Single-Source Shortest Paths “Camino más corto desde/a una fuente”

Agustín J. González

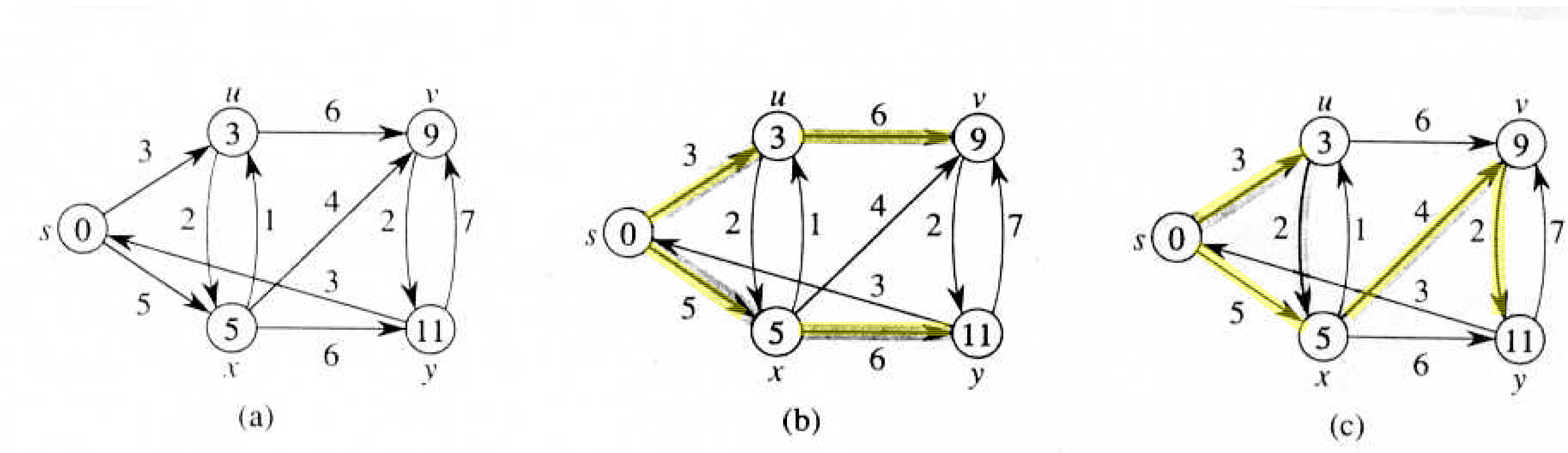
ELO320: Estructura de Datos y Algoritmos

1.er. Sem. 2002

Introducción

- Un estudiante quiere buscar el camino más corto que le permita llegar desde su casa a la de la polola (pololo) pasando sólo por avenidas (para hacerlo doble sentido).
- Un computador debe determinar la ruta más conveniente para un paquete hacia su destino.
- Se trata entonces de buscar la ruta más "económica" para ir desde un nodo a cada uno de los otros. Se trata de buscar las rutas de menor costo a cada uno de los nodos.
- El algoritmo breadth-first search obtiene la ruta más corta en grafos sin peso, en el cual cada arco se puede considerar de peso unitario.
- Variantes:
  Camino más corto a un destino único
  Camino más corto entre dos nodos cualquiera.

Ejemplos de Rutas más cortas desde una fuente



Grafo original

Una solución

Otra solución

Observaciones

- Lema: Subcaminos del camino más corto son caminos más cortos.
  Dado un grado dirigido con peso G=(V,E), sea p = <$v_1,v_2,v_3,..,v_k$> el camino más corto desde v_1 a v_k, para $1 \leq i \leq j \leq k$ sea p_{ij} = <$v_i,v_{i+1}, ,, v_j$> el subcamino de p desde el vertice v_i al v_j. Entonces p_{ij} es un camino más corto para ir de v_i a v_j.
- “Relajación” a través de un arco:
  Sea d[v] la estimación para la distancia más cortas desde el nodo fuente.
  Al estudiar e arco (u,v) podemos mejorar la estimación dependiendo si la ruta vía u es mejor. Esta operación es conocida como “Relajar”. El algoritmo es:

```
Relax (u,v, w){
        if (d[v] > d[u] + w(u,v) ){
                d[v] = d[u] + w(u,v);
                p[v] = u;
        }
}
```

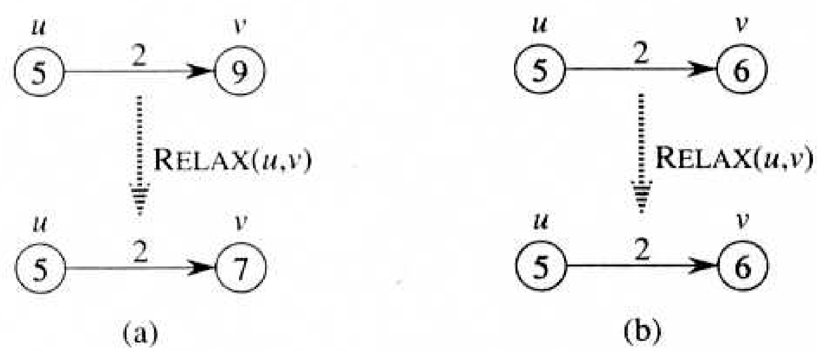


Figure 25.3 Relaxation of an edge (u, v). The shortest-path estimate of each vertex is shown within the vertex. **(a)** Because $d[v] > d[u] + w(u, v)$ prior to relaxation, the value of $d[v]$ decreases. **(b)** Here, $d[v] \leq d[u] + w(u, v)$ before the relaxation step, so $d[v]$ is unchanged by relaxation.

Algoritmo de Dijkstra

- Se supone que todos los arcos tienen peso no negativo. ¿Qué pasa si hay pesos negativos?
- Dijkstra(G, w, s) {

```
Dijkstra(G, w, s) {
        for (cada vértice v en V[G] ) {
                d [v] = infinito; /* -"MAX_INT" por ejemplo*/
                p [v] = NIL;
        }
        d [s] = 0;
        S = {};   /* S Contiene el arreglo de los nodos cuyo camino más corto ya ha sido encontrado */
        Q = V [G];
        while (Q != {}) {
                u = Extract_Min(Q);
                S = S ∪ {u};
                for ( cada vértice v en adj[u] )
                        Relax(u,v, w);
        }
}
```

- El tiempo de ejecución depende de la implementación de la cola de prioridad. Si se usara un arreglo lineal, Extract_Min toma O(V) con lo cual el algoritmo toma $O(V^2 + E) = O(V^2)$.
  Si se usa un heap binario Extract_Min toma O(lg V) con lo cual se reduce el tiempo total a =O(E lg V).

Ejemplo de ejecución del algoritmo de Dijkstra:

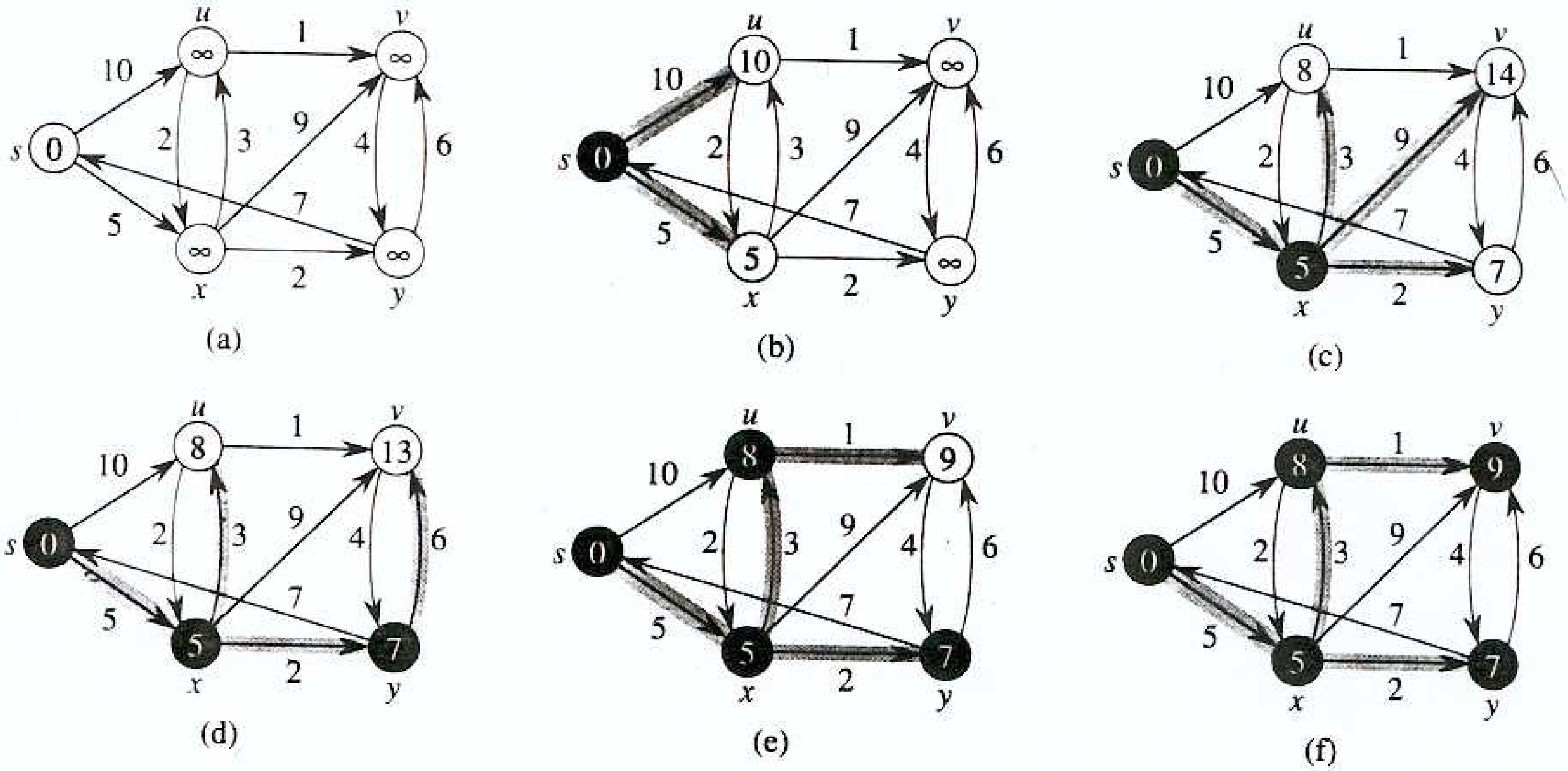


Figure 25.5 The execution of Dijkstra's algorithm. The source is the leftmost vertex. The shortest-path estimates are shown within the vertices, and shaded edges indicate predecessor values: if edge (u, v) is shaded, then $\pi[v] = u$. Black vertices are in the set S, and white vertices are in the priority queue $Q = V - S$. **(a)** The situation just before the first iteration of the **while** loop of lines 4–8. The shaded vertex has the minimum d value and is chosen as vertex u in line 5. **(b)**–**(f)** The situation after each successive iteration of the **while** loop. The shaded vertex in each part is chosen as vertex u in line 5 of the next iteration. The d and π values shown in part (f) are the final values.

Algoritmo de Bellman-Ford

- Se trata de resolver el mismo problema en el caso de pesos negativos y positivos.

```
Int Belman-Ford(G, w, s) {
        for (cada vértice v en V[G] ) {
                d [v] = infinito; /* -”MAX_INT” por ejemplo*/
                p [v] = NIL;
        }
        d [s] = 0;
        for (i=1 to |V [G]| -1 )
                for (cada arco (u,v) en E[G] )
                        Relax(u,v,w);
        for ( cada arco (u,v) en  E[G] )
                if(d [v] > d [u]+w(u,v))
                        return 0;  /* False; no existe camino mínimo */
        return 1;
}
```

- El tiempo de ejecución es O(EV).

Ejemplo: Algoritmo de Bellman-Ford

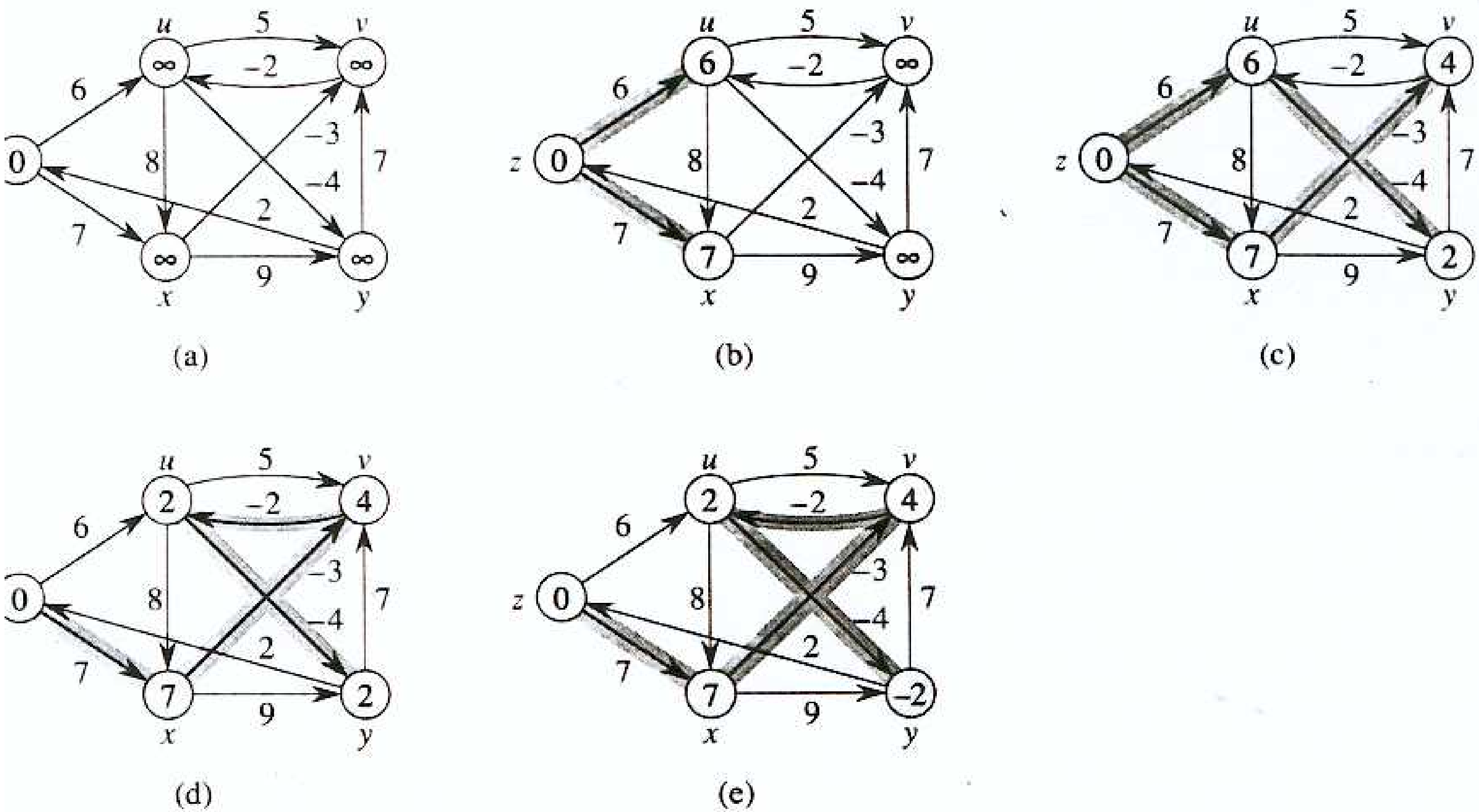


Figure 25.7 The execution of the Bellman-Ford algorithm. The source is vertex z. The d values are shown within the vertices, and shaded edges indicate the π values. In this particular example, each pass relaxes the edges in lexicographic order: $(u, v), (u, x), (u, y), (v, u), (x, v), (x, y), (y, v), (y, z), (z, u), (z, x)$. **(a)** The situation just before the first pass over the edges. **(b)–(e)** The situation after each successive pass over the edges. The d and π values in part (e) are the final values. The Bellman-Ford algorithm returns TRUE in this example.