Dynamic Programming algorithms for all-pairs Shortest Paths
“Shortest Path”

• Given graph G=(V,E) with positive weights W(u,v) on the edges (u, v), and given two vertices a and b.
• Find the “shortest path” from a to b (where the length of the path is the sum of the edge weights on the path). Perhaps we should call this the minimum weight path!
Greedy algorithm

- Start at a, and greedily construct a path that goes to the next closest vertex from a, until you reach b.
- Dijkstra’s Algorithm: $O(n + m \lg n)$
- Problem 1: it doesn’t work correctly if negative weights are presented.
- Problem 2: To compute the shortest paths between all pairs, we have to call Dijkstra’s algorithm $n$ times.
Dynamic Programming

- Subproblem Property: The problem can be recursively defined by the subproblem of the same kind.
- Trade space for time: A table is used to store the solutions of the subproblems (the meaning of “programming” before the age of computers).
Designing a DP solution

- How are the subproblems defined?
- Where are the solutions stored?
- How are the base values computed?
- How do we compute each entry from other entries in the table?
- What is the order in which we fill in the table?
Two DP algorithms for All-pairs shortest paths

• Both are correct. Both produce correct values for all-pairs shortest paths.
• The difference is the subproblem formulation, and hence in the running time.
• The reason both algorithms are given is to teach you how to do DP algorithms!
• But, be prepared to provide one or both of these algorithms, and to be able to apply it to an input (on some exam, for example).
Dynamic Programming

First attempt: let \( \{1,2,\ldots,n\} \) denote the set of vertices.

Subproblem formulation:
- \( M[i,j,k] = \text{min length of any path from } i \text{ to } j \text{ that uses at most } k \text{ edges.} \)

All paths have at most \( n-1 \) edges, so \( 1 \leq k \leq n-1 \). Minimum paths from \( i \) to \( j \) are found in \( M[i,j,n-1] \).
DP approach: Two Questions

• $M[i,j,k] = \text{min length of any path from } i \text{ to } j \text{ that uses at most } k \text{ edges.}$

• How to set the base case ($k=1$)?
• How to set $M[i,j,k]$ from other entries?
• How to set the base case (k=1)?
  – Easy: \( M[x,y,1] = \)
    • 0 if \( x = y \)
    • \( w(x,y) \) if \( x \) and \( y \) are different, and \( (x,y) \) is an edge, and
    • infinity otherwise
  – If using adjacency matrix, \( M[x,y,1] = W[x,y] \).
• How to set $M[i,j,k]$ from other entries, for $k>1$?
• Consider a minimum weight path from $i$ to $j$ that has at most $k$ edges.
  – Case 1: The minimum weight path has at most $k-1$ edges.
    • $M[i,j,k] = M[i,j,k-1]$
  – Case 2: The minimum weight path has exactly $k$ edges.
    • $M[i,j,k] = \min\{M[i,x,k-1] + w(x,j) : x \in V\}$

• Combining the two cases:
$M[i,j,k] = \min\{\min\{M[i,x,k-1] + w(x,j) : x \in V\}, \, M[i,j,k-1]\}$
Finishing the design

• Where is the answer stored?
• How are the base values computed?
• How do we compute each entry from other entries?
• What is the order in which we fill in the matrix?
• Running time?
Running time analysis

For \( k = 1 \) to \( n-1 \)
   for \( j = 1 \) to \( n \)
     for \( i = 1 \) to \( n \)
       \[
       M[i,j,k] = \min\{\min\{M[i,x,k-1] + w(x,j) : x \in V\}, M[i,j,k-1]\}
       \]

• How many entries do we need to compute? \( O(n^3) \)
  \[ 1 \leq i \leq n; 1 \leq j \leq n; 1 \leq k \leq n-1 \]
• How much time does it take to compute each entry? \( O(n) \)
Next DP approach

• Try a new subproblem formulation!
• $Q[i,j,k] = \text{minimum weight of any path from } i \text{ to } j \text{ that uses internal vertices drawn from } \{1,2,\ldots,k\}.$
Designing a DP solution

• How are the subproblems defined?
• Where is the answer stored?
• How are the base values computed?
• How do we compute each entry from other entries?
• What is the order in which we fill in the matrix?
• $Q[i,j,k] =$ minimum weight of any path from $i$ to $j$ that uses internal vertices (other than $i$ and $j$) drawn from \{1,2,\ldots,k\}.

• Base cases: $Q[i,j,0] = W[i,j]$ for all $i,j$

• Minimum paths from $i$ to $j$ are found in $Q[i,j,n]$

• Once again, $O(n^3)$ entries in the matrix
Solving subproblems

• $Q[i,j,k] = \text{minimum weight of any path from } i \text{ to } j \text{ that uses internal vertices drawn from } \{1,2,\ldots,k\}$.

• The minimum cost such path either includes vertex $k$ or does not include vertex $k$. 
Solving subproblems

- $Q[i,j,k] =$ minimum weight of any path from $i$ to $j$ that uses internal vertices drawn from $\{1,2,\ldots,k\}$.
- If the minimum cost path $P$ includes vertex $k$, then you can divide $P$ into the path $P_1$ from $i$ to $k$, and $P_2$ from $k$ to $j$.
- What is the weight of $P_1$?
- What is the weight of $P_2$?
Solving subproblems

- $Q[i,j,k] =$ minimum weight of any path from $i$ to $j$ that uses internal vertices drawn from $\{1,2,\ldots,k\}$.
- $P$ is a minimum cost path from $i$ to $j$ that uses vertex $k$, and has all internal vertices from $\{1,2,\ldots,k\}$.
- Path $P_1$ from $i$ to $k$, and $P_2$ from $k$ to $j$.
- The weight of $P_1$ is $Q[i,k,k-1]$ (why??).
- The weight of $P_2$ is $Q[k,j,k-1]$ (why??).
- Thus the weight of $P$ is $Q[i,k,k-1] + Q[k,j,k-1]$. 
New DP algorithm

for $j = 1$ to $n$
  for $i = 1$ to $n$
    $Q[i,j,0] = W[i,j]$
  for $k = 1$ to $n$
    for $j = 1$ to $n$
      for $i = 1$ to $n$
        $Q[i,j,k] = \min\{Q[i,j,k-1],\quad Q[i,k,k-1] + Q[k,j,k-1]\}$
Running time analysis

• Each entry only takes $O(1)$ time to compute
• There are $O(n^3)$ entries
• Hence, $O(n^3)$ time.
Reusing the space

// Use R[i,j] for Q[i,j,0], Q[i,j,1], ..., Q[i,j,n].
for j = 1 to n  for i = 1 to n  R[i,j] = W[i,j];
for k= 1 to n
    for j = 1 to n
        for i = 1 to n
            R[i,j] = min{R[i,j], R[i,k] + R[k,j]}
How to check negative cycles

// Use R[i,j] for Q[i,j,0], Q[i,j,1], ..., Q[i,j,n].
for j = 1 to n  for i = 1 to n R[i,j] = W[i,j];
for k= 1 to n
  for j = 1 to n
    for i = 1 to n
      R[i,j] = min{R[i,j], R[i,k] + R[k,j]};
for i = 1 to n
  if (R[i,i] < 0) print(“There is a negative cycle”);
How to check negative cycles