In the name of God

Network Flows

3. Shortest Path Problems
3.2 Dijkstra's Algorithm

Fall 2010

Instructor: Dr. Masoud Yaghini

Dijkstra's Algorithm

- ***Dijkstra's algorithm***
  - finds shortest paths from the source node *s* to all other nodes in a network with ***nonnegative arc lengths***.
  - It is a ***label setting algorithm***
- Dijkstra's algorithm maintains a distance label $d(i)$ with each node i, which is an upper bound on the shortest path length to node i.
- At any intermediate step, the algorithm divides the nodes into two groups:
  - those which it designates as ***permanently labeled*** (or permanent)
  - those it designates as ***temporarily labeled*** (or temporary).

Dijkstra's Algorithm

- The distance label
  - to any permanent node represents the shortest distance from the source to that node.
  - to any temporary node represents an upper bound on the shortest path distance to that node.
- The basic idea of the algorithm is to fan out from node s and permanently label nodes in the order of their distances from node s.
- Initially, we give node s a permanent label of zero, and each other node j a temporary label equal to ∞.

Dijkstra's Algorithm

- At each iteration, the label of a node i is its shortest distance from the source node along a path whose internal nodes (i.e., nodes other than s or the node i itself) are all permanently labeled.
- The algorithm selects a node i with the minimum temporary label (breaking ties arbitrarily), makes it permanent, and reaches out from that node-that is, scans arcs in $A(i)$ to update the distance labels of adjacent nodes.
- The algorithm terminates when it has designated all nodes as permanent.

Dijkstra's Algorithm

- Dijkstra's algorithm maintains a ***directed out-tree T*** rooted at the source that spans the nodes with finite distance labels.
- The algorithm maintains this tree using predecessor indices [i.e., if $(i, j) \in T$, then $pred(j) = i$].
- The algorithm maintains the invariant property that every tree arc (i, j) satisfies the condition

$$d(j) = d(i) + c_{ij}$$

- with respect to the current distance labels.
- At termination, when distance labels represent shortest path distances, T is a ***shortest path tree***.

Dijkstra's Algorithm

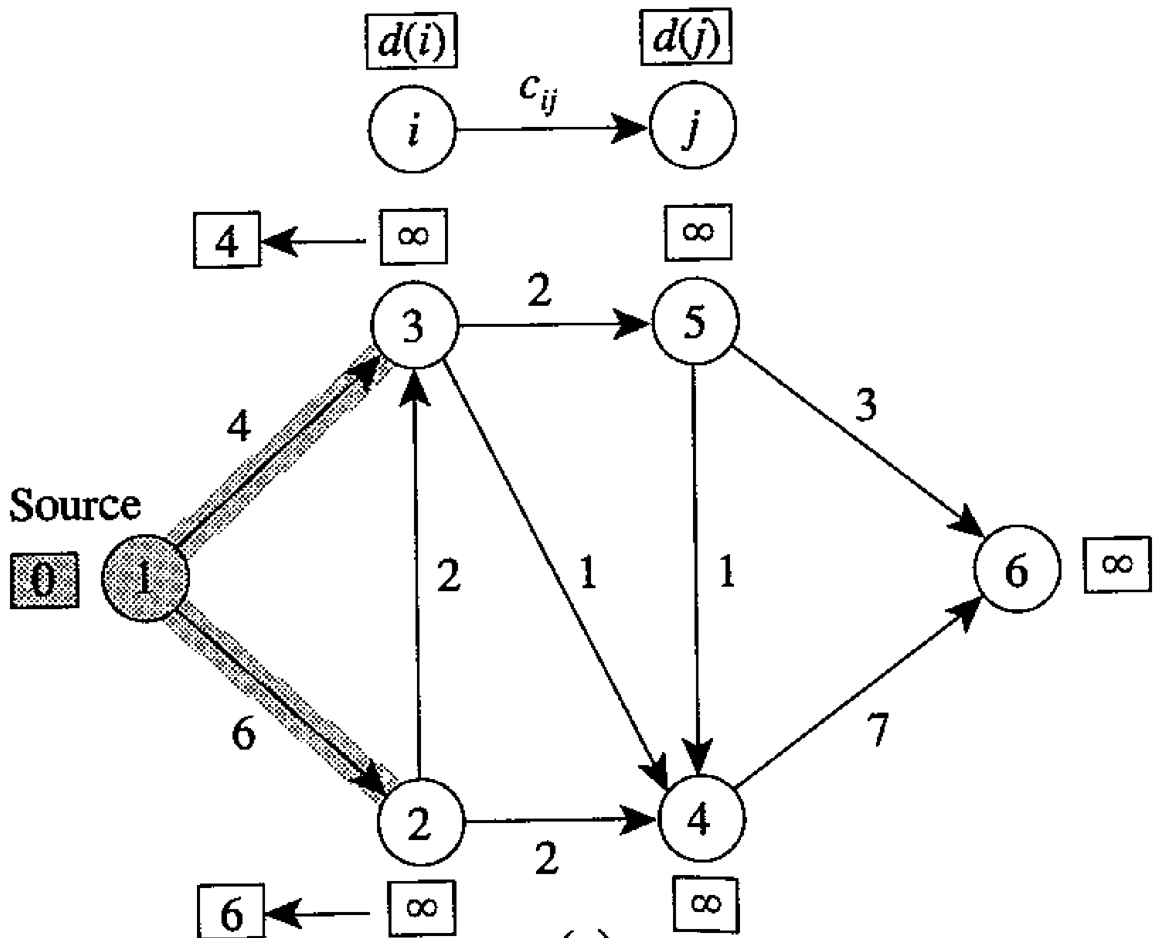
- Let
  - $d^*(j)$: denote the shortest path distance from node s to node j.
  - S : denotes the set of ***permanently labeled nodes***.
    - That is, $d(j) = d^*(j)$ for $j \in S$.
  - $\bar{S}$: denotes the set of **temporarily labeled nodes**.
    - That is, $d(j) \geq d^*(j)$ for $j \in T$.

- Dijkstra'salgorithm will determine $d^*(j)$ for each j, in order of increasing distance from the origin node 1.

Dijkstra's Algorithm

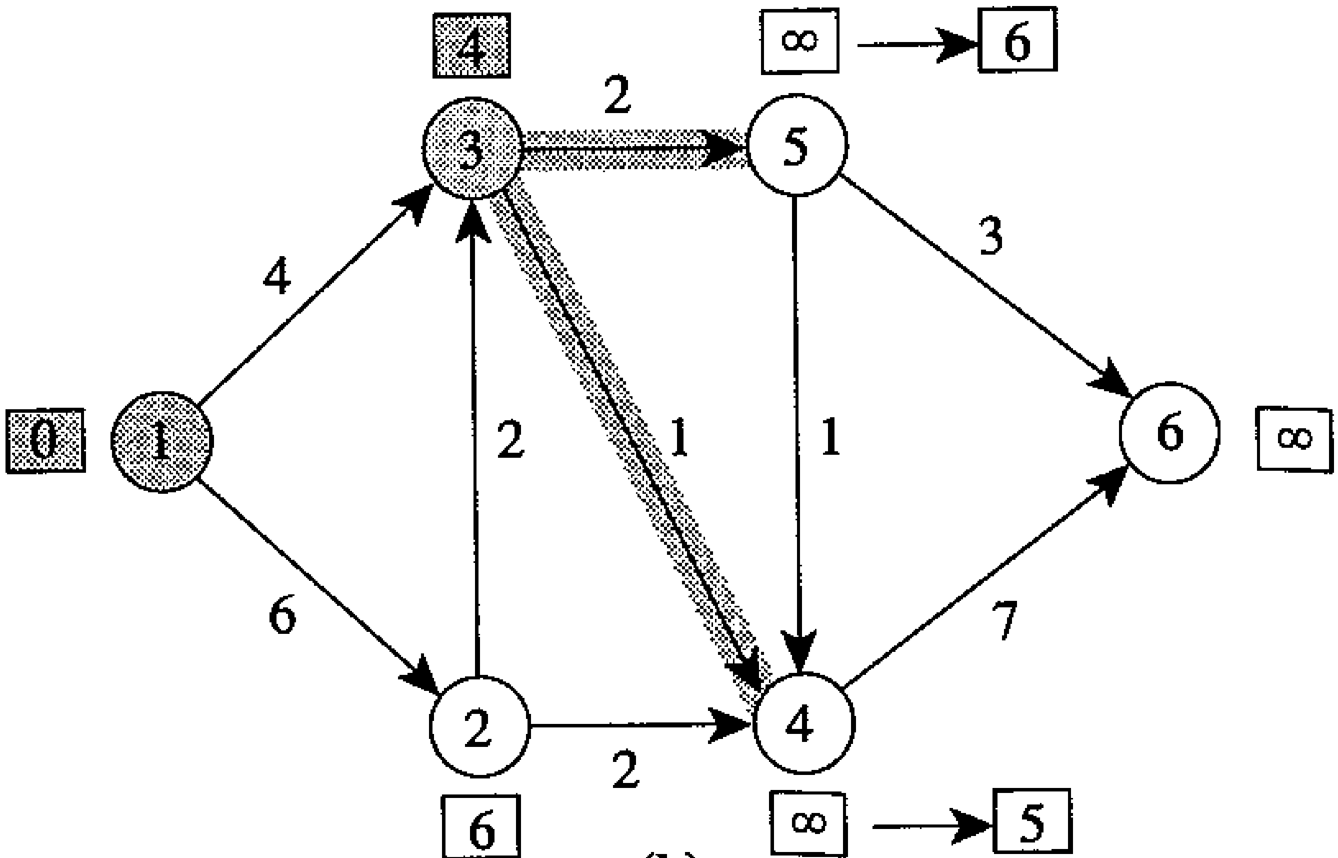
algorithm *Dijkstra*;
begin
    $S := \emptyset$; $\bar{S} := N$;
    $d(i) := \infty$ for each node $i \in N$;
    $d(s) := 0$ and pred$(s) := 0$;
    while $|S| < n$ **do**
    begin
        let $i \in \bar{S}$ be a node for which $d(i) = \min\{d(j) : j \in \bar{S}\}$;
        $S := S \cup \{i\}$;
        $\bar{S} := \bar{S} - \{i\}$;
        for each $(i, j) \in A(i)$ **do**
            if $d(j) > d(i) + c_{ij}$ **then** $d(j) := d(i) + c_{ij}$ and pred$(j) := i$;
    end;
end;

Dijkstra's Algorithm



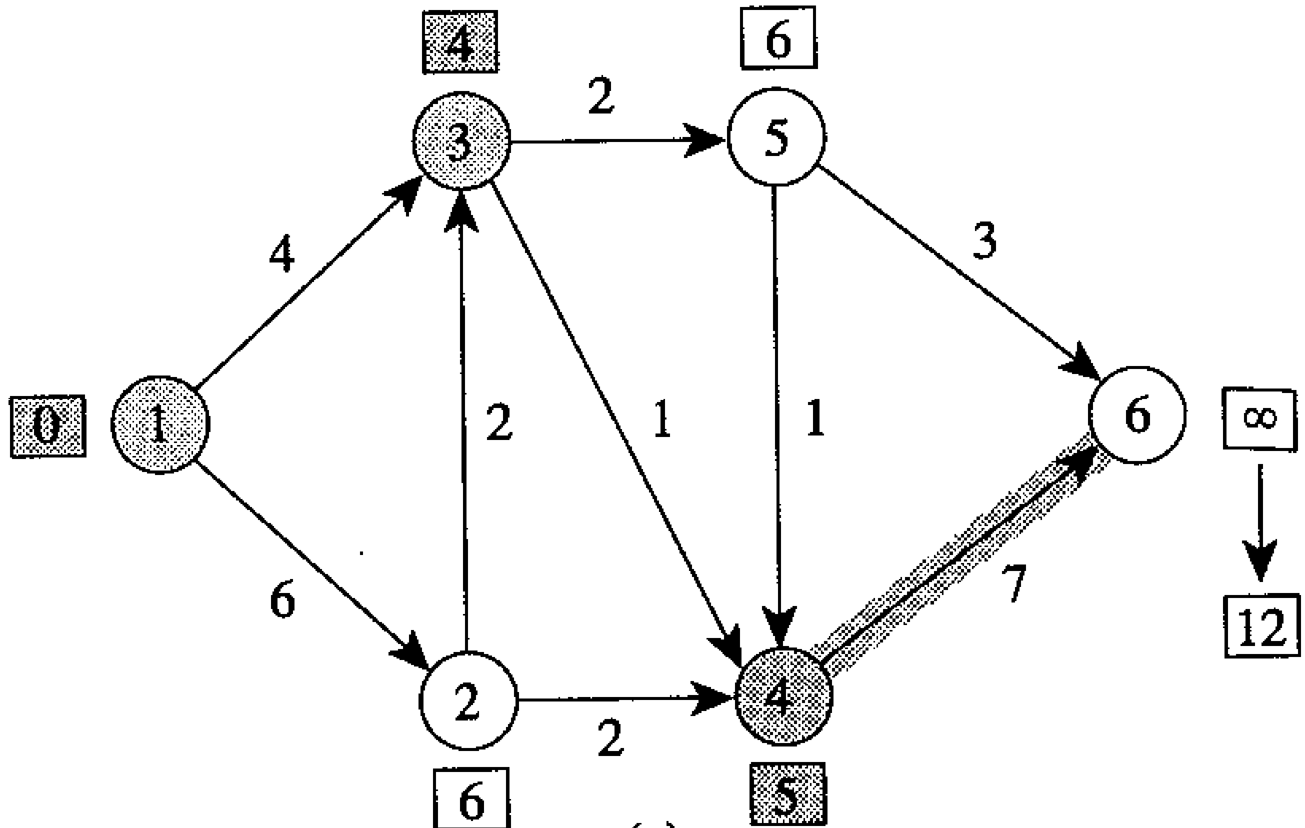
(a)

Dijkstra's Algorithm



(b)

Dijkstra's Algorithm



(c)

Dial's Algorithm

Dial’s Algorithm

- The bottleneck of ***Dijkstra's algorithm*** is node selection.
  - Scanning all temporarily labeled nodes at each iteration to find the one with the minimum distance label, takes high computation time
- ***Dial's algorithm***
  - tries to maintain distances in some sorted fashion and reduces the algorithm's computation time in practice

Dial's Algorithm

- Dial's algorithm
  - stores nodes with finite temporary labels in a sorted fashion.
  - It maintains $nC + 1$ sets, called ***buckets***, numbered 0,1,2, . . . , nC:
  - Bucket k stores all nodes with temporary distance label equal to k.
  - C : represents the largest arc length in the network, and therefore nC is an upper bound on the distance label of any finitely labeled node .
  - We need not store nodes with infinite temporary distance labels in any of the buckets-we can add them to a bucket when they first receive a finite distance label.

Dial's Algorithm

- We represent the content of bucket k by the set *content*(k).
- In the node selection operation:
  - we scan buckets numbered 0, 1, 2, . . . , until we identify the first ***nonempty bucket***.
  - Suppose that bucket k is the first nonempty bucket.
  - Then each node in *content*(k) has the minimum distance label.
  - One by one, we delete these nodes from the bucket, designate them as permanently labeled, and scan their arc lists to update the distance labels of adjacent nodes.
  - Whenever we update the distance label of a node i from d_1 to d_2; we move node i from *content*(d_1) to *content*(d_2).

Dial's Algorithm

- In the next node selection operation, we resume the scanning of buckets numbered $k + 1$, $k + 2$, ... to select the next nonempty bucket.
- Property 4.5 implies that the buckets numbered 0, 1, 2, . . . , k will always be empty in the subsequent iterations and the algorithm need not examine them again.

The End