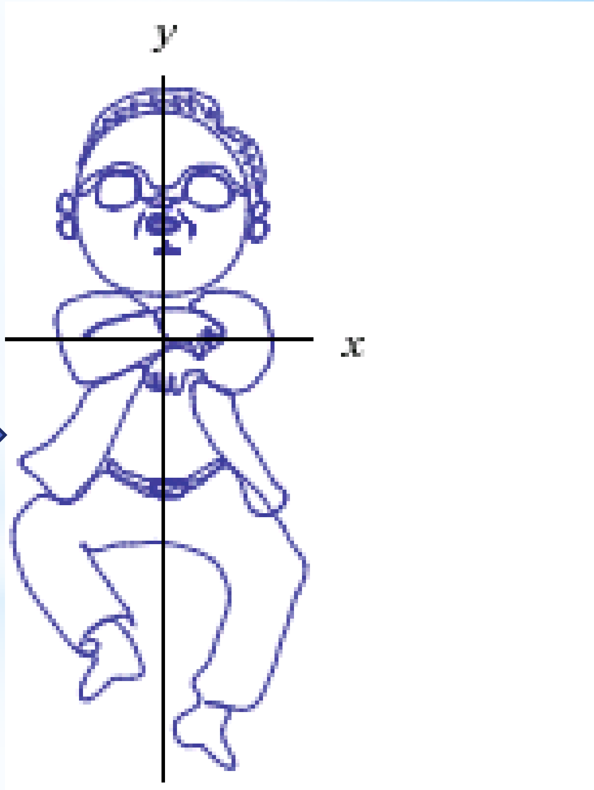
Graph Algorithms



(plotted for t from 0 to 72π)

Property:

person curve

*Overview

* Bellman-Ford: single-source shortest distance
  * O(VE) for graphs with negative edges
  * Detects negative weight cycles
* Floyd-Warshall: All pairs shortest distance
  * O(V^3)

*Bellman-Ford

Weight function w(a,b): weight of the direct path from a to b

Each vertex v has attributes:

d: current minimum distance from start to v

Previous: the vertex in the current shortest path from start to v just before v

```
Relax(u,v,w):
    if v.d>u.d+w(u,v)
        v.d=u.d+w(u,v)
        v.previous=u
```

Bellman Ford

* Bellman-Ford(G,w,s)
  Set all v.d=∞ and s.d=0
  Set all v.previous=null
  For i=1 to |G.V|-1
    For each edge (u,v) in G.E
      Relax(u,v,w)

* Why does it work?

* The shortest path from s to v contains at most |G.V|-1 edges. Consider a shortest path p <s,v1,v2,v3,v4…,vn>. For each iteration of the first for loop we add a vertex to this shortest path. E.g.: v1.previous=S. The shortest path from S to v1 is saved as v1.d. We now add v2 (because it is reachable from v1) and v2.previous=v1. This is never overwritten because v1 is indeed in the shortest path from s to v2 (because otherwise p would not be the shortest path: otherwise replace v1 by vx and we have created a shorter path). By induction it follows that after |G.V|-1 iterations we have considered all shortest paths with |G.V|-1 edges.

*Negative Cycle Check

```
For each edge (u,v) in G.E
        if v.d>u.d+w(u,v)
                return False
Return True
```

Proof:

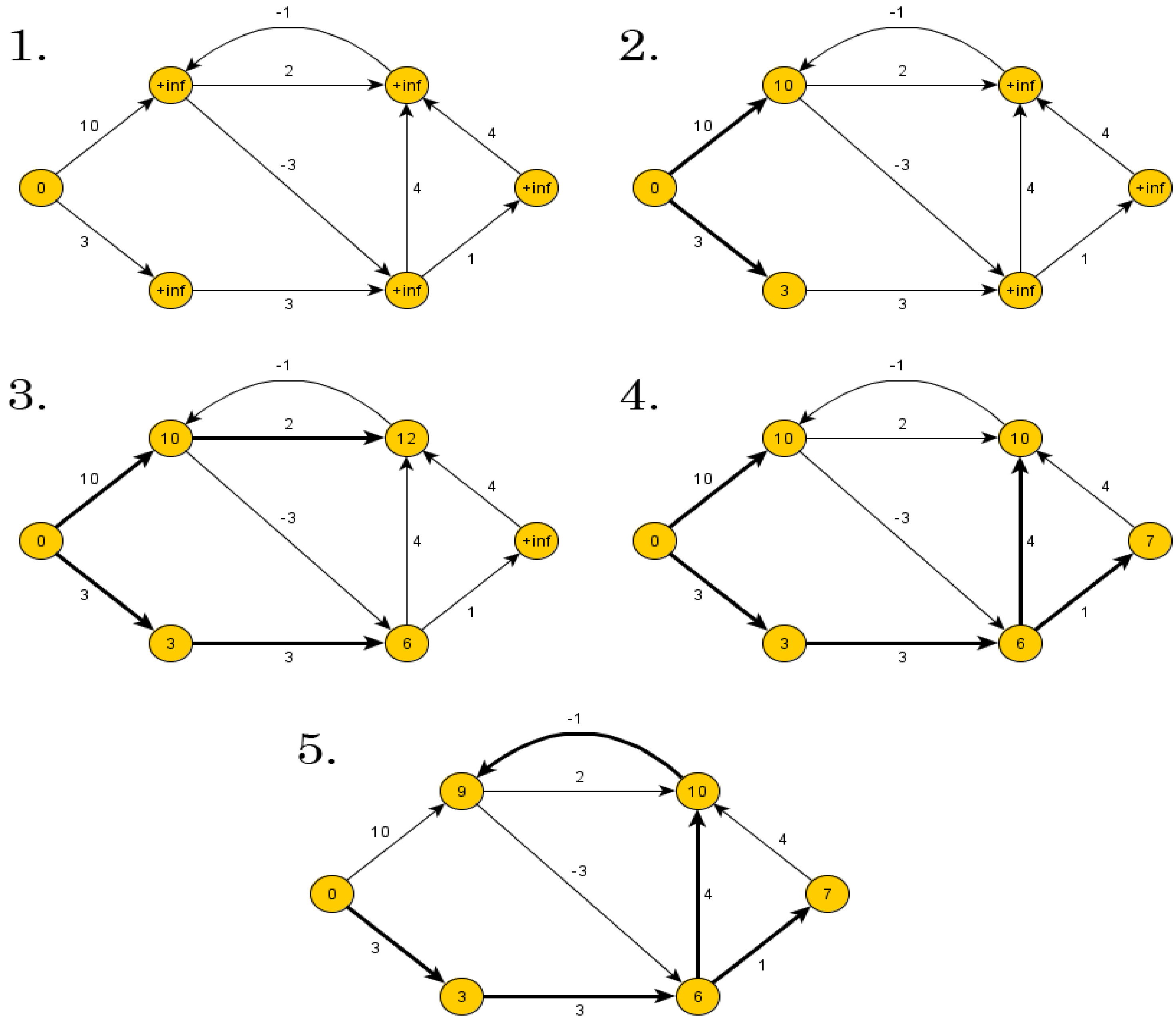
Returns true correctly because: v.d=s(s,d)<=s(s,u)+w(u,v)=u.d+w(u,v)

Suppose <v0,v2,….,vn> is a negative edge cycle accessible from s where v0=vn:

Assume returns true. Then vi.d<=v(i-1).d+w(v(i-1),vi) for 1 to n.

Sum from i=1 to n-> 0<=w(v(i-1),vi), which contradicts:

sum_1_n(w(vi-1,vi))<0


1.
-1
+inf
2
+inf
10
-3
4
4
0
+inf
3
1
+inf
3
+inf
2.
-1
10
2
+inf
10
-3
4
4
0
+inf
3
1
3
3
+inf
3.
-1
10
2
12
10
-3
4
4
0
+inf
3
1
3
3
6
4.
-1
10
2
10
10
-3
4
4
0
7
3
1
3
3
6
5.
-1
9
2
10
10
-3
4
4
0
7
3
1
3
3
6

Floyd-Warshall

- Shortest-Path algorithm has optimum substructure:
- Consider shortest path v1->v2 with intermediate vertex vi. Then v1->…->vi must be the shortest path from v1->vi and vi->…->v2 must be the shortest path from vi->v2. Dynamic programming:
- We don’t know which intermediate vertex to choose. We want all pair shortest-paths.

*Floyd-Warshall

- Define A[a][b][k] to be the length of the shortest path from a to b with possible intermediate vertices 1,2,3,…k.
- Set A[a][b][0] to the weight of the edge connecting a to b.
- Recursive relation: A[a][b][k+1]=
- Min(A[a][k+1][k]+A[K+1][b][k]), A[a][b][k])
- Because we only need a single level of k we can use O(V^2) memory if desirable.

* Code:
* Set A[a][b][0]=w(a,b)
* For k=1 to |V|

```
For a=1 to |V|
    for b=1 to |V|
        A[a][b][k]=min(A[a][b][k-1],A[a][k][k-1]+A[k][b][k-1])
```

O(V^3) time

$$D_0 = \begin{pmatrix} 0 & 5 & \infty & 2 & \infty \\ \infty & 0 & 2 & \infty & \infty \\ 3 & \infty & 0 & \infty & 7 \\ \infty & \infty & 4 & 0 & 1 \\ 1 & 3 & \infty & \infty & 0 \end{pmatrix} \quad D_1 = \begin{pmatrix} 0 & 5 & \infty & 2 & \infty \\ \infty & 0 & 2 & \infty & \infty \\ 3 & 8 & 0 & 5 & 7 \\ \infty & \infty & 4 & 0 & 1 \\ 1 & 3 & \infty & 3 & 0 \end{pmatrix} \quad D_2 = \begin{pmatrix} 0 & 5 & 7 & 2 & \infty \\ \infty & 0 & 2 & \infty & \infty \\ 3 & 8 & 0 & 5 & 7 \\ \infty & \infty & 4 & 0 & 1 \\ 1 & 3 & 5 & 3 & 0 \end{pmatrix}$$

$$D_3 = \begin{pmatrix} 0 & 5 & 7 & 2 & 14 \\ 5 & 0 & 2 & 7 & 9 \\ 3 & 8 & 0 & 5 & 7 \\ 7 & 12 & 4 & 0 & 1 \\ 1 & 3 & 5 & 3 & 0 \end{pmatrix} \quad D_4 = \begin{pmatrix} 0 & 5 & 6 & 2 & 3 \\ 5 & 0 & 2 & 7 & 8 \\ 3 & 8 & 0 & 5 & 6 \\ 7 & 12 & 4 & 0 & 1 \\ 1 & 3 & 5 & 3 & 0 \end{pmatrix} \quad D_5 = \begin{pmatrix} 0 & 5 & 6 & 2 & 3 \\ 5 & 0 & 2 & 7 & 8 \\ 3 & 8 & 0 & 5 & 6 \\ 2 & 4 & 4 & 0 & 1 \\ 1 & 3 & 5 & 3 & 0 \end{pmatrix}$$