UNT
UNIVERSITY OF NORTH TEXAS

CSCE 3110
Data Structures and Algorithms

Graph III

Floyd–Warshall Algorithm

- Main idea:
  - To decrease the distance $d(s, t)$ between two vertices s and t, we can try to add some intermediate vertices in the path $s \rightarrow t$.
    - If adding a new vertex u can decrease the distance $d(s, t)$, i.e.,

      $$d(s, u)+d(u, t)<d(s, t)$$

      we need to update $d(s, t)=d(s, u)+d(u, t)$
    - Otherwise, we do nothing

Example

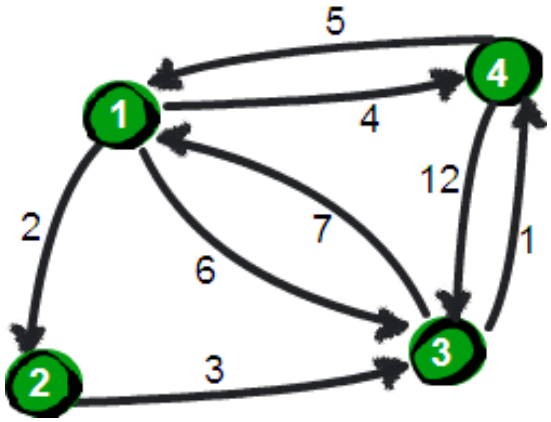
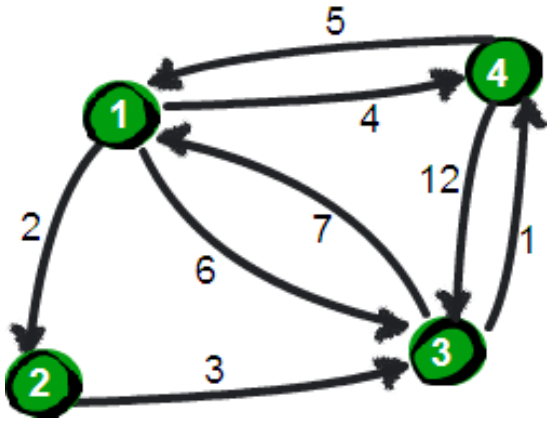


Image credit: https://www.cnblogs.com/wangyuliang/p/9216365.html

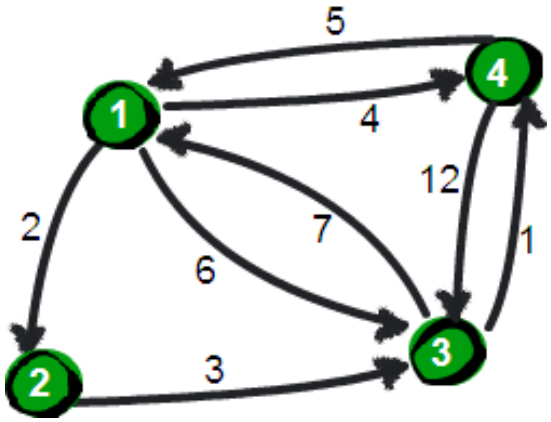
Example



	1	2	3	4
1	0	2	6	4
2	∞	0	3	∞
3	7	∞	0	1
4	5	∞	12	0

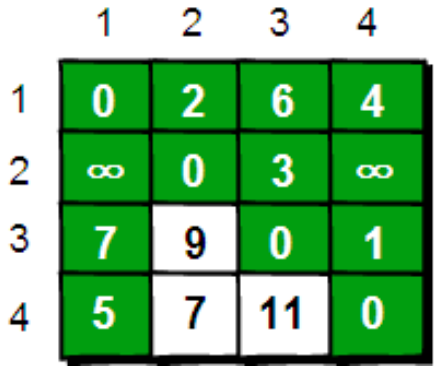
initial $d_0(s, t)$: not adding any vertex

Example



	1	2	3	4
1	0	2	6	4
2	∞	0	3	∞
3	7	∞	0	1
4	5	∞	12	0

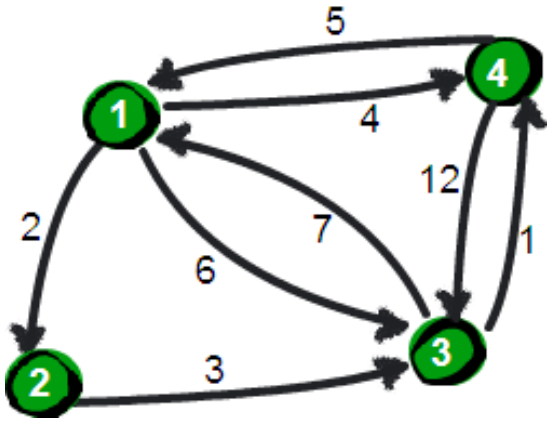
initial $d_0(s, t)$: not adding any vertex



	1	2	3	4
1	0	2	6	4
2	∞	0	3	∞
3	7	9	0	1
4	5	7	11	0

updated $d_1(s, t)$: adding vertex 1 to the path

Example



	1	2	3	4
1	0	2	6	4
2	∞	0	3	∞
3	7	∞	0	1
4	5	∞	12	0

initial $d_0(s, t)$: not adding any vertex

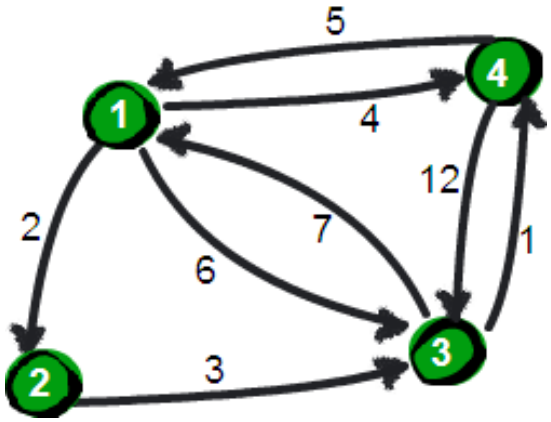
	1	2	3	4
1	0	2	6	4
2	∞	0	3	∞
3	7	9	0	1
4	5	7	11	0

updated $d_1(s, t)$: adding vertex 1 to the path

	1	2	3	4
1	0	2	5	4
2	∞	0	3	∞
3	7	9	0	1
4	5	7	10	0

updated $d_2(s, t)$: adding vertices 1 and 2 to the path

Example



	1	2	3	4
1	0	2	6	4
2	∞	0	3	∞
3	7	∞	0	1
4	5	∞	12	0

initial $d_0(s, t)$: not adding any vertex

	1	2	3	4
1	0	2	6	4
2	∞	0	3	∞
3	7	9	0	1
4	5	7	11	0

updated $d_1(s, t)$: adding vertex 1 to the path

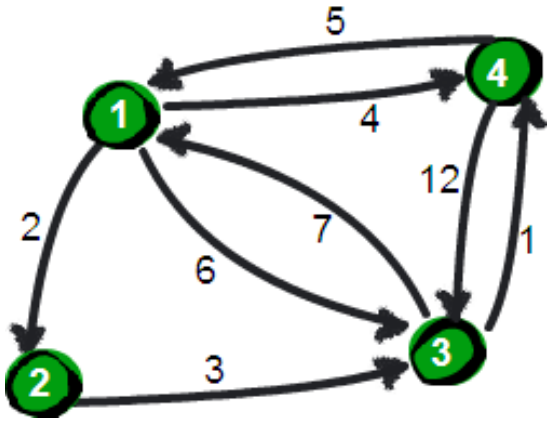
	1	2	3	4
1	0	2	5	4
2	∞	0	3	∞
3	7	9	0	1
4	5	7	10	0

updated $d_2(s, t)$: adding vertices 1 and 2 to the path

	1	2	3	4
1	0	2	5	4
2	10	0	3	4
3	7	9	0	1
4	5	7	10	0

updated $d_3(s, t)$: adding vertices 1, 2 and 3 to the path

Example



	1	2	3	4
1	0	2	6	4
2	∞	0	3	∞
3	7	∞	0	1
4	5	∞	12	0

initial $d_0(s, t)$: not adding any vertex

	1	2	3	4
1	0	2	6	4
2	∞	0	3	∞
3	7	9	0	1
4	5	7	11	0

updated $d_1(s, t)$: adding vertex 1 to the path

	1	2	3	4
1	0	2	5	4
2	∞	0	3	∞
3	7	9	0	1
4	5	7	10	0

updated $d_2(s, t)$: adding vertices 1 and 2 to the path

	1	2	3	4
1	0	2	5	4
2	10	0	3	4
3	7	9	0	1
4	5	7	10	0

updated $d_3(s, t)$: adding vertices 1, 2 and 3 to the path

	1	2	3	4
1	0	2	5	4
2	9	0	3	4
3	6	8	0	1
4	5	7	10	0

updated $d_4(s, t)$: adding vertices 1, 2, 3 and 4 to the path

Algorithm

- 5 lines, incredibly simple:

```
for (k=1; k<=n; k++)
    for (i=1; i<=n; i++)
        for (j=1; j<=n; j++)
            if (d(i, j) > d(i, k) + d(k, j)
                d(i, j) = d(i, k) + d(k, j)
```

Question

- How to obtain the shortest path by modifying the algorithm?

Algorithm

- 6 lines, still incredibly simple:

```
for (k=1; k<=n; k++)
    for (i=1; i<=n; i++)
        for (j=1; j<=n; j++)
            if (d(i, j) > d(i, k) + d(k, j)
                d(i, j) = d(i, k) + d(k, j);
                next(i, j) = next(i, k);
```