Diseño y análisis de algoritmos

Práctica 1

Mario Valdemaro García Roque
Alberto Cabello Álvarez

Cuestiones sobre los ejercicios

1. ¿Cuál es el coste teórico del algoritmo de Dijkstra? Justificar brevemente dicho coste.

El coste teórico es: O((V+E)*log(V)) aunque se podría resumir como que el coste del algoritmo es O(E*log(V))

Puesto en detalle:

> La operación que mas nos limita es O(E*log(V)), esta operación surge de buscar el camino mas corto como tal, este coste se debe a que encontrar el camino mas corto dado un vértice es log(V) pero al tener que repetirse esta operación para todos las aristas entonces tendremos que el coste total sera el que hemos puesto al principio.
>
> El coste derivado de O(V*log(V)) se debe al uso de la cola de prioridad.

2. ¿Cuál es el coste teórico del algoritmo de Dijkstra iterado para encontrar las distancias mínimas entre todos los vértices de un grafo? ¿Cuál es el coste teórico del algoritmo Floyd–Warshall para la misma tarea?

El coste teórico de Dikstra iterado será el propio coste de Dijkstra multiplicado por V, que es el número de nodos que hay en el grafo, es decir, aplicar Dijkstra para cada vértice del grafo. Por tanto el coste resulta:

$$O(V*((V+E)*\log(V)))$$

Ahora el factor importante, que varía para grafos del mismo tamaño de vértices, es cuánto es el valor de E. El valor de E es el numero de aristas que hay en un grafo, es decir, es el numero de caminos entre vértices. Será interesante expresar el numero de vértices en función de un factor de densidad, que consideraremos que estará entre un 0% y un 100%, significando que si hay un 100% de factor de densidad todos los nodos se conectan con todos los nodos. Por lo que nuestra función la podemos expresar ahora como una función que depende del numero de vértices y del factor de densidad.

$$O(V*((V+V^2*SF)\log(V))$$

El coste del algoritmo de FW es cúbico como ya sabemos, siendo mas específicos su coste es:

$$O(|V|^3)$$

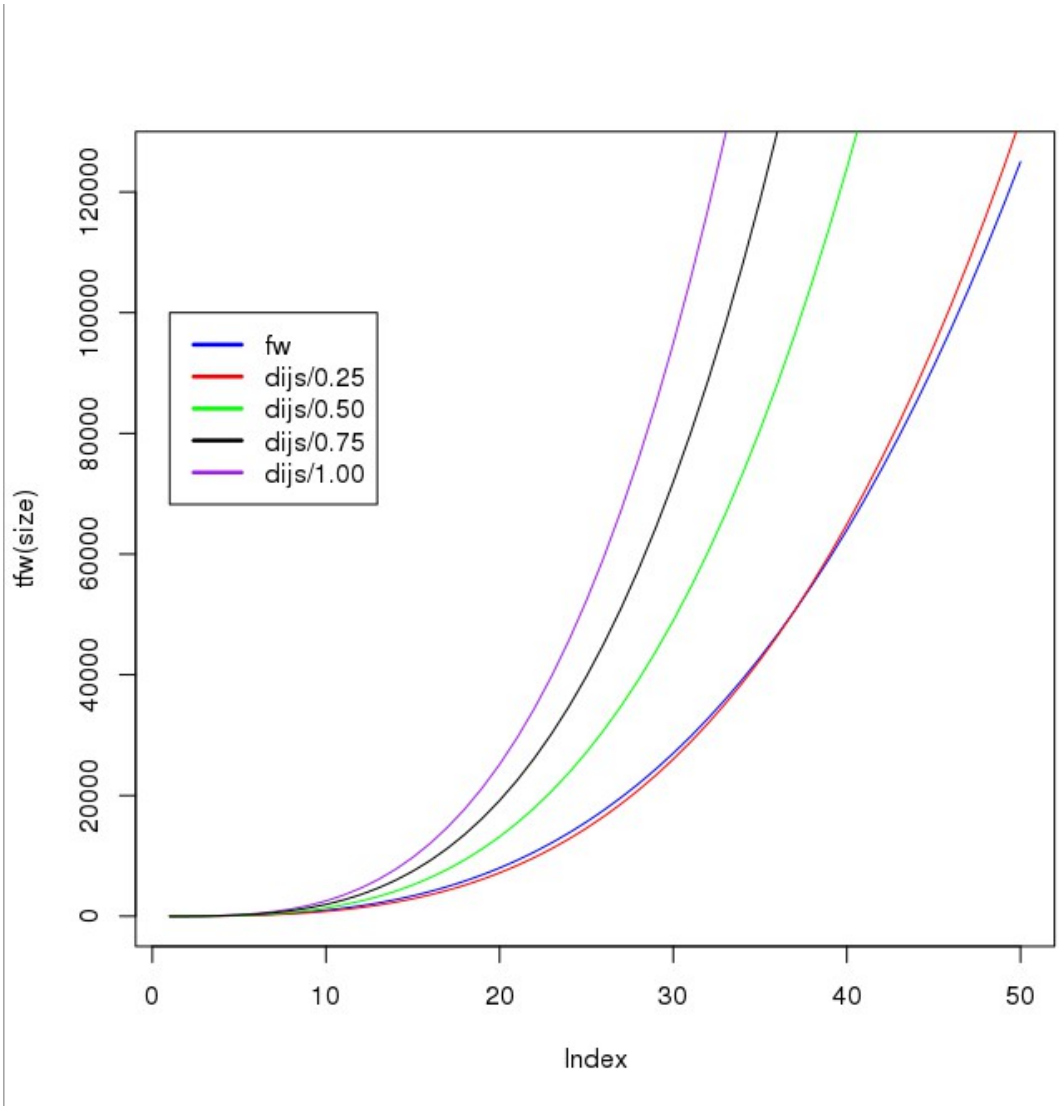
Siendo este coste igual para cualquier tipo de grafo ya que Floyd Warshall itera sobre todos los posibles caminos, existan o no.

3. Suponiendo que se va a trabajar con grafos con un sparsity factor ρ, ¿para qué valores de ρ y del número de nodos N debería ser Floyd–Warshall más competitivo que Dijkstra iterado? Contrastar las conclusiones obtenidas con ejemplos y graficas concretos.

Ejecutamos el modelo teórico y el practico:

Para el modelo teórico usamos 5 funciones, 1 de FW, ya que el factor de densidad no hace que varíen , y 4 de Dijkstra con distintos factores de densidad (0.25,0.50,0.75,1.00). Para la función de FW generaremos los valores con una función cubica, y para la función de Dijkstra usaremos los valores teóricos que comentamos arriba, es decir:

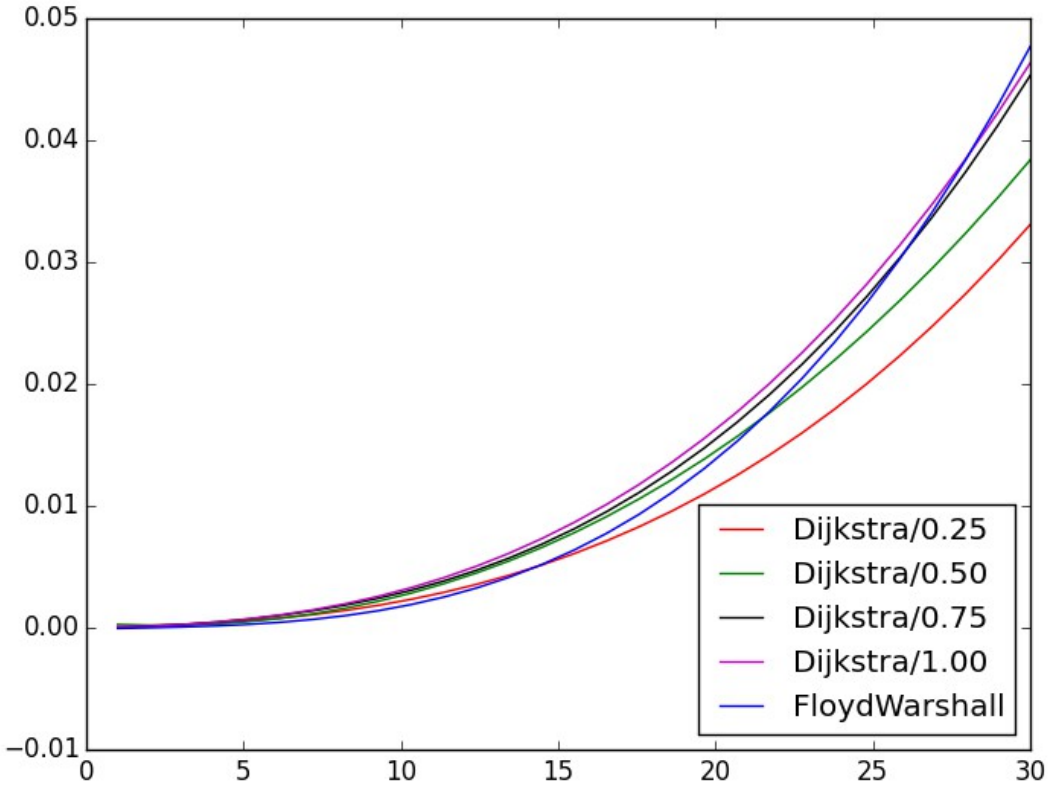
$$f(V,SF)=V*((V+V^2*SF)\log(V)$$



Observamos que los resultados de los tiempos teóricos indica que FW es mas rápido para estos tipos de gráficos que están muy poblados. Sin embargo, para grafos menos densos, como el caso de Dijkstra con factor de densidad 0.25, cuanto menos denso mejores resultados obtendremos, llegando a ser mejor utilizar Dijkstra si el grafo esta extremadamente poco poblado. He aquí el problema, ya que los grafos pocos densos tienen poco interés analítico por tener pocas posibilidades de recorrelos.

A continuación obtenemos los valores de muestro reales que hemos obtenido:

Para realizar esta gráfica hemos generado grafos de entre 1 y 30 nodos para Dijkstra y FW y hemos usado distintos factores de densidad para los valores de Dijkstra (0.25,0.50,0.75,1.00).



Observamos algo curioso y es que a pesar de los resultado que hemos obtenido en los datos teóricos los datos prácticos nos dicen otra cosa, ya que obtenemos que Dijkstra es mejor que FW para todos los casos que hemos probado. Aunque esto se puede deber a distintos factores lo mas probable es que la operación básica de uno y otro algoritmo tengan una diferencia de tiempos significativa.

Los puntos donde obtenemos los cortes de la gráficas son:

- Densidad 0,25: grafos de alrededor de 14 vértices.
- Densidad 0,50: grafos de alrededor de 21 vértices.
- Densidad 0,75: grafos de alrededor de 26 vértices.
- Densidad 1,00: grafos de alrededor de 26 vértices.

4. ¿Como se podrían recuperar los caminos mínimos si se utiliza Dijkstra iterado? ¿Y si se usa Floyd–Warshall?

Para el algoritmo de Dijkstra, basta con una vez encontrado el camino más corto, hacer backtracking del destino hacia sus predecesores. De esta manera, recorremos la lista de predecesores hacia atrás, obteniendo el camino.

Una opción para FW sería mantener una segunda matriz con los vertices predecesores. De esta manera, cada vez que se actualiza el coste de un camino a uno inferior, también se actualiza el nodo padre.

```
[...]
for j in range(N):
    if d[i][j] > d[i][k] + d[k][j]:
        d[i][j] = d[i][k] + d[k][j]
        parent[i][j] = parent[k][j]

[...]
```

Una vez construida la matriz, se puede seguir un procedimiento similar a dijkstra para reconstruir el camino.