Parma University, Math. Department

Algorithms and Data Structures SSSP with negative weights

Marco Pellegrini

Summary

- This lecture:
  SSSP with negative weights:
  Bellman-Ford algorithm
  SSSP in directed acyclic graphs.
- Material: CLR 25.3, 25.4

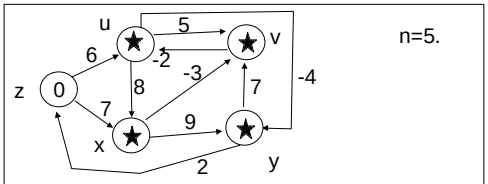
Graphs with negative weights.

- Dijkstra's algorithm solves SSSP for positive weights and is more efficient for sparse graphs.
- If we have negative weights (but no negative cycle) we can solve SSSP problems by using an algorithm by *Bellman and Ford*.
- The idea is to use Relaxation operation on edges not once per edge (like in Dijkstra) but many times.
- Also BF is able to detect the presence of a negative cycle and report this fact.
- We have a function d[v] that for each v in V contains an upper estimate of the minimum distance from s to v. Initially d[s] = 0 and d[v] = infinity for v≠s.

Bellman-Ford

```
Bellman-Ford((V,E),s): boolean
    FOR i :=1 to n DO
      d[i] := infinity
    END FOR
    d[s] := 0
    FOR i := 1 TO n -1 DO
      FOR each edge (u,v) in E DO
        Relax(u,v)
      END FOR
    END FOR
    FOR each edge (u,v) in E DO
      IF d[v] < d[u] + w(u,v) THEN
        RETURN false
      END IF
    END FOR
    RETURN true
```

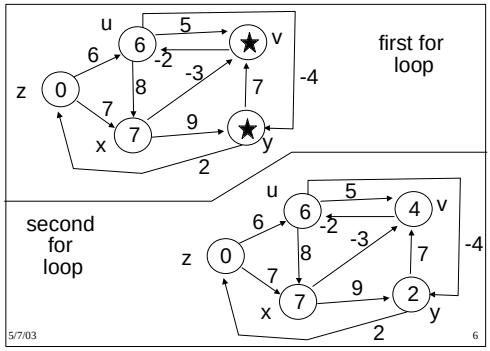

Example (1)



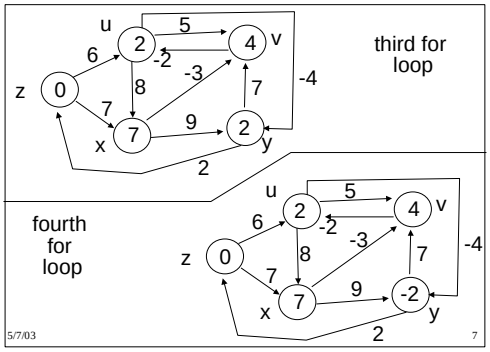
n=5.

Edges are visited in this order:
(u,v), (u,x), (u,y), (v,u), (x,v), (x,y), (y,v), (y,z), (z,u), (z,x).

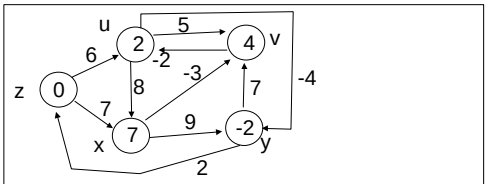
Example (2)



Example (3)



Example (4)



final loop to check for negative cycles

Observations.

- Initialization takes time O(|V|).
  Then we loop |V|-1 times over the edges. The body of the loop is an O(1) operation. So We have O(|V||E|).
- The last loop is O(|E|). The total running time is O(|V||E|), which is better than Floyd-Warshall for spares graphs.
- Exercise: Simulate Bellman-Ford on the graph in the example starting from y.

Proof of correctness.

- Lemma 25.12 If there are no negative cycles and v is reachable from s, then at the end of the main for loop d[v] = D[s,v], the array d holds the length of the shortest path between s and v.
- Proof. By induction on the length of the shortest path from s to v. Let $p=v_0, ...,v_k$ such path. $s=v_0$, $v=v_k$.
- If there are no negative cycles p cannot pass on the same node twice, so $k \leq n-1$.
- For i=0, $d[v_0]=d[s]=0=D[s,s]$. Base case.
- Suppose after the (i-1)-th round of the external for loop $d[v_{i-1}]=D[s,v_{i-1}]$. Then in the i-th loop when we reduce (v_{i-1},v_i) we get $d[v_i]=D[s,v_i]$ by lemma 25.7 which we have seen in lecture 16 on Dijkstra's alg.

Proof of correctness (2).

- If v is not reachable form s then it is easy to see d[v] = infinity = D[s,v].
- What remains to be proved is:
  If there is no negative cycle, BF returns TRUE
  If there a negative cycle reachable from s, then the algorithm returns FALSE.
- From lemma 25.12 at the end of the main loop for any node u and v, and d[u]=D[s,u], d[v]=D[s,v].
- But by lemma 25.3:
  $d[v] = D[s,v] \leq D[s,u] + w(u,v) = d[u] + w(u,v)$.
  So the test in the check loop always fail, we get out and return TRUE.

Negative loops.

- Let $l=(v_0,...,v_k=v_0)$ be a negative loop reachable from s. We show now by contradiction that the algorithm cannot fail all checks in the “check” loop.
- The cycle l is negative: $A= \Sigma_{i=1..k} w(v_{i-1},v_i) <0$.
- If the check in the checkloop is always false we have $\Sigma_{i=1..k} d[v_i] \leq \Sigma_{i=1..k} d[v_{i-1}] + A$.
- But since $v_0=v_k$, the two summations range on the same nodes and they have equal value. Simplifying: $A \geq 0$. Contradiction.
- So if there is a negative cycle the “check” loop will find it.

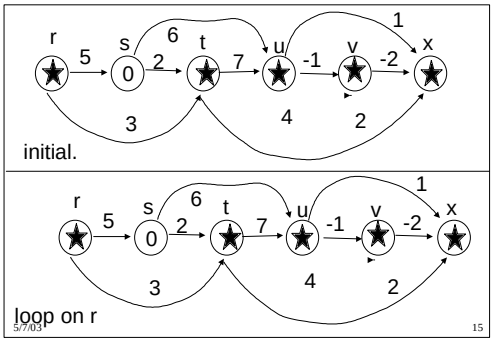
Homework:

- HW5.3: (CLR 25.3-4) Modify BF so that it sets d[v] = infinity to each node v for which there is a negative cycle on some path from s to v.

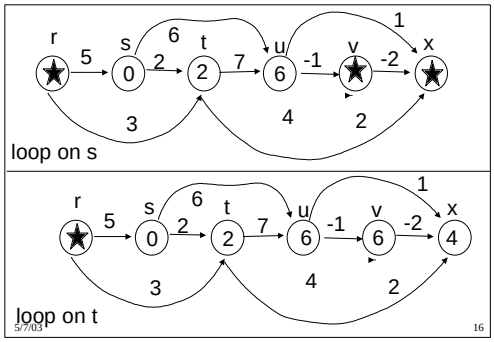
SSSP in Directed Acyclic Graphs.

- If G is a directed acyclic graph, we know that even if we have negative weights there cannot be a negative cycle.
- Can we do better than BF on a DAG?
- yes, the idea is that there is a natural order of the vertices (and edges) of a DAG that is given by topological sorting.
-
```
DAG-Shortest-path((V,G),s)
    Topologically sort V,
    Initialize(d,V,s)
    FOR each vertex u in topological order DO
        FOR each edge (u,v) adjacent to u DO
            Relax(u,v)
        END FOR
    END FOR
```

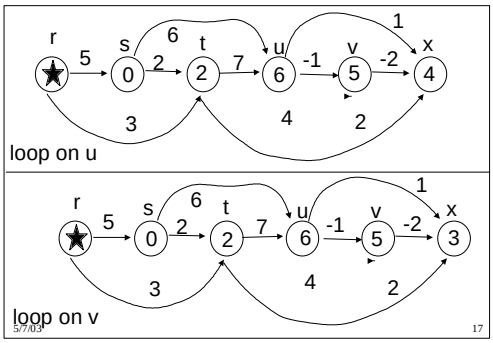

Example (1)



Example (2)



Example (3)



Exercise

- Simulate DAG-shortest-path starting from r.

Observations

- We can do the topological sorting by DFS which takes O(|V|+|E|), then we loop through each edge once. Overall the time is O(|V|+|E|) that is faster than Bellman-Ford.
- The proof of correctness is by induction on the length of the number of nodes on the shortest path between s and a node v: $p=(v_0,..,v_k)$ with $v_0=s$, $v_k=v$.
- We run through the nodes in topological order: the edges of p ar relaxed in the order of the path (we did not have this property in Bellman-Ford).
- Assuming inductively $d[v_{i-1}]=D[s,v_{i-1}]$ when the for loop reached v_{i-1} the relaxation of (v_{i-1},v_i) has the effect of setting $d[v_i]=D[s,v_i]$. Initially $d[v_0]=0=D[s,s]$.

Conclusion

- We have seen Bellman-Ford's algorithm for solving SSSP with negative weights.
- Bellman-Ford is efficient on sparse graphs and runs in time O(|V||E|).
- For Directed acyclic graphs, with negative weight we can solve SSSP in time O(|V|+|E|).