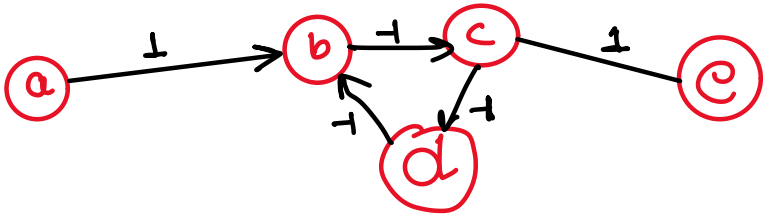
Lecture 14

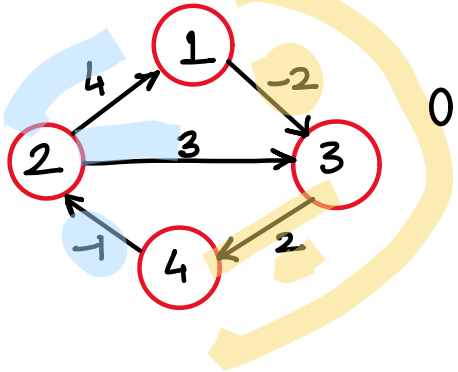
## Shortest Path Algorithm: Floyd–Warshall algorithm

Finding shortest paths in a directed weighted graph G=(V,E,wt) with positive or negative edge weights (but with no negative cycles).




dist(a,c) can be made very small.

Eg:




0

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | -1 | -2 | |
| 2 | | 0 | | |
| 3 | | | 0 | |
| 4 | | -1 | | 0 |

$n^2$-sized

← complete all entries of distance metric

**Shortest-path(i,j,k)** := shortest path from i to j using internal vertices from set [1...k]


Lemma 1: Shortest-path(i,j,n) := distance from i to j in the graph.

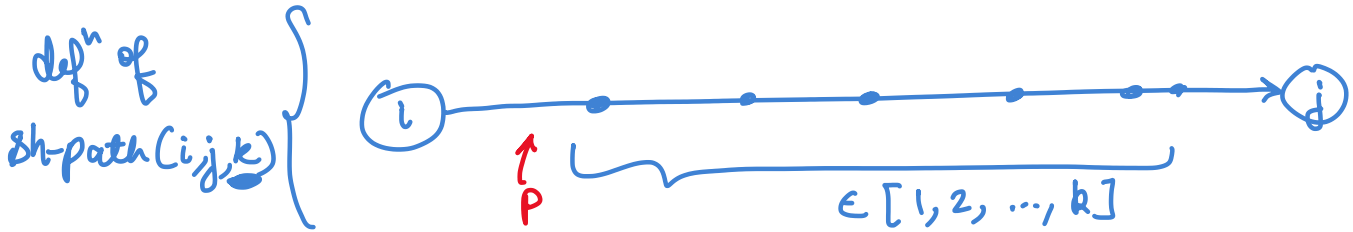
Observation 1: Shortest-path(i,j,0) := wt(i,j)

weight of edge (i,j). If no edge exists, then ∞.

Observation 2: For k>0

Shortest-path(i,j,k) := Min{Shortest-path(i,j,k-1) ,
Shortest-path(i,k,k-1)+Shortest-path(k,j,k-1)}

Proof

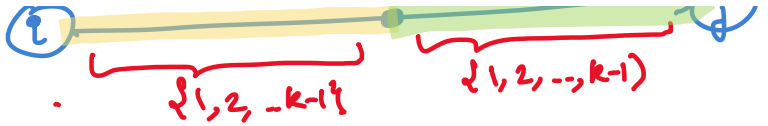



Case 1
k ∉ P

Case 2
k ∈ P

$\Rightarrow$ intermediate vertices $\in [1, k-1]$

$sh\text{-}path(i,j,k) = sh\text{-}path(i,j,k-1)$




$sh\text{-}path(i,j,k) = sh\text{-}path(i,k,k-1) + sh\text{-}path(k,j,k-1)$

Algorithm:

1. Create 2-D array dist of size n × n with all entries initialized to ∞
2. for each edge (u, v) do
   **dist**[u][v] ← wt(u, v) // weight of the edge (u, v)
3. for each vertex v do
   **dist**[v][v] ← 0

$\forall i,j$ dist[i][j] = shortest-path(i, j, 0) — No internal vertices

4. for k=1 to n:
   for j=1 to n: for j=1 to n :
   dist[i][j] = min{**dist**[i][j] , **dist**[i][k] + **dist**[k][j]}

Recurrence Relation

$SP[i,j,k] = \min\{SP[i,j,k-1], SP[i,k,k-1] + SP[k,j,k-1]\}$

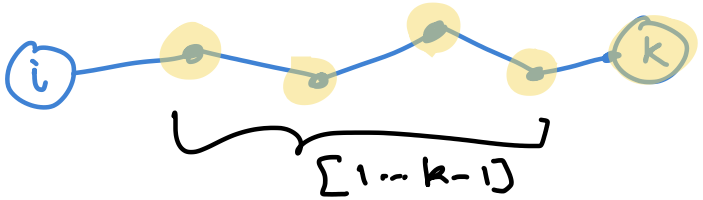
The value stored in dist[i,j] after $k^{th}$ round = shortest-path(i, j, k)

**Correctness:**

HYP(k): After execution of iteration k, we have ∀ i,j ∈ [1,n], dist[i][j]= shortest-path(i,j,k).

Proof: Suppose claim is true for round k-1. We will prove that claim is true for round k.
Choose a pair (i,j). We will consider three cases

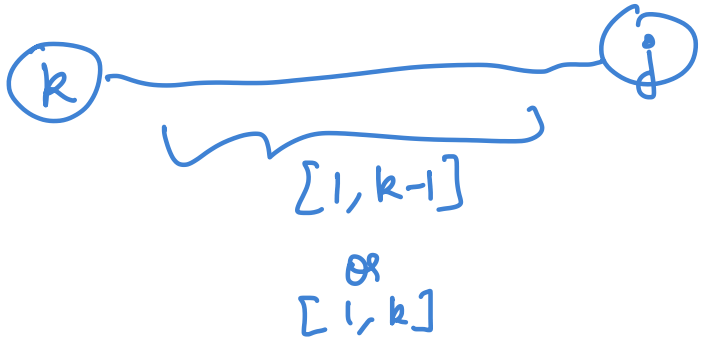
**Case 1:** (i,j) = (i,k):
**Proof:**




- Notice that Shortest-path(i,k,k)=Shortest-path(i,k,k-1).

- Also, our algorithm updates (i,k)$^{th}$ entry as dist[i][k] = min{dist[i][k] , dist[i][k] + dist[k][k]}.
  So, no change is made.

Sh-path(i,k,k-1) = 0

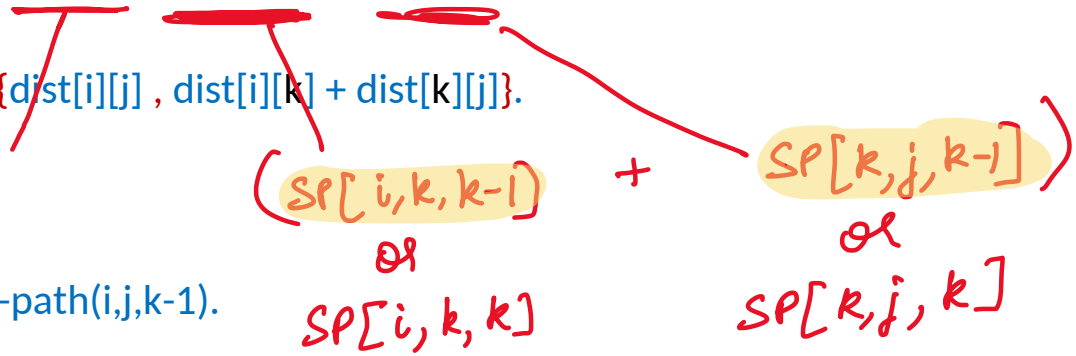



**Case 2:** (i,j) = (k,j):
Similar to Case 1, no change is made.

**Case 3:** Both i,j are not equal to k:

**Proof:**

Our algorithm updates $(i,j)^{th}$ entry as dist[i][j] = min{dist[i][j] , dist[i][k] + dist[k][j]}.




Before this update by induction, **dist**[i][j] = Shortest-path(i,j,k-1).

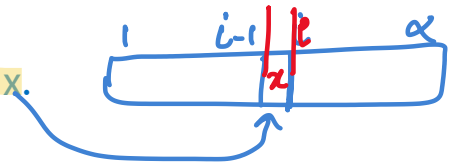
Run Time $= O(n^3)$

Space complexity $= O(n^2)$

## Edit Distance

Given two strings A=(a1,…,am) and B=(b1,…,bn), convert A to B using following operations:

- Replace i-th symbol with x.
- Remove i-th symbol.
- Add to i-th location symbol x.




Instructions

GOLDEN
MOLDEN G -> M at position 1
MODEN Remove L from position 3
MODERN Add R at position 5

Space allowed = $O(m+n)$

- Create 2-D array dist of size (m+1)x(n+1).

1. **For** j=0 to n: dist[0,j] = j
2. **For** i=1 to m:
   - dist[i,0] = i
   - **For** j=1 to n:

```
If ( A[i]=B[j] ):
    dist[i,j] = dist[i-1,j-1]
Else
    dist[i,j] = 1 +min {dist[i-1,j], dist[i-1,j-1], dist[i,j-1]}.
```

3. **Return** dist[m,n].

Space = O(mn)
Time = O(mn)

| | "" | M | O | D | E | R | N |
|---|---|---|---|---|---|---|---|
| "" | 0 | 1 | 3 | 3 | 4 | 5 | 6 |
| G | 1 | 1 | | | | | |
| O | 2 | | | ✓ | ✓ | | |
| L | 3 | | | ✓ | 3 | | |
| D | 4 | | | | | | |
| E | 5 | | | | | | |
| N | 6 | | | | | | |

- Create 2-D array dist of size (2)x(n+1).

1. **For** j=0 to n: dist[1,j] = j

2. **For** i=1 to m:
   - Copy dist[1,*] to dist[0,*].
   - dist[1,0] = i.
   - For j=1 to n:
     If ( A[i]=B[j] ):
       dist[1,j] = dist[0,j-1].
     Else
       dist[1,j] = 1 +min {dist[0,j], dist[0,j-1], dist[1,j-1]}.

3. **Return** dist[0,n].

Space = O(n)
Time = O(mn)

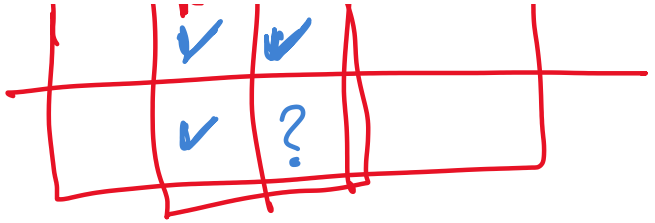
- Create an Instruction stack S initialized to empty.
- Create 2-D array dist of size (2)x(n+1).

Print-Last-Instruction(A,m,B,n)

1. **For** j=0 to n: dist[1,j] = j

2. **For** i=1 to m:
   - Copy dist[1,*] to dist[0,*].
   - dist[1,0] = i.

- For j=1 to n:
  - If ( A[i]=B[j] ):
    - dist[1,j] = dist[0,j-1].
  - Else
    - dist[1,j] = 1 +min {dist[0,j], dist[0,j-1], dist[1,j-1]}.




3. If ( A[m]=B[n] ):
   - str = ""
   - (x,y) = (m-1,n-1)

   Else
   - If dist[1,n]=1 + dist[0,n]
     - str="Remove symbol at index _____", and set (x,y) = (m-1,n)
   - If dist[1,n]=1 + dist[0,n-1]
     - str="Replace symbol at ____ with B[n]", and set (x,y) = (m-1,n-1)
   - If dist[1,n]=1 + dist[1,n-1]
     - str="Add B[n] at position _____", and set (x,y) = (m,n-1)

4. **Return** <Print-Last-Instruction(A,x,B,y) , str>

Time = $O(m \cdot n \cdot (m+n))$

Space = $O(n)$