MA3705. Algoritmos Combinatoriales. 2014.
Profesor: José Soto
Escriba(s): Pedro Espinoza, Felipe Salas y Pablo Ugalde
Fecha: 22 de Septiembre de 2014.

Cátedra 12

Algorithm 1 Algoritmo Dijkstra

INPUT: $G = (V, E)$ digrafo, $\ell : E \to \mathbb{R}_+$
for $v \in V$ **do**

$$D[v] = \begin{cases} 0 & \text{si } v = s. \\ \ell(s, v) & \text{si } (s, v) \in E. \\ +\infty & \text{si no.} \end{cases}$$

$$\Pi[v] = \begin{cases} \perp & \text{si } v = s. \\ s & \text{si } (s, v) \in E. \\ \perp & \text{si no.} \end{cases}$$

end
$U = s.$
$F = \varnothing.$
while $U \neq V$ **do**
  if $D[w] = +\infty\ \forall w \in V \setminus U$ **then**
    $U \leftarrow V$.
  end
  Elegir $u^* \in U \setminus V$ con $D[u^*]$ mínimo.
  $U \leftarrow U + u^*$.
  $F \leftarrow F + (\Pi(u^*), u^*)$.
  for $w \in V \setminus U, (u^*, w) \in E$ **do**
    if $D[w] > D[u^*] + \ell(u^*, w)$ **then**
      $D[w] \leftarrow D[u^*] + \ell(u^*, w)$.
      $\Pi[w] \leftarrow u^*$.
    end
  end
end
return $H = (U, F)$.

Notación. Si T es un digrafo que contiene un único a-b dicamino, entonces denotamos a tal dicamino como aTb.

Teorema 1. *En cada paso:*

(i) $H = (U, F)$ *es una arborecencia con raíz en* s*, tal que para todo nodo* v *en* U*, el único camino de* s *a* $v \in H$*, denotado por* (sHv) *es de largo mínimo en* G.

(ii) $\forall v \in U, D[v] = \ell(sHv) = d(s, v)$.

(iii) $\forall v \in U, D[v] = \min\{\ell(P) : P \text{ es } s\text{-}v \text{ camino en } G \text{ con } V(P) - v \subseteq U\}$
(de hecho, si $w = \Pi[v] \in U$ *(y* $D[v] < \infty$*), entonces:* $D[v] = \ell(sHwv)$*)*

Observación. *La clase pasada probamos* (i) *y* (ii) *por inducción en* $i = |U|$.

Probemos (iii), el caso $i = 1$ es trivial.

- $i \geq 2$: Sea u^* el i-ésimo nodo que Dijkstra elige
  Llamemos U, D, H, Π a los objetos al final de la iteración en la que u^* es agregado a U, y llamemos U', D', H', Π' a los objetos en la iteración anterior. En particular, $U = U' + u^*$.
  Sea $v \in V \backslash U$, consideremos P un s-v camino con $V(P) - v \subseteq U$ de largo mínimo.

  Observación. *De todos los P posibles, elijamos uno donde u^* aparece lo más tarde posible (o no aparece).*

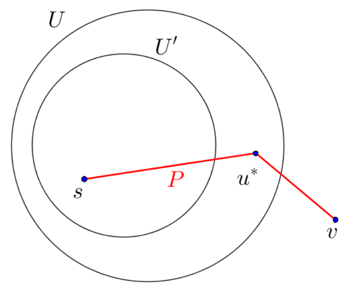


Figura 1: P donde u^* aparece lo más tarde posible.

Afirmación. Si u^* está en P, entonces u^* es el penúltimo nodo de P.

Demostración. Si no fuera el penúltimo nodo de P:
Sea w el sucesor de u^* en P. Como P es mínimo, sPw es de largo mínimo en G.
Como $w \in U'$, $\ell(sPw) = d(s, w) = \ell(sH'w)$ (por (ii)). Concatenando $sH'w$ con wPu, tenemos un paseo Q con $V(Q) - v \subseteq U'$ (todos los vértices de Q contenidos en U').
Luego Q contiene un s-v camino Q' con todos sus vértices (excepto v) en U' donde u^* no aparece, y $\ell(Q') \leq \ell(P)$, es decir, encontramos un s-v camino más corto, lo que contradice el hecho de que P es de largo mínimo. $\square$

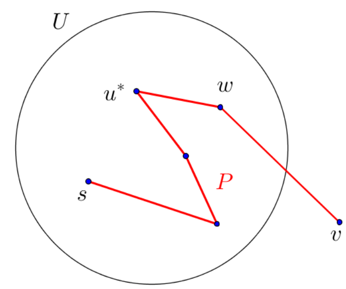


Figura 2: P con w sucesor de u^*.

Usemos la afirmación para probar (iii):

Demostración. (iii): Cuando u^* entra a U, $D[v] = min\{D'[v], D'[v] + \ell(u^*, v)\}$.
Si P no toca a u^*, por inducción, $\ell(P) = D'[v]$.
Si P toca a u^*; $\ell(P) = \ell(sPu^*) + \ell(u^*, v) = D'[v] + \ell(U^*, v)$.
Por lo tanto, $D[v] = \ell(P)$. $\square$

Probamos así que el algoritmo de Dijkstra es correcto, es decir que devuelve las distancias mínimas y los caminos más cortos de s a todos los nodos.

Corolario 1. *Dijkstra funciona y encuentra los caminos más cortos de s a todos los demás nodos.*

Una pequeña intuición sobre cómo funciona Dijkstra en cada paso:

1. Para todo $v \in V \backslash U$, calcular:
   $D[v] := \underset{x \in U}{min} = \ell(sHx) + \ell(xv)$.
2. Elegir el mínimo $D[v]$ y agregarlo a H.

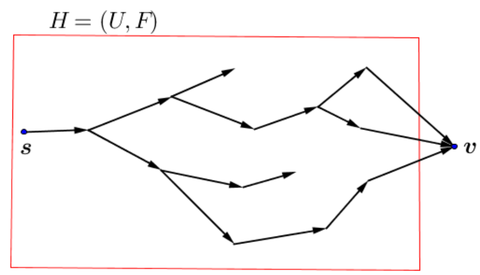


Figura 3: Caminos de s a v.

Complejidad

Analicemos la complejidad del algoritmo de Dijkstra.

- Inicializacioń : $O(n)$ pues se calcula el par $(D[v], \pi[v])$ para cada nodo.
- Desde linea 4 a linea 8: Calcular el mínimo toma $O(n)$.
- Desde linea 9 a linea 14: Actualizar los valores toma $O(|N^+(u^*)|)$.

Por lo tanto, en total la complejidad es $\sum_{v \in V} O(n + |N^+(u^*)|) = O(n^2 + m)$.
Es interesante notar que existen otras implementaciones de este mismo algoritmo que permiten mejorar la complejidad.

Definición. Llamaremos *cola de prioridad* a una estructura de datos en la que se pueden realizar las siguientes operaciones:

- Agregar un par (etiqueta, valor).
- Extraer mínimo.
- Cambiar el valor de alguna etiqueta.

Usando estas estructuras se puede reducir considerablemente la complejidad de este algoritmo, algunas implementaciones que se destacan son mediante *Heaps Binarios*, con la cual el algoritmo Dijkstra tiene complejidad $O(m \log n)$ y mediante *Heaps de Fibonacci* que logra una complejidad $O(m + n \log n)$.

Observación. *En la practica usar* Heaps binarios *es mucho más conveniente que usar* Heaps de Fibonacci*, debido a su fácil implementación y además garantiza $O(\log n)$ para todas las operaciones de cola de prioridad.*

Observación. *Para encontrar caminos mínimos desde s, usando colas de prioridad se tiene que:*

$$\text{Dijkstra es } O(m + n \log n). \qquad \text{Bellman-Ford es } O(n(m + n)).$$

Observación. *Si quisiéramos calcular las distancias entre todos los pares, se tiene que:*

$$\text{Dijkstra desde todos los nodos es } O(nm + n^2 \log n). \qquad \text{Floyd-Warshall es } O(n^3).$$

Observación. *Si G no tiene ciclos (dirigidos), Orden topológico (visto en auxiliar) es:*

$$O(n + m) \text{ desde algún } s. \qquad O(n^2 + mn) \text{ entre todos los pares.}$$

Observación. *Dijkstra funciona también en grafos no dirigidos con largos positivos. Para aplicar Dijkstra en estos grafos, basta reemplazar cada arista $e = \{u, v\} \in E$ por dos arcos antiparalelos (u, v) y (v, u) y asignarles $\ell(e)$ a ambos arcos. Para encontrar caminos mínimos en el grafo original, basta encontrar caminos dirigidos mínimos en el grafo dirigido auxiliar.*

Finalmente, es bueno notar que si todos los arcos (o aristas) tienen el mismo largo, digamos igual a 1, entonces podemos encontrar un árbol de caminos de costo mínimo a partir de s calculando simplemente un árbol de búsqueda horizontal (BFS). Como consecuencia, calcular caminos mínimos con respecto al número de arcos (o aristas), se puede hacer en tiempo $O(n+m)$, lo cual es incluso más rápido que Dijkstra.

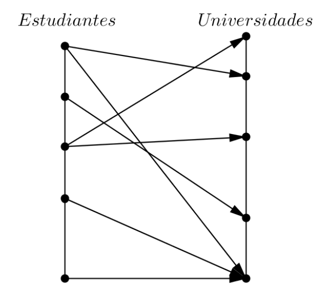
III. Matching

Comenzaremos recordando la definición de matching en un grafo no dirigido:

Definición. Sea $G = (V, E)$ un grafo no dirigido. Se dice que $M \subseteq E$ es un matching en G si $\nexists\, e, f \in M$ con $e \neq f$, tal que son incidentes en un mismo $v \in V$, o equivalentemente $\forall e, f \in M\ e \cap f = \emptyset$.

Motivación. Un ejemplo en donde se necesita encontrar un matching de cardinalidad máxima es el siguiente.

Consideremos un grafo cuyos vértices son todos los estudiantes y las universidades, y las postulaciones de un estudiante a una universidad son sus aristas. El objetivo es encontrar el matching que encuentre la combinación con más postulaciones, dependiendo del cupo de cada universidad
Notemos que hay muchos problemas similares a este.



En el contexto del problema de matching de cardinalidad máxima surgen dos preguntas: Dado un matching M de un grafo G,

1. ¿Cómo certificar que M no es de cardinalidad máxima?
2. ¿Cómo certificar que M es de cardinalidad máxima?

En respuesta a la primera pregunta bastaría encontrar otro matching más grande. La respuesta a la segunda pregunta es en general un problema difícil al cual se le intentará dar una respuesta.

Definición. Dado M matching, $v \in V$ se dice M-cubierto si $\exists e \in M$ tal que e es incidente en v. En caso contrario v se dice M-expuesto.

Definición. Dado un camino simple $P = e_1 e_2 \ldots e_l$, donde $e_i \in E$, diremos que P es M-alternante en G si: $e_i \in M \Leftrightarrow e_{i+1} \notin M, \forall i \in \{1, \ldots, l-1\}$.

Observación. *Se define un ciclo alternante de forma análoga.*

Definición. Se dice que un camino P M-alternante es M-aumentante si los vértices extremos de P están M-expuestos.

Observación. *Si P es M-aumentante, entonces:* $|P \setminus M| = |P \cap M| + 1$.

Ahora nuestro problema a resolver será el de encontrar un matching de cardinal máximo, para esto nos será útil el siguiente teorema:

Teorema 2. *M es maching de cardinal máximo $\Leftrightarrow \nexists$ un camino M-aumentante.*

Antes de probar el teorema estudiemos un lema que nos ayudará en la demostración de este:

Lema 1. *Dado M matching. Si P es M-aumentante $\Rightarrow M' = M \Delta P$ es un matching tal que $|M'| = |M| + 1$.*

Demostración. Tenemos que $|M'| = |M| + 1$, pues $|P \setminus M| = |P \cap M| + 1$. Además M' es matching, ya que si no lo fuera, tomando $e, f \in M'$, $e \neq f$ y ambos incidentes en v, por la alternancia de P tenemos que sin pérdida de generalidad $e \in M \setminus P \wedge f \in P \setminus M$. Luego, como P es M-aumentante, $\exists g \in M$ incidente en v, con $g \neq e$, lo cual es una contradicción con que M sea matching. Por lo tanto M' es matching. □

Ahora probemos el teorema que enunciamos anteriormente:

Demostración.

- $\Rightarrow$ Directo del lema pues si existiese un camino M-aumentante, en virtud de lo probado, M' es un matching de cardinal mayor estricto que M.

- $\Leftarrow$ Por contradicción supongamos que M' es matching tal que $|M'| > |M|$. Entonces dado $N = M'\Delta M$ tendremos que en cada vértice de G inciden a lo más dos aristas de N, de esta forma nos damos cuenta de que N es una colección de caminos o ciclos disjuntos que alternan sus aristas entre M y M'. Finalmente tendremos que como $|M'| > |M|$, entonces existe un camino P en $M'\Delta M$ tal que $|M' \cap P| > |M \cap P|$. De aquí se deduce que P es M-aumentante, llegando así a una contradicción.

□