Lecture 14

Graphs II

Dr. Yusuf H. Sahin
Istanbul Technical University

sahinyu@itu.edu.tr

From Aysegul Yayimli's slides

Shortest Path

- Weighted digraph: A directed graph with real valued weights assigned to each edge.
  - G(V,E,w)
- Length of a path in a weighted digraph: Sum of the lengths of the edges on the path.
- Shortest path: A path between two nodes of least length.

Dijkstra's Method

- Let G(V,E) be a weighted digraph all of whose edge weights are positive.
- x and y are vertices of G.

Aim: Find the shortest path from x to y and its length, or show there is none.

- The method uses a search tree technique based on:
  - k^{th} nearest vertex to x is the neighbor of one of the j^{th} nearest vertices to x for some $j < k$.

Dijkstra's Method

- Let:
  - `Near(j)` denote the j^{th} nearest vertex to x
  - `Dist(u)` distance from x to any vertex u
  - `Length(u,v)` edge length from u to any neighbor v.
- Then, the k^{th} nearest vertex to x is v that minimizes:

  `Dist(Near(j)) + Length(Near(j),v)`

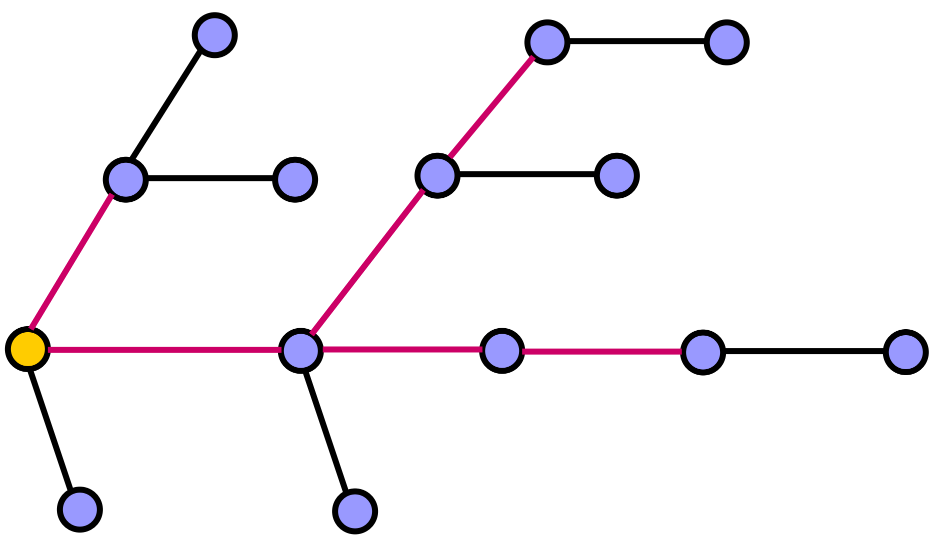
  where the minimum is taken over all $j < k$.
- So, to find the distance to y, we first find the distances to all vertices closer to x than y.

Dijkstra's Method

- Successively more distant vertices from x are found using a search procedure which explores the graph in a tree-like manner.
- This search induces a subgraph of G called a search tree.
- This tree contains a subtree called a shortest path subtree.
- At each phase, a new vertex v lying in the search tree is explored, and the search tree is extended from v to its neighbors.

Dijkstra's Method

- Initially, the search tree fans out from x to its immediate neighbors.
- After k stages, the shortest path subtree of the search tree contains the k nearest vertices to x.
  - The path through this tree from x to any of its vertices is a shortest path.



• **Black Edges**:
lead to vertices of the search tree, but not yet in the shortest path subtree.
• **Pink Edges**:
lead to vertices which are in the shortest path subtree.
• **Any edge not shown**:
  • unexplored
  • Don't to lie on the shortest path

Dijkstra's Algorithm

- Function Dijkstra (G, x, y)
  - Returns the shortest distance from x to y in Dist[y]
  - Returns the shortest path using the Pred field starting at y
  - or fails.
- Dist[0..|V|]: real
  - Contains the current estimated distance to v from x.
- Pred[0..|V|]: 0..|V|
  - Gives the index of the search tree predecessor of v.

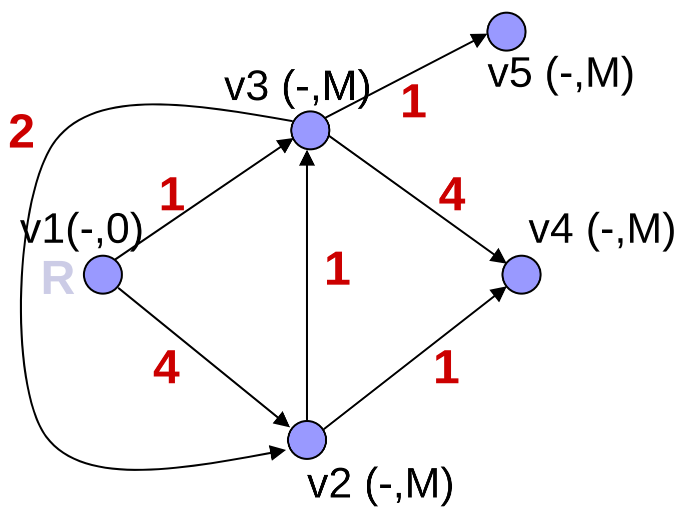
Function Dijkstra

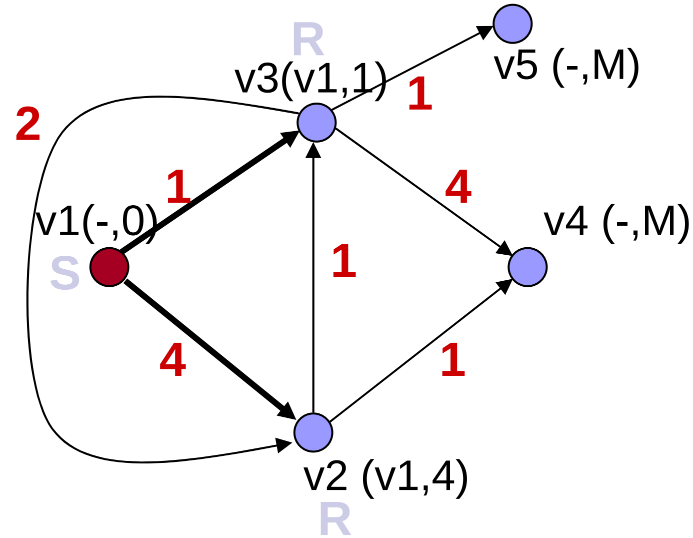
```
Reached = {x}
Pred(w) = 0 for each vertex w in G
Dist(x) = 0
Dist(w) = M, for each w <> x
while getmin(v) and v <> y do
  for each neighbor w of v do
    if w unreached then
      add w to Reached
      Dist(w) = Dist(v) + Length(v,w)
      Pred(w) = v
    else
      if w in Reached and Dist(w) > Dist(v) +
                                    Length(v,w) then
        Dist(w) = Dist(v) + Length(v,w)
        Pred(w) = v
Dijkstra = (v = y)
```

getmin(v):
- returns the vertex v in Reached with the minimum value of Dist(v)
- removes v from Reached
- places v in shortest path tree

Example

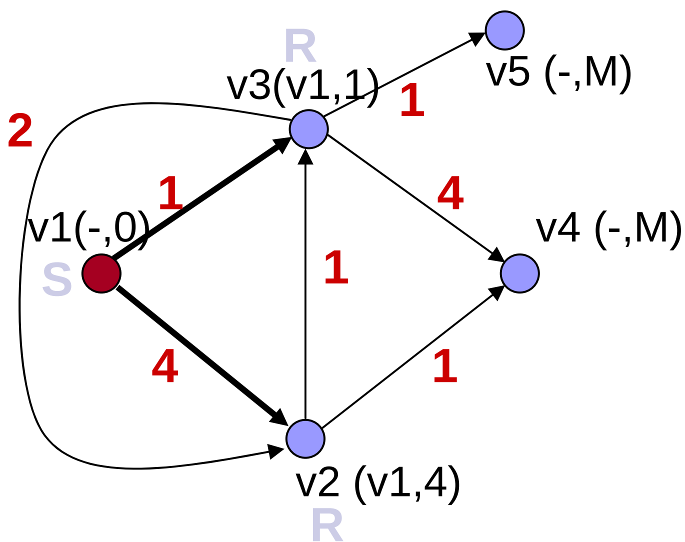
The shortest path from v1 to v4 is sought.



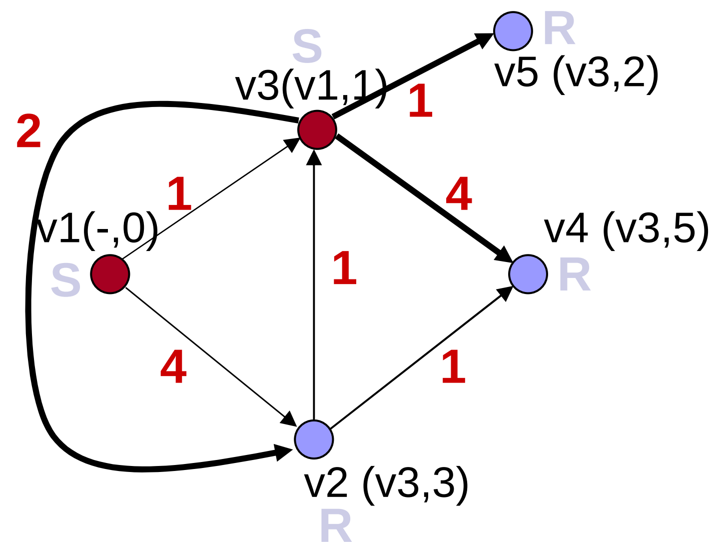


Weighted digraph G

Example

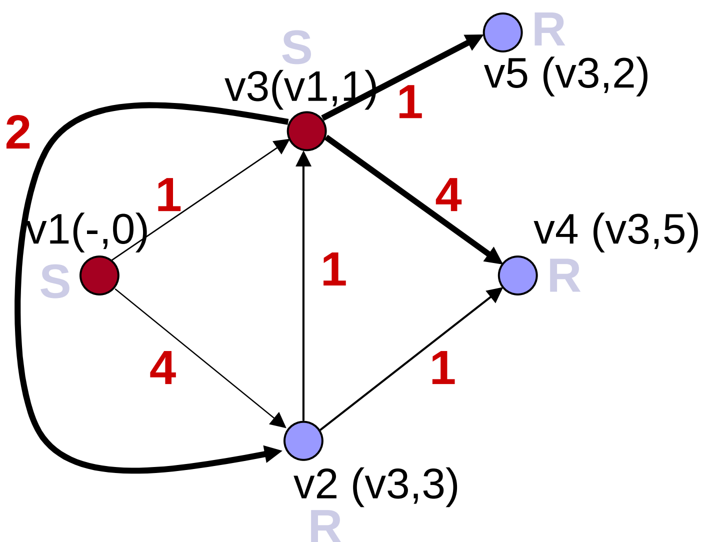


R
v3(v1,1)
v5 (-,M)
2
1
1
4
v1(-,0)
v4 (-,M)
S
1
4
1
v2 (v1,4)
R

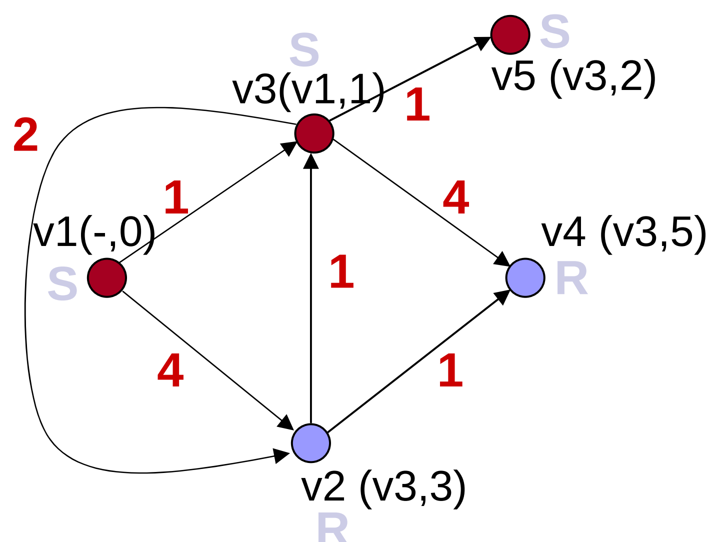


S
R
v3(v1,1)
v5 (v3,2)
2
1
1
4
v1(-,0)
v4 (v3,5)
S
1
R
4
1
v2 (v3,3)
R

Example

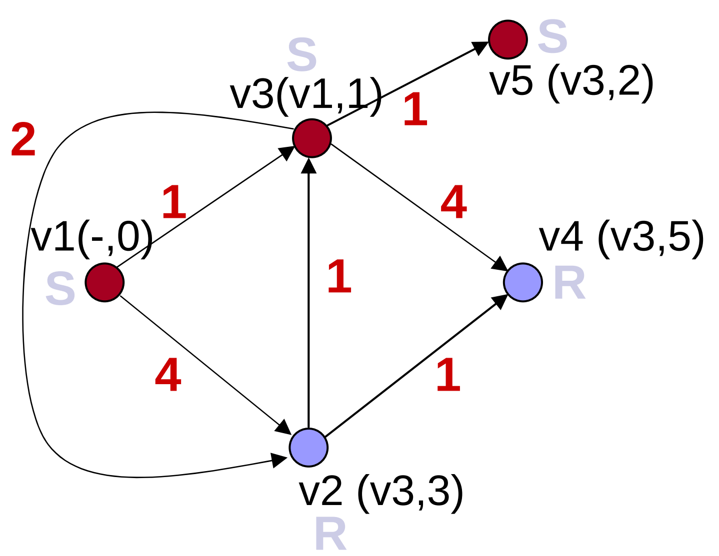


R
S
v5 (v3,2)
v3(v1,1)
1
2
1
4
v1(-,0)
v4 (v3,5)
S
1
R
4
1
v2 (v3,3)
R



S
S
v5 (v3,2)
v3(v1,1)
1
2
1
4
v1(-,0)
v4 (v3,5)
S
1
R
4
1
v2 (v3,3)
R

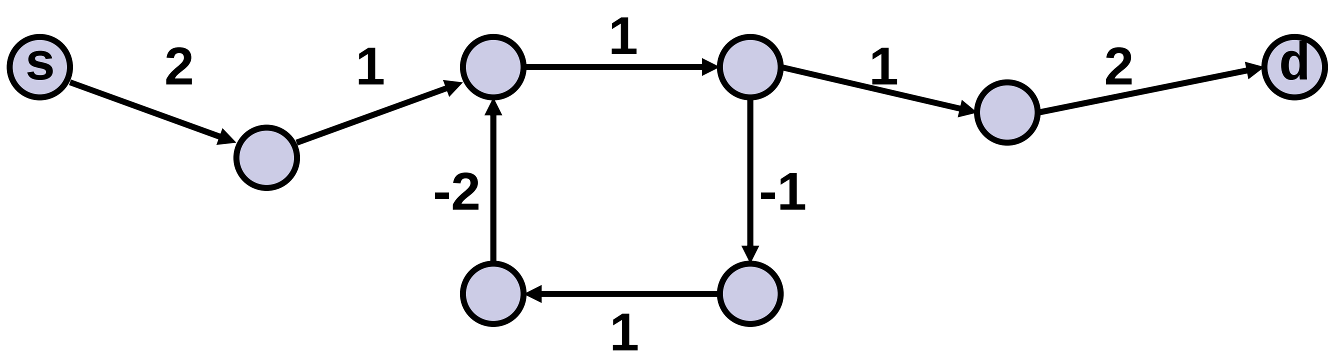
Example



S
v5 (v3,2)
S
v3(v1,1)
1
2
1
4
v1(-,0)
v4 (v3,5)
S
1
R
4
1
v2 (v3,3)
R
S
v5 (v3,2)
S
v3(v1,1)
1
2
1
4
v1(-,0)
v4 (v2,4)
S
1
R
4
1
v2 (v3,3)
S

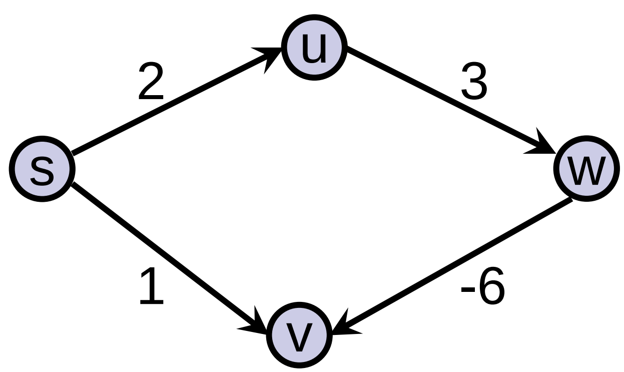
Negative Cycles

- Shortest path problem is considered under the assumption that there are no negative cycle in the graph.
- If there is a negative cycle C:
  - Path P_s from source to C
  - Go around C as many times as you want
  - Path P_d from cycle to destination



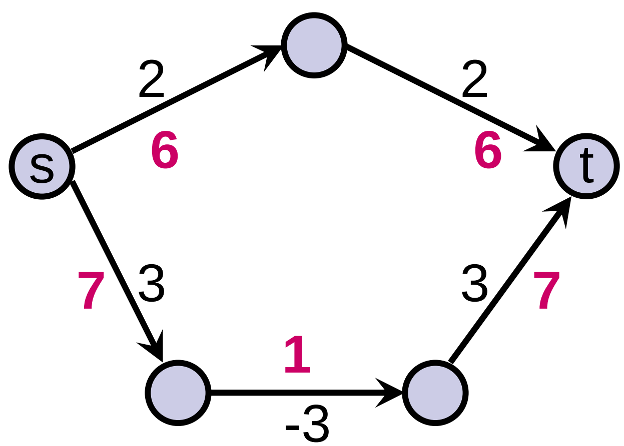
Why Dijkstra don't work with negative cycles

- Start with S = {s}
- Minimum cost path leaving s is (s,v): Add v to S
- Shortest path from s to v is (s,v) assuming there are no negative weighted edges.
- But, this is no longer true:
  - Minimum length path from s to v: s-u-w-v



Can we modify costs?

- A natural idea:
  - Modify costs by adding some large constant M
  - $c_{ik}^{new} = c_{ik} + M$ for each edge
  - M is large enough, all c_{ik}^{new} are positive.
  - Then, use Dijkstra's method.



- Changing costs changes the minimum cost paths.
- We added:
  - 2M to upper path
  - 3M to the lower path