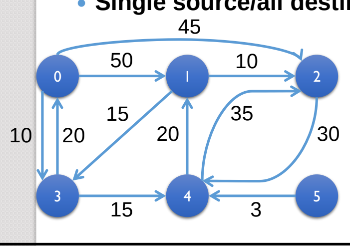
6.4

Shortest Paths and Transitive Closure

6.4

Single Source Shortest Paths

- Given a **digraph** with **nonnegative edge costs**, we want to compute the **shortest path** from a source vertex to all other vertices.
- **Single source/all destinations** problem.



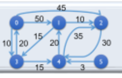
Paths from 0 to 1:
0→1 : 50
0→2→4→1 : 95
…
0→3→4→1 : 45

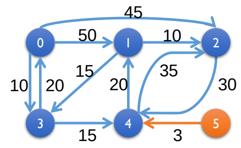
Dijkstra's Algorithm

- Similar to Prim's algorithm
- Use a set *S* store the vertices whose shortest path have been found
- Use an array *dist* store the shortest distances from source v to all visited vertices

- The algorithm
  1. Let *S={v}*, all entries in *dist* = ∞
  2. For each vertex *w* not in *S*, update *dist*
     $dis[w]=min(dist[u]+length(\langle u,w\rangle),dist[w])$
     u is the newly added vertex to *S* adjacent to w
  3. Add to *S* the vertex *x* not in *S* but of the minimum cost in *dist*.
  4. Repeat last two steps until *S* include all vertices.

Running Example

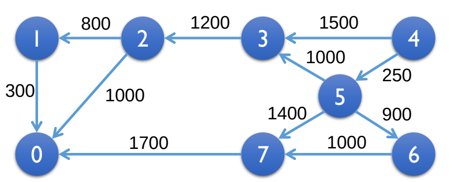




S	0	1	2	3	4	5
{0}	0	50	45	10	∞	∞
{0, 3}	0	50	45	10	25	∞
{0, 3, 4}	0	45	45	10	25	∞
{0, 3, 4, 1}	0	45	45	10	25	∞
{0, 3, 4, 1, 2}	0	45	45	10	25	∞

63

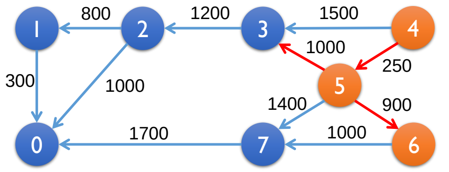
Running Example 2: 1/3



S	0	1	2	3	4	5	6	7
{4}	∞	∞	∞	1500	0	250	∞	∞
{4, 5}	∞	∞	∞	1250	0	250	1150	1650
{4, 5, 6}	∞	∞	∞	1250	0	250	1150	1650

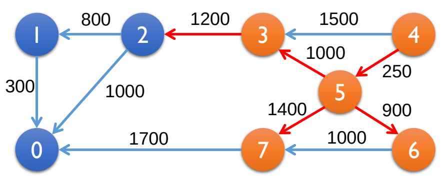
64

Running Example 2: 2/3



S	0	1	2	3	4	5	6	7
{4, 5, 6}	∞	∞	∞	1250	0	250	1150	1650
{4, 5, 6, 3}	∞	∞	2450	1250	0	250	1150	1650
{4, 5, 6, 3, 7}	3350	∞	2450	1250	0	250	1150	1650

Running Example 2: 3/3



S	0	1	2	3	4	5	6	7
{4, 5, 6, 3, 7}	3350	∞	2450	1250	0	250	1150	1650
{4, 5, 6, 3, 7, 2}	3350	3250	2450	1250	0	250	1150	1650
{4, 5, 6, 3, 7, 2, 1}	3350	3250	2450	1250	0	250	1150	1650
{4, 5, 6, 3, 7, 2, 1, 0}	3350	3250	2450	1250	0	250	1150	1650

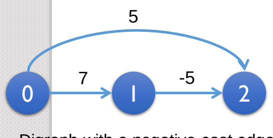
Dijkstra's Algorithm

```
void MatrixWDigraph::ShortestPath(const int n, const int v)
{ // dist[j], 0 ≤ j < n, stores the shortest path from v to j
  // length[i][j] stores length of edge <i, j>
  for(int i=0; i<n; i++){ s[i]=false; dist[i]=length[v][i];}
  s[v] = true;
  dist[v] = 0;
  // find n − 1 paths starting from v
  for(int i=0; i<n−1 ;i++){ -------------------------> O(n)
    // Choose a vertex u, such that dist[u]
    // is minimum and s[u] = false
    int u = Choose(n); ---------------------------> O(n)
    s[u] = true;
    for(int w=0; w<n; w++) -------------------------> O(n)
    if(!s[w] && dist[u] + length[u][w] < dist[w])
       dist[w] = dist[u] + length[u][w];
  } // end of for (i = 0; ...)
}
```

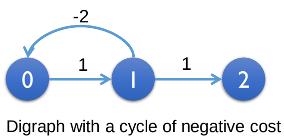
Time complexity: $O(n^2)$

Digraph with Negative Costs

- This algorithm also applies to **digraph with negative cost edges**.
- However, the digraph **MUST NOT** contain cycles of negative length.



Digraph with a negative cost edge



Digraph with a cycle of negative cost

6.4.3

All-Pairs Shortest Paths

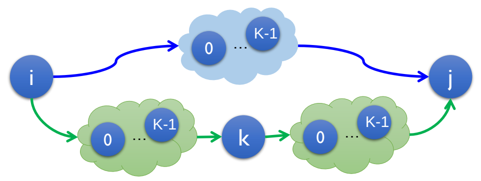
- Apply the single source shortest path to each of n vertices.
- Floyd-Warshall's algorithm
  - $A^{-1}[i][j]$: is just the $length[i][j]$
  - $A^{n-1}[i][j]$: the length of the shortest i-to-j path in G
  - $A^{k}[i][j]$: the length of the shortest path from i to j going through no intermediate vertex of index greater than k.
  - $A^{k}[i][j]$= min{$A^{k-1}[i][j]$, $A^{k-1}[i][k]$+ $A^{k-1}[k][j]$ }, $k \geq 0$

69

Floyd-Warshall's Algorithm

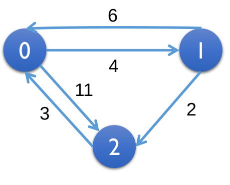
- There are only two possible paths for $A^{k}[i][j]$!
  - The path dose not pass vertex k.
  - The path dose pass vertex k.

$$A^{k}[i][j] = \min\{A^{k-1}[i][j], A^{k-1}[i][k] + A^{k-1}[k][j]\}, k \geq 0$$



70

Example



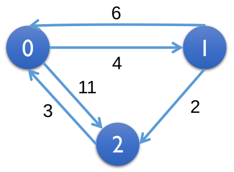
A^{-1}	0	1	2
0	0	4	11
1	6	0	2
2	3	∞	0

$$A^{0}[2][1] = \min(A^{-1}[2][1], A^{-1}[2][0] + A^{-1}[0][1])$$
$$= \min(\infty, 3+4) = 7$$

$$A^{0}[1][2] = \min(A^{-1}[1][2], A^{-1}[1][0] + A^{-1}[0][2])$$
$$= \min(2, 6+11) = 2$$

71

Example



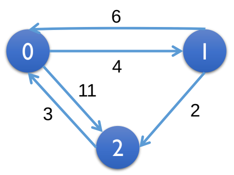
A^0	0	1	2
0	0	4	11
1	6	0	2
2	3	7	0

$$A^1[2][0] = \min(A^0[2][0], A^0[2][1] + A^0[1][0]) = \min(3, 7+6) = 3$$

$$A^1[0][2] = \min(A^0[0][2], A^0[0][1] + A^0[1][2]) = \min(11, 4+2) = 6$$

72

Example



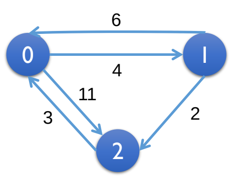
A^1	0	1	2
0	0	4	6
1	6	0	2
2	3	7	0

$$A^2[0][1] = \min(A^1[0][1], A^1[0][2] + A^1[2][1]) = \min(4, 6+3) = 4$$

$$A^2[1][0] = \min(A^1[1][0], A^1[1][2] + A^1[2][0]) = \min(6, 2+3) = 5$$

73

Example



A^2	0	1	2
0	0	4	6
1	5	0	2
2	3	7	0

74

Floyd-Warshall's Algorithm

```
void MatrixWDigraph::AllLengths(const int n)
{// length[n][n] stores edge length between
 // adjacent vertices
 // a[i][j] stores the shortest path from i to j
 for (int i = 0; i<n; i++)-----------------> O(n)
  for (int j = 0; j<n; j++)----------------> O(n)
     a[i][j]= length[i][j];

 // path with top vertex index k
 for (int k= 0; k<n; k++) -----------------> O(n)
  // all other possible vertices
  for (int i= 0; i<n; i++)-----------------> O(n)
   for (int j= 0; j<n; j++)----------------> O(n)
    if((a[i][k]+a[k][j])<a[i][j])
      a[i][j] = a[i][k] + a[k][j];
}
```

Time complexity: $O(n^3)$

75

Transitive Closure

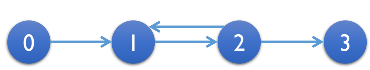
- The **transitive closure matrix A^+**:
  - A^+ is a matrix such that $A^+[i][j] = 1$ if there is **a path of length > 0 from i to j** in the graph; otherwise, $A^+[i][j] = 0$.
- The **reflexive transitive closure matrix A^***:
  - A^* is a matrix such that $A^*[i][j] = 1$ if there is **a path of length ≥ 0 from i to j** in the graph; otherwise, $A^*[i][j] = 0$.
- Use Floyd-Warshall's algorithm!

$A^k[i][j] = A^{k-1}[i][j] \,||\, (A^{k-1}[i][k] \,\&\&\, A^{k-1}[k][j])$

76

6.4.4

Example: Transitive Closure



A+	0	1	2	3
0	0	1	1	1
1	0	1	1	1
2	0	1	1	1
3	0	0	0	0

Transitive closure matrix

A*	0	1	2	3
0	1	1	1	1
1	0	1	1	1
2	0	1	1	1
3	0	0	0	1

Reflexive transitive closure matrix

77