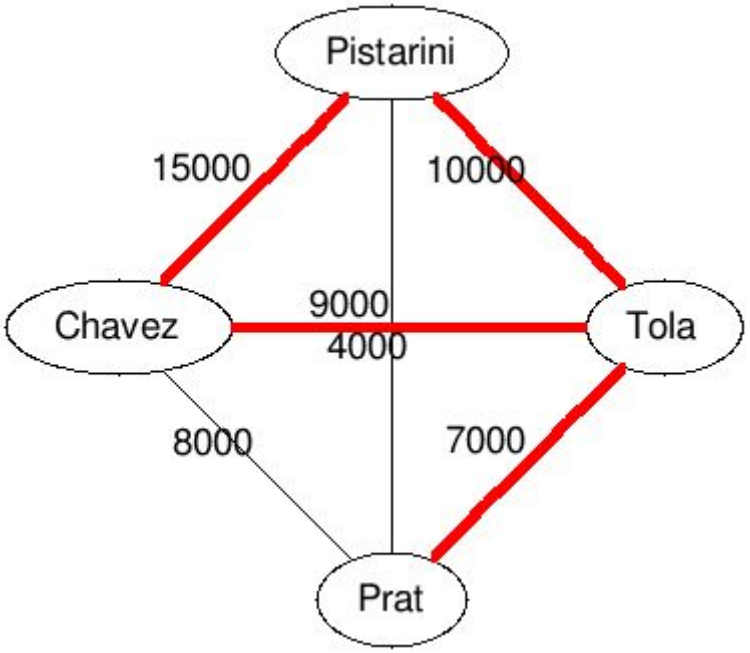


Figura 7

A diferencia del greedy que terminaría en ese momento, este sigue buscando todas las combinaciones posibles. Este va deshaciendo el último paso que se realizó y probando si existe otro posible camino. (Ver figura 8 y 9)



Se deshace la última selección (Figura 8)



Elección de otra ruta (Figura 9)

Luego se debería evaluar si es estado final y si ese estado final es solución al problema que estamos buscando. De ser así se debe evaluar si la solución que se consiguió con anterioridad es mejor o peor que la que acabamos de encontrar si es mejor, el algoritmo se quedaría con esta hasta que encuentre otra que sea mejor.

Conclusiones

La conclusión a la que llegamos es que dependiendo el problema y el resultado que se desee obtener es muy importante saber elegir el algoritmo que se debe aplicar, porque se pueden estar utilizando recursos de manera ineficiente.

Bibliografía

Universidad de granada - Algoritmos greedy
https://elvex.ugr.es/decsai/algorithms/slides/4%20Greedy.pdf

Universidad Rey Juan Carlos - Algoritmos Voraces
http://www.cartagena99.com/recursos/alumnos/temarios/Algoritmos_voraces.pdf

Material de la cátedra - Clase 4: Algoritmos greedy
https://docs.google.com/viewer?a=v&pid=sites&srcid=YWx1bW5vcy5leGEudW5pY2VuLmVkdS5hcnx0dWRhaS0yLTF8Z3g6N2IzNDVlMzE5YTA5MmVjZg

Wikipedia - Algoritmo voraz
https://es.wikipedia.org/wiki/Algoritmo_voraz

Wikipedia - Vuelta atrás
https://es.wikipedia.org/wiki/Vuelta_atr%C3%A1s