S. Schwoon / D. Suwimonteerabuth | Winter Term 2008/09

Fundamental Algorithms

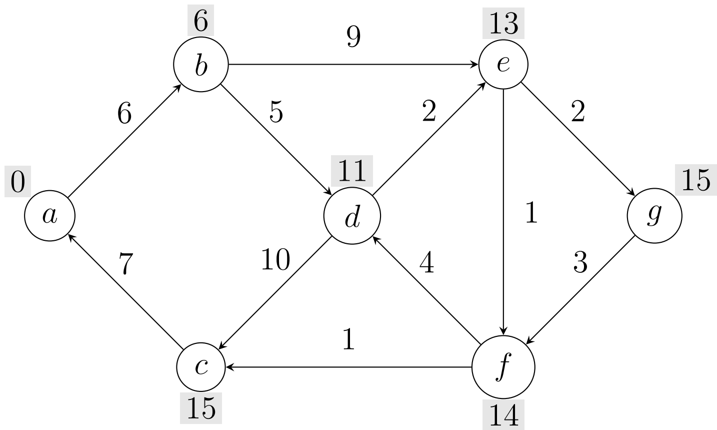
Solution Keys 9

1. Let w and v be arrays of length n denoting weights and values, respectively, for all object types. The maximum value can be obtained by calling the following procedure $\texttt{fill}(1, W)$.

   Procedure `fill`
   Input: object type i, weight r
   Output: the maximum value obtained from filling with object types i to n with total weight not exceeding r

   ```
   m := 0;
   for k = i to n do
       if w[k] ≤ r then
           m := max(m, v[k] + fill(k, r − w[k]));
       fi
   od
   return m;
   ```

2. (a)



   (b) The resulting distances are identical to (a). However, the distances for f and c are wrong because there is a shorter parth from e to f, i.e. via g.

   (c) The shortest path has infinite length, since the weight of the cycle d, e, f is negative. Therefore, one can always obtain shorter paths by repeating the cycle again and again.

   (d) In the following, we use a data structure called *queue*. In a queue, the first element added to the queue will be the first one to be removed. Given a queue and a node v, the operation $\texttt{enqueue}(v)$ adds v into the queue. The operation $\texttt{dequeue}()$, on the other hand, removes the first element from the queue.

   Given a graph and a node u, the following algorithm calculates the shortest distances from u to all nodes in the graph.

Input: graph (V, E), distance function x, node u
Output: array of distances from u

```
foreach v ∈ V do  d[v] = ∞;
d[u] = 0;

enqueue(u);
while queue not empty do
    v := dequeue();
    foreach w adjacent to v do
        if d[v] + x(v, w) < d[w] then
            d[w] = d[v] + x(v, w);
            if w not in queue then
                enqueue(w);
            fi
        fi
    od
od
return d;
```

Notice that the algorithm does not terminate when negative cycles are present. However, by stopping after any vertex has dequeued $|V| + 1$ times, termination can be guaranteed.

Since each vertex can dequeue at most $|V|$ times, it can be shown that the running time of the algorithm is $\mathcal{O}(|V| \cdot |E|)$.