

SHORTEST AND LONGEST PATH ALGORITHMS

YOU CAN OFTEN MODEL

a minimization problem: as a shortest path

a maximization problem: as a longest path

IN A SUITABLE DIRECTED GRAPH

WHEN

the solution consists of several sequential decisions,
with each decision influenced by one or more previous decisions.

A SEQUENTIAL DECISION PROBLEM

Problem:

- Given an array $\text{nums}[1..N]$, find a longest increasing subsequence (LIS), i.e., find $\text{nums}[i_1] < \text{nums}[i_2] < \dots < \text{nums}[i_k]$ and $i_1 < i_2 < \dots < i_k$, with k as large as possible.
- Here, the choice of i_j affects the choice of i_{j+1} in two ways.
- As N increases, both the number of increasing subsequences and their lengths tend to increase, and finding an LIS becomes more difficult.

Example. Let $\text{LIS}(i) =$ a longest increasing subsequence in $\text{nums}[1..i]$ and containing $\text{nums}[i]$.

$\text{nums}[i]:$	4	5	2	9	7	6	8	1	3
$\text{LIS}[i]:$	$\langle 4 \rangle$	$\langle 4, 5 \rangle$	$\langle 2 \rangle$	$\langle 4, 5, 9 \rangle$	$\langle 4, 5, 7 \rangle$	$\langle 4, 5, 6 \rangle$	$\langle 4, 5, 6, 8 \rangle$	$\langle 1 \rangle$	$\langle 1, 3 \rangle$
							$\langle 4, 5, 7, 8 \rangle$		$\langle 2, 3 \rangle$

A final solution: $\text{LIS} = \langle 4, 5, 6, 8 \rangle$.

Question:

- ? How many subsequences are there for $\text{nums}[1..N]$? What is the maximum number of increasing subsequences?
- ? Give an equation for computing $\text{LIS}(i)$ from $\text{LIS}(j)$, $j < i$, and that for the final LIS. Use these to design an algorithm (pseudocode) for efficient computation of LIS.
- ? Is this algorithm a "method of extension" or a D.P.? Explain.
- ? Let $\text{LIS}'(i) =$ an LIS for $\text{nums}[1..i]$. Can we use the property $\text{LIS}'(i+1) \supseteq \text{LIS}'(i)$, meaning that each $\text{LIS}'(i+1)$ contains some $\text{LIS}'(i)$, for a more efficient or simpler algorithm for LIS? Explain.

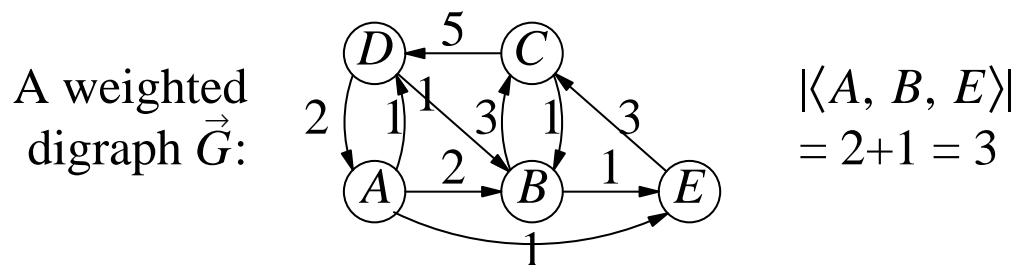
WEIGHTED DIGRAPH AND SHORTEST PATHS

Weighted Digraph:

Each link (x, y) has a *length* or *cost* $c(x, y)$, which may or may not be positive; also, we may have $c(x, y) \neq c(y, x)$.

Path Length or Cost:

For a path $\pi = \langle x_1, x_2, \dots, x_n \rangle$, its length is $|\pi| = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{n-1}, x_n)$.



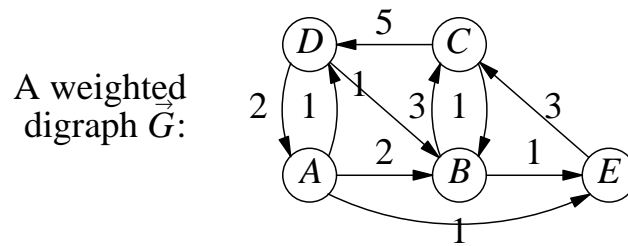
Shortest Path:

- $\pi_m(x, y)$ = an xy -path which has the smallest length among all xy -paths; if there is no xy -path, then we let $|\pi_m(x, y)| = \infty$.
- If an xy -path contains a node which is on a cycle of length < 0 , then $\pi_m(x, y)$ is not well-defined (becomes $-\infty$).

There is no shortest AD -path in \vec{G} above if we add the link (C, E) with $c(C, E) = -4$; it gives a negative cycle at nodes on some AD -paths.

- Henceforth, assume that all links (x, y) have $x \neq y$.
- If \vec{G} is acyclic, there are only a finite number of xy -paths for any x and y , and hence we always have a shortest xy -path if there is at least one xy -path.

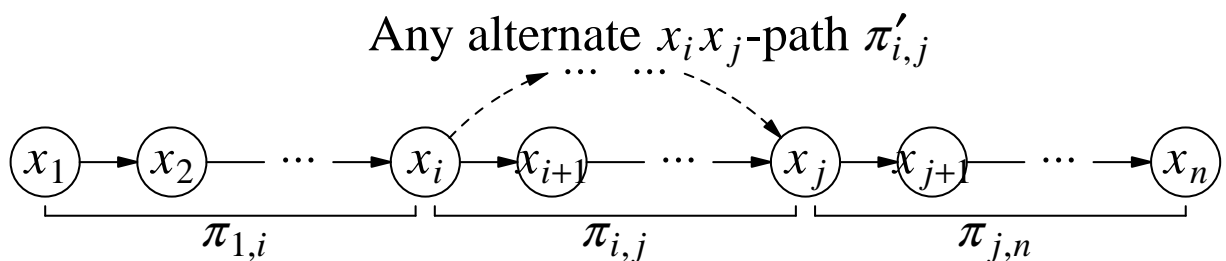
AN EXAMPLE OF SHORTEST PATHS



	$y = A$	$y = B$	$y = C$	$y = D$	$y = E$
$x = A$	$\langle A, A \rangle$ 0	$\langle A, B \rangle$ 2	$\langle A, E, C \rangle$ $4=1+3$	$\langle A, D \rangle$ 1	$\langle A, E \rangle$ 1
$x = B$	$\langle B, C, D, A \rangle$ $10=3+5+2$	$\langle B, B \rangle$ 0	$\langle B, C \rangle$ 3	$\langle B, C, D \rangle$ $8=3+5$	$\langle B, E \rangle$ 1
$x = C$	$\langle C, D, A \rangle$ $7=5+2$	$\langle C, B \rangle$ 1	$\langle C, C \rangle$ 0	$\langle C, D \rangle$ 5	$\langle C, B, E \rangle$ $2=1+1$

Bellman's Optimality Principle:

- Each subpath of a shortest path is itself a shortest path. That is, if $\pi = \langle x_1, x_2, \dots, x_n \rangle$ is a shortest path, then each subpath $\pi_{i,j} = \langle x_i, x_{i+1}, \dots, x_j \rangle$ is a shortest $x_i x_j$ -path for $1 \leq i < j \leq n$.



$$\begin{aligned}
 |\pi_{1,i}| + |\pi_{i,j}| + |\pi_{j,n}| = |\pi| &\leq |\text{the alternate } x_1 x_n\text{-path using } \pi'_{ij}| \\
 &= |\pi_{1,i}| + |\pi'_{i,j}| + |\pi_{j,n}| \\
 \text{i.e., } |\pi_{i,j}| &\leq |\pi'_{i,j}|.
 \end{aligned}$$

Conclusion: Any method for finding a shortest xy -path is likely to find a shortest path between many other node-pairs.

Question:

- ? What does the following shortest path $\pi_m(x_3, x_9)$ in some weighted digraph \vec{G} say about some other shortest paths and their lengths?



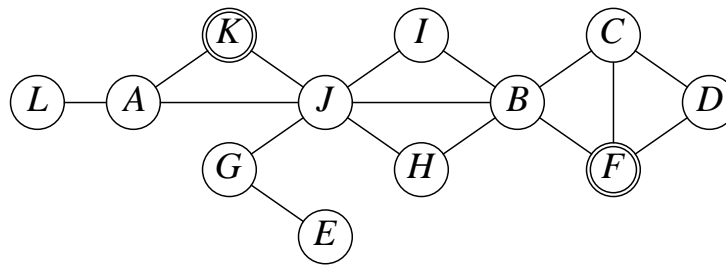
- ? Assume that $c(x, y) > 0$ for each link (x, y) and let $d(x, y) = |\pi_m(x, y)|$, with $d(x, x) = 0$ for all x . Which of the following are true?
 - $d(x, y) \geq 0$ for all x and y , and $= 0$ if and only if $x = y$.
 - $d(x, y) = d(y, x)$ for all x, y .
 - $d(x, z) \leq d(x, y) + d(y, z)$ for all x, y , and z .

If $c(x, y) = 0$ for one or more links (x, y) , $x \neq y$, then which of the properties (i)-(iii) might cease to hold?

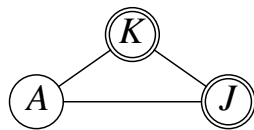
- ? Under what condition can we delete a link (x_p, x_q) in \vec{G} without affecting any $|\pi_m(x_i, x_j)|$, $1 \leq i, j \leq N$? Verify your solution using the digraph \vec{G} on page 3.4. Can the deletions be performed in any order?
- ? Assume that $\vec{G} = (V, \vec{E})$ contains no extraneous link (see the previous problem) and that each $c(x_i, x_j) > 0$. If you are given all $|\pi_m(x_i, x_j)|$, $1 \leq i \neq j \leq N$, how will you determine the links in \vec{G} and the weights? Verify your solution using the digraph \vec{G} on page 3.4. Does your solution work if one or more $c(x_i, x_j) \leq 0$ but there is no negative cost cycle?

ROLE OF BI-COMPONENTS IN FINDING A SHORTEST-PATH

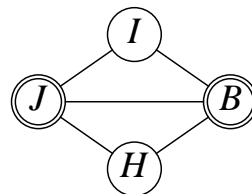
- Suppose following is the *undirected* graph \vec{G}_u when we remove the direction of each link of a digraph \vec{G} ; even if both (x, y) and $(y, x) \in \vec{G}$, we get only one edge (x, y) in \vec{G}_u .



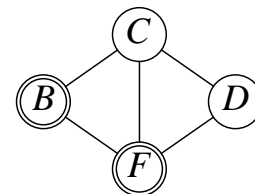
- Any KF -path in \vec{G} , and hence any shortest KF -path, must go through each of the cut-vertices J and B (in that order) in \vec{G}_u which separate K and F .
- This reduces the shortest KF -path problem to three shortest-path problems in smaller digraphs:



- (i) Find a shortest KJ -path in the subdigraph of \vec{G} on $\{A, J, K\}$.



- (ii) Find a shortest JB -path in the subdigraph of \vec{G} on $\{B, H, I, J\}$.

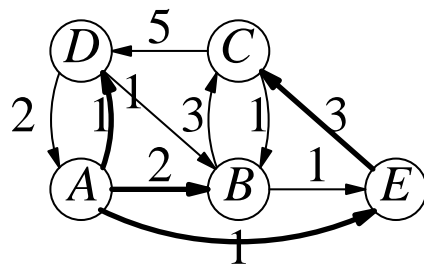


- (iii) Find a shortest BF -path in the subdigraph of \vec{G} on $\{B, C, D, F\}$.

TREE OF SHORTEST PATHS FROM A NODE

Tree $T(x)$:

- For each node y reachable from x , there is a shortest-path $\pi_m(x, y)$ such that these paths together form a rooted tree at x .
- $T(x)$ may not be unique since there may be several shortest xy -paths $\pi_m(x, y)$.



The bold links show the tree $T(A)$.

A Possible Definition of $T(x)$:

- $\text{Children}(x) = \{y: \langle x, y \rangle \text{ is a shortest } xy\text{-path}\}$.
- If we label the children as y_1, y_2, \dots, y_k in some order, then we define the descendants of y_i in $T(x)$ as follows:

$$D(y_1) = \{y: \text{there is a } \pi_m(x, y) \text{ using the link } (x, y_1) \text{ and } y \notin \text{Children}(x)\}$$

$$D(y_2) = \{y: \text{there is a } \pi_m(x, y) \text{ using the link } (x, y_2), y \notin D(y_1) \cup \text{Children}(x)\}$$

$$D(y_3) = \{y: \text{there is a } \pi_m(x, y) \text{ using the link } (x, y_3), y \notin D(y_1) \cup D(y_2) \cup \text{Children}(x)\}, \text{ and so on.}$$

Conclusion: The existence of a tree $T(x)$ suggests that we might have an efficient algorithm for computing shortest-paths.

NUMBER OF ACYCLIC PATHS IN A COMPLETE DIGRAPH

NumPaths_i(1, N): Number of acyclic paths $\pi(1, N)$ with i intermediate nodes.

i	#(paths)	The paths
0	1	$\langle 1, N \rangle$
1	$N - 2$	$\langle 1, j_1, N \rangle, 2 \leq j_1 < N$
2	$(N - 2)(N - 3)$	$\langle 1, j_1, j_2, N \rangle, 2 \leq j_1 \neq j_2 < N$
...
$(N - 3)$	$(N - 2)(N - 3) \cdots 2$	$\langle 1, j_1, j_2, \dots, j_{N-3}, N \rangle, 2 \leq j_k < N$ all j_k distinct
$(N - 2)$	$(N - 2)(N - 3) \cdots 1$	$\langle 1, j_1, j_2, \dots, j_{N-2}, N \rangle$

Total Number of Acyclic Paths NumPaths(1, N):

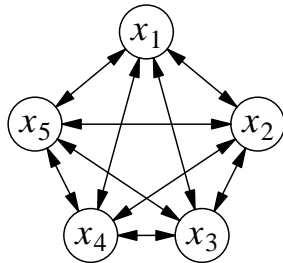
$$\begin{aligned} & \frac{(N-2)!}{(N-2)!} + \frac{(N-2)!}{(N-3)!} + \frac{(N-2)!}{(N-4)!} + \cdots + \frac{(N-2)!}{1!} + \frac{(N-2)!}{0!} \\ & = (N-2)! \left[1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{(N-2)!} \right] \approx e \cdot (N-2)! \end{aligned}$$

- This shows the search space for a shortest $x_i x_j$ -path problem is indeed very large.

Question:

- ? The estimate $e \cdot (N-2)!$ has error < 1 for $N \geq 3$ – why?

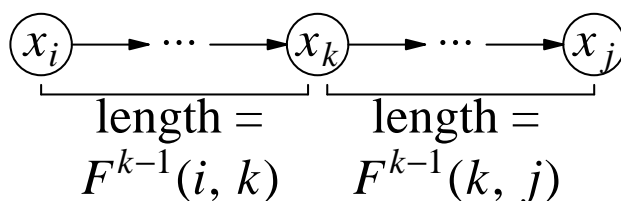
FLOYD'S METHOD FOR SHORTEST-PATHS FOR ALL NODE-PAIRS



- If $(x_i, x_j) \notin \vec{G}$, then let $c(x_i, x_j) = \infty$ (i.e., a large number L , say, $L = 1 + \sum |c(x_p, x_q)|$ (summed over all links in \vec{G}))
- Assume momentarily that each $c(x_i, x_j) \geq 0$ and each $c(x_i, x_i) = 0$.

Floyd's Equations:

- Let $F^k(i, j)$ = The shortest length of an $x_i x_j$ -path which uses zero or more intermediate node from $\{x_1, x_2, \dots, x_k\}$.
- For all $1 \leq i, j \leq N$ and $k \geq 1$,
 - (1) $F^0(i, j) = c(x_i, x_j)$ if $i \neq j$ and $F^0(i, i) = 0$ for each i .
 - (2) $F^k(i, j) = \min \begin{cases} F^{k-1}(i, j) & \text{(if } x_k \text{ is not used)} \\ F^{k-1}(i, k) + F^{k-1}(k, j) & \text{(if } x_k \text{ is used)} \end{cases}$
 - (3) $F^N(i, j) = |\pi_m(x_i, x_j)|$, where $N = \#(\text{nodes})$



The case when x_k is used for $F^k(i, j)$.

Remarks:

- The equations (1)-(3) hold even if one or more $c(x_i, x_j) < 0$ as long as there is no cycle whose total cost is < 0 . A negative cost cycle is detected if $F^k(i, i) < 0$ for some k and i .
- $F^k(i, j) \leq F^{k-1}(i, j)$, i.e., $F^k(i, j)$ gradually decreases to the final value $|\pi_m(x_i, x_j)|$ as k increases from 0 to N .

FLOYD'S SHORTEST PATH ALGORITHM FOR ALL NODE-PAIRS

Observation:

- For $k \geq 1$, $F^k(i, k) = F^{k-1}(i, k)$ and $F^k(k, j) = F^{k-1}(k, j)$ if no negative cycle is detected at the iteration for $(k - 1)$.

$$F^k(i, k) = \min \begin{cases} F^{k-1}(i, k), \\ F^{k-1}(i, k) + F^{k-1}(k, k) \end{cases} \\ = F^{k-1}(i, k)$$

Similarly, $F^k(k, j) = F^{k-1}(k, j)$.

Algorithm FLOYD:

Input: The link-costs $c(x_i, x_j)$ of a digraph on N nodes; $c(x_i, x_i) = 0$ for each $1 \leq i \leq N$.

Output: The costs $F[i, j]$ of an optimal $x_i x_j$ -path for all $1 \leq i, j \leq N$ if there is no negative cycle.

1. [Initialize.] For $(1 \leq i, j \leq N)$, let $F[i, j] = c(x_i, x_j)$.
2. For $(k = 1, 2, \dots, N)$ do the following:
 - For $(1 \leq i, j \leq N$ and $k \neq i, j)$, let $F[i, j] = \min\{F[i, j], F[i, k] + F[k, j]\}$.
 - If (some $F[i, i] < 0$) stop.

Complexity: $= \Theta(N^3)$:

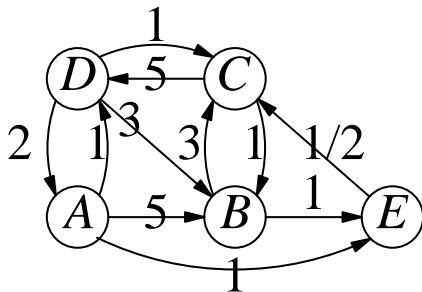
- Step (1) takes $\Theta(N^2)$ time.
- Each iteration for k in step (2) takes $\Theta(N^2)$ time.

KEEPING TRACK OF SHORTEST PATHS

Path $\pi_{i,j}^k$: An $x_i x_j$ -path corresponding to $F^k(i, j)$, i.e., has length $F^k(i, j)$ and uses only the nodes $\{x_1, x_2, \dots, x_k\}$ as possible intermediate nodes.

- Let $Next^k(i, j) =$ the node next to x_i on $\pi_{i,j}^k$.
- Compute $Next^k(i, j)$ along with $F^k(i, j)$ as follows.
 - (1) $Next^0(i, j) = j$
 - (2) $Next^k(i, j) = \begin{cases} Next^{k-1}(i, j), & \text{if } F^k(i, j) = F^{k-1}(i, j) \\ Next^{k-1}(i, k), & \text{if } F^k(i, j) < F^{k-1}(i, j) \end{cases}$
- The final $x_i x_j$ -path is given by $\langle x_i, x_{i_1}, x_{i_2}, \dots, x_j \rangle$, where $i_1 = Next^N(i, j)$, $i_2 = Next^N(i_1, j)$, and so on.

Example. Consider a slightly different digraph shown below.



$$F^3(A, B) = 5, \quad Next^3(A, B) = B$$

$$F^3(D, B) = 2, \quad Next^3(D, B) = C$$

$$F^4(A, B) = 3, \quad \pi_{A,B}^4 = \langle A, D, C, B \rangle, \quad Next^4(A, B) = D.$$

$$F^4(D, B) = 2, \quad \pi_{D,B}^4 = \langle D, C, B \rangle, \quad Next^4(D, B) = C.$$

$$F^4(C, B) = 1, \quad \pi_{C,B}^4 = \langle C, B \rangle, \quad Next^4(C, B) = B.$$

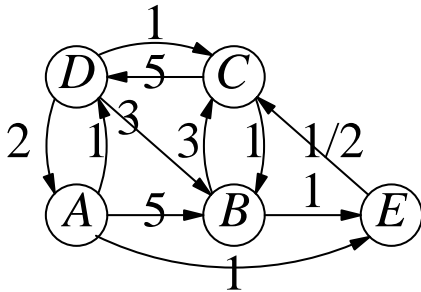
Question:

- ? Show the successive values of $Next^5(\cdot, \cdot)$ in relation to $F^5(A, B)$ and $\pi_{A,B}^5$.

ALTERNATE METHOD FOR KEEPING TRACK OF SHORTEST PATHS

- Let $Best^k(i, j) = \min \{p: F^p(i, j) = F^k(i, j)\}$.
- Compute $Best^k(i, j)$ along with $F^k(i, j)$ as follows:
 - (1) $Best^0(i, j) = 0$ (initialization).
 - (2) $Best^k(i, j) = k$ if $F^k(i, j) < F^{k-1}(i, j)$ for $k \geq 1$; otherwise, $Best^k(i, j) = Best^{k-1}(i, j)$.

Example. Consider the digraph shown below.



$$F^3(A, B) = 5, \quad Best^3(A, B) = 0$$

$$F^3(D, B) = 2, \quad Best^3(D, B) = 3$$

$$F^4(A, B) = 3, \quad \pi_{A,B}^4 = \langle A, D, C, B \rangle, \quad Best^4(A, B) = 4.$$

$$F^3(A, D) = 1, \quad \pi_{A,D}^3 = \langle A, D \rangle, \quad Best^3(A, D) = 0.$$

$$F^3(D, B) = 2, \quad \pi_{D,B}^3 = \langle D, C, B \rangle, \quad Best^3(D, B) = 3.$$

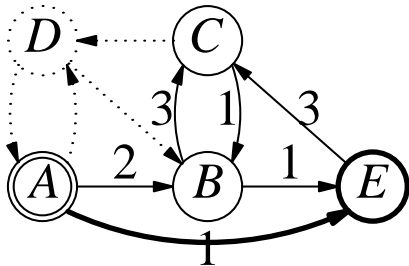
$$F^2(D, C) = 1, \quad \pi_{D,C}^2 = \langle D, C \rangle, \quad Best^2(D, C) = 0.$$

$$F^2(C, B) = 1, \quad \pi_{C,B}^2 = \langle C, B \rangle, \quad Best^2(C, B) = 0.$$

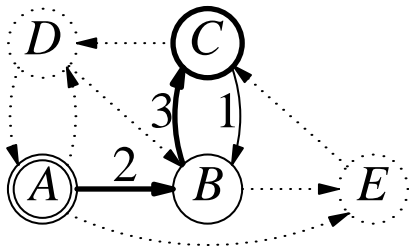
Question:

- ? Show the successive values of $Best^k(\cdot, \cdot)$ in relation to $F^5(A, B)$ and $\pi_{A,B}^5$.
- ? Give a pseudocode for constructing a shortest $x_i x_j$ -path $\pi_m(x_i, x_j)$ from $Best^N(i, j)$'s.

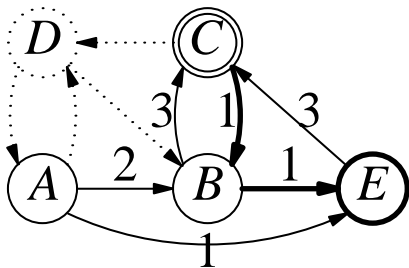
THE NEW PATH $\langle A, B, C, B, E \rangle$ EXAMINED IN COMPUTING $F^3(A, E)$



The part of \vec{G} used for computing $F^3(A, E) = 1$, with nodes A and E specially marked; $\pi_{A,E}^2 = \langle A, E \rangle$.

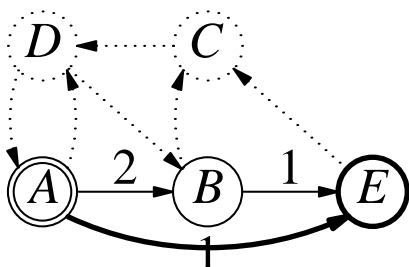


The part of \vec{G} used for computing $F^2(A, C) = 5$, with nodes A and C specially marked; $\pi_{A,C}^2 = \langle A, B, C \rangle$.



The part of \vec{G} used for computing $F^2(C, E) = 2$, with nodes C and E specially marked; $\pi_{C,E}^2 = \langle C, B, E \rangle$.

$$F^3(A, E) = 1 = \min \begin{cases} F^2(A, E) = 1, \\ F^2(A, C) + F^2(C, E) = 5 + 2 = 7 \end{cases}$$



The part of \vec{G} used for computing $F^2(A, E) = 1$, with nodes A and E specially marked; $\pi_{A,E}^2 = \langle A, E \rangle$.

Question: Do we examine the path $\langle C, D, A, E \rangle$ in \vec{G} shown on previous page – explain your answer.

MOST ACYCLIC PATHS ARE NOT EXAMINED IN FLOYD'S ALGORITHM

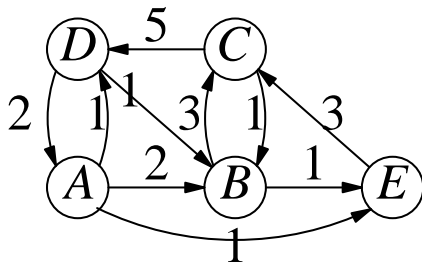
One Additional $x_i x_j$ -path is Examined per Iteration:

- In computing $F^{k-1}(i, k) + F^{k-1}(k, j)$ we implicitly consider the path $\pi_{i,k}^{k-1} \cdot \pi_{k,j}^{k-1}$ for $k \geq 1$ and $k \neq i, j$.

Total Number of $x_i x_j$ -Paths Considered:

- At most $N + 1$, which may include some cyclic paths.
- Most of $\Theta((N - 2)!)$ acyclic $x_i x_j$ -paths are not examined.

Total Number of Paths Considered: $\Theta(N^3)$.

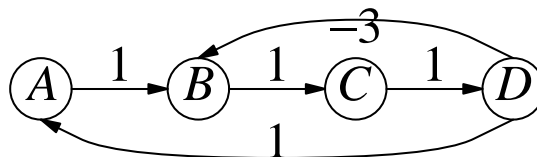


- All links not shown have cost 20.
- $x_1 = A, x_2 = B, \dots, x_5 = E$.

Path $\langle A, B, C, E \rangle$ Not Examined:

- If this path were examined in computing $F^k(A, E)$, i.e., $F^k(1, 5)$, then $k = 3$ (why?). However, the new path examined in computing $F^3(A, E)$ is $\pi_{A,C}^2 \cdot \pi_{C,E}^2 = \langle A, B, C, B, E \rangle$.

Question: What is the smallest k such that $F^k(i, i) < 0$ for some i for the digraph below? What is that i ?



FLOYD'S METHOD: A DYNAMIC-PROGRAMMING SOLUTION

Notion of States:

- The states are (i, j, k) , $1 \leq i, j \leq N$ and $0 \leq k \leq N$.
- Initial states: $(i, j, 0)$:
- Final states: (i, j, N) ; the value computed for each final state constitutes part of the solution.

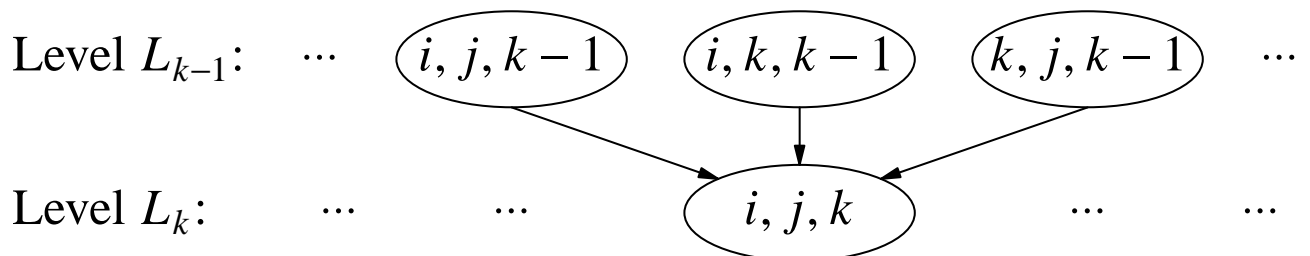
Meaning (Value) Associated with a State:

- $V(i, j, k) = F^k(i, j)$ = the shortest length of an $x_i x_j$ -path which uses only $\{x_1, x_2, \dots, x_k\}$ as possible intermediate nodes.

Computation of $V(i, j, k)$:

- The dependence among the states is determined by the meaning of the states and not by their associated values, which may change when the input-data changes.
- The value at a state in level k may depend on several states in levels $(k - 1)$ or lower:

$$F^k(i, j) = \min \{F^{k-1}(i, j), F^{k-1}(i, k) + F^{k-1}(k, j)\}$$



- Computation of $V(i, j, k)$'s proceeds level by level.

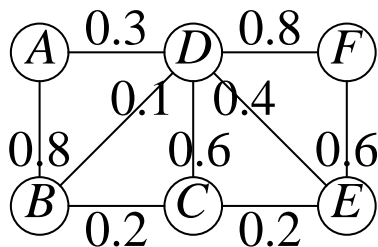
EXERCISE

1. Suppose the following path is a shortest x_3x_9 -path in some digraph G with non-negative link-costs. For each shortest subpath $\pi_m(x_i, x_j)$ of $\pi_m(x_3, x_9)$, indicate the smallest k (which may depend on i and j) such that we will *definitely* have $F^k(i, j) = |\pi_m(x_i, x_j)|$ no matter what the costs of other links in G are. Give an example to show that we may have $F^n(i, j) = |\pi_m(x_i, x_j)|$ for some $n < k$ in some cases, though this may not be guaranteed for all costs.



2. Suppose $0 \leq c(x, y) \leq 1$ for each link (x, y) and $a \wedge b = \min(a, b)$. We define the cost of a path $\pi(x_1, x_n) = \langle x_1, x_2, \dots, x_n \rangle$ by $C_{\min}(\pi) = c(x_1, x_2) \wedge c(x_2, x_3) \wedge \dots \wedge c(x_{n-1}, x_n)$ and let $M_{\min}(x, y) = \max\{C_{\min}(\pi) : \text{for all paths } \pi = \pi(x, y)\}$. If we think of $c(x, y)$ as the traffic flow from x to y , then $C_{\min}(\pi)$ is the traffic flow along the path π .

- (a) Show $M_{\min}(A, B)$, $M_{\min}(B, C)$, and $M_{\min}(A, C)$ and a corresponding path for each case for the digraph below.



- $c(x, y) = c(y, x)$ for all (x, y) .
- $c(x, y) = 0$ if (x, y) is not present.
- $c(x, x) = 1$ for all x .

- (b) Is it true that $M_{\min}(x, z) \geq \min\{M_{\min}(x, y), M_{\min}(y, z)\}$ for all x, y , and z ? Explain.

- (c) Which of the equations (i)-(iii) in Problem 2 on page 3.5 are still true for $d(x, y) = M_{\min}(x, y)$ and why?
- (d) Write recursive equations (similar to Floyd's equations) for computing $M_{\min}(x, y)$.
- (e) Is it true that we need to consider only the acyclic-paths in computing $M_{\min}(x, y)$ (why)?
3. [Recursive approach to computing all shortest-path lengths $d(i, j) = |\pi_m(x_i, x_j)|$.] Assume that you have computed the shortest-path lengths $d_N(i, j)$, $1 \leq i, j \leq N$, in the digraph G_N on the first N nodes $\{x_1, x_2, \dots, x_N\}$. Show how to compute $d_{N+1}(i, j)$, $1 \leq i, j \leq N + 1$, in the digraph G_{N+1} using the results for G_N . Verify your results using the digraph on page 3.12 (with $A = x_1$, $B = x_2$, etc.) and show the 4×4 matrix for $d_4(i, j)$'s and the 5×5 matrix of $d_5(i, j)$'s. Show the additional term in $T(N + 1) = T(N) + \dots$ using the appropriate notation $O(\cdot)$, or $\Theta(\cdot)$, or $\Omega(\cdot)$, where $T(N + 1)$ is the time required to compute $d_{N+1}(i, j)$'s; also, express $T(N)$ in the form $O(\cdot)$, $\Omega(\cdot)$, or $\Theta(\cdot)$, as appropriate.
4. Let $count^k(i, j) = \#(\text{acyclic } x_i x_j\text{-paths corresponding to } F^k(i, j), \text{ i.e., with length } F^k(i, j) \text{ and } \{x_1, x_2, \dots, x_k\} \text{ as possible intermediate nodes})$. Argue that the equations for computing $count^k(i, j)$ below are correct. For each k , $1 \leq k \leq 6$, show the matrix of $count^k(i, j)$ for the digraph with $N = 6$ nodes and the costs $c(x_i, x_j) = |i - j|$ for $1 \leq i \neq j \leq N$; show only the non-zero counts for improved readability. List the paths corresponding to $count^4(2, 6)$.

$$count^k(i, j) = \begin{cases} count^{k-1}(i, k) \times count^{k-1}(k, j), & \text{if } F^{k-1}(i, j) > F^{k-1}(i, k) + F^{k-1}(k, j) \\ count^{k-1}(i, j), & \text{if } F^{k-1}(i, j) < F^{k-1}(i, k) + F^{k-1}(k, j) \\ count^{k-1}(i, k) \times count^{k-1}(k, j) + count^{k-1}(i, j), & \text{if } F^{k-1}(i, j) = F^{k-1}(i, k) + F^{k-1}(k, j) \end{cases}$$

5. Let $\overline{count}^k(i, j) = \#(\text{acyclic } x_i x_j \text{ paths using the nodes } \{x_1, x_2, \dots, x_k\} \text{ as possible intermediate nodes and which have length } > F^k(i, j))$. Use the formula for the total number of acyclic paths among which $F^k(i, j)$ is the $x_i x_j$ shortest-path length and $count^k(i, j)$ to obtain a formula for $\overline{count}^k(i, j)$. Why is it not possible to obtain formulas for $\overline{count}^k(i, j)$ in a way similar to those for $count^k(i, j)$?
6. Let G be an *acyclic* digraph. Give suitable recursive equations for computing $numPaths_{ij} = \#(x_i x_j\text{-paths using one or more links in } G)$. Note that $numPaths_{ii} = 0$ for all i . If G is not acyclic, what will go wrong with the equations? What will be the equations if we define $numPaths_{ij}^{k+1} = \#(x_i x_j\text{-paths using exactly } k+1 \text{ steps})$? How about if $numPaths_{ij}^{k+1} = \#(x_i x_j\text{-paths using at most } k+1 \text{ steps})$?

A VARIATION OF FLOYD'S METHOD

The New Equations:

- Let $Z^k(i, j)$ = The shortest length of an $x_i x_j$ -path with *at most* k intermediate nodes (assume no negative cycle in \vec{G}).
- $Z^0(i, j) = c(x_i, x_j)$

$$Z^k(i, j) = \min \begin{cases} Z^{k-1}(i, j) \\ Z^{k-1}(i, q) + c(x_q, x_j), 1 \leq q \leq N \end{cases}$$

$$(1) |\pi_m(i, j)| \leq Z^k(i, j) \leq Z^{k-1}(i, j) \leq F^{k-1}(i, j) \leq c(x_i, x_j)$$

$$(2) |\pi_m(i, j)| = Z^{N-2}(i, j) \text{ for } i \neq j$$

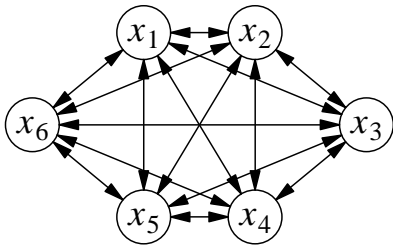
Comparison With Floyd:

- Complexity = $\theta(N^4)$, with $\Theta(N)$ work for each $Z^k(i, j)$.
- $F^k(i, j)$ converges to $|\pi_m(i, j)|$ slower than $Z^k(i, j)$ and yet takes less time for computation.
- This is also a D.P. method, with the same states as in Floyd's method, but with a different meaning of the states and a corresponding different relationship among them.
- Not all D.P. methods are equally good.

Question:

- ? How many acyclic paths are considered in the definition of $Z^k(i, j)$ for $i \neq j$ and in the equation for $Z^k(i, j)$?
- ? Which of the paths $\langle A, C, B \rangle$ and $\langle A, C, B, D \rangle$ in \vec{G} on page 3.3 are looked at by Floyd's and the Z^k -method (explain)?
- ? Can we say $Z^k(i, j) = \min\{Z^{k-1}(i, j), c(x_i, x_q) + Z^{k-1}(q, j): 1 \leq q \leq N\}$? Will both forms of Z^k look at the same paths?

FINDING AN n -STEP SHORTEST-PATH



- \vec{G} may have a negative-cost cycle.
- $S^n(i, j)$ = The shortest length of an n -step $x_i x_j$ -path.
- We are not restricting to acyclic paths as verification of acyclicity becomes expensive.

Example. Paths $\pi(1, 6)$ using 3 steps.

$\langle 1, 1, 1, 6 \rangle, \langle 1, 1, 2, 6 \rangle, \dots$

The length of shortest path in this group is $c(x_1, x_2) + S^2(2, 6)$

$\left\{ \begin{array}{l} \langle 1, 2, 1, 6 \rangle, \langle 1, 2, 2, 6 \rangle, \\ \langle 1, 2, 3, 6 \rangle, \langle 1, 2, 4, 6 \rangle, \\ \langle 1, 2, 5, 6 \rangle, \langle 1, 2, 6, 6 \rangle \\ \dots \dots \end{array} \right.$

$$S^3(1, N) = \min_{1 \leq p \leq N} \{c(x_1, x_p) + S^2(p, N)\}$$

General Case:

- $S^1(i, j) = c(x_i, x_j)$ for all i, j

$$S^n(i, j) = \min_{1 \leq p \leq N} \{c(x_i, x_p) + S^{n-1}(p, j)\}, \text{ for } n \geq 2$$

- One can also keep track of the paths along with the path-length computations.

Complexity: $\Theta(N^2 + (n-1) \cdot N \cdot N^2) = \Theta(n \cdot N^3)$ for n -step shortest paths for all (i, j) -pairs.

Question: What is $\#(n\text{-step } x_i x_j\text{-paths})$? What does it say about the above method for computing $S^n(i, j)$?

HISTOGRAM EQUALIZATION

Problem: Decompose the list of numbers $L = \langle n_1, n_2, \dots, n_N \rangle$ into $k > 1$ groups of consecutive items (in short, a k -grouping) such that: each group-total is as close to the ideal value T/k , where $T = n_1 + n_2 + \dots + n_N$. More precisely, if $s_i = i$ th group-total, then minimize

$$E = \sum_{i=1}^k (s_i - T/k)^2 = \text{the sum of squared errors.}$$

Example. $L = \langle 3, 2, 1, 1, 2 \rangle$ and $k = 3$; $T/k = 3$.

The groups in a 3-grouping			Group totals
$\langle 3 \rangle$	$\langle 2 \rangle$	$\langle 1, 1, 2 \rangle$	3, 2, 4
$\langle 3 \rangle$	$\langle 2, 1 \rangle$	$\langle 1, 2 \rangle$	3, 3, 3 ← optimal solution
$\langle 3 \rangle$	$\langle 2, 1, 1 \rangle$	$\langle 2 \rangle$	3, 4, 2
$\langle 3, 2 \rangle$	$\langle 1 \rangle$	$\langle 1, 2 \rangle$	5, 1, 3
$\langle 3, 2 \rangle$	$\langle 1, 1 \rangle$	$\langle 2 \rangle$	5, 2, 2
$\langle 3, 2, 1 \rangle$	$\langle 1 \rangle$	$\langle 2 \rangle$	6, 1, 2

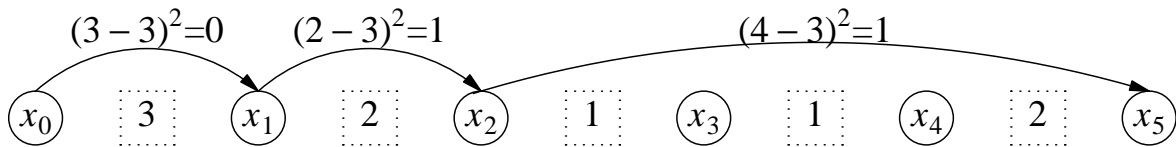
Question:

- ? What is $\#(k\text{-groupings})$? How to generate them systematically and how long will it take?
- ? If $s_{i,j} = n_{i+1} + n_{i+2} + \dots + n_j$, then show a table of all $s_{i,j}$ that are relevant in the optimal k -grouping problem? How many of them are there and what is the complexity for an efficient algorithm for computing them?
- ? Give the pseudocode for finding an optimal k -grouping of N items using the $s_{i,j}$'s. Give its complexity.

k -STEP SHORTEST-PATH FORMULATION

Digraph \vec{G}_N :

- $V = \{x_0, x_1, \dots, x_N\}$, $\vec{E} = \{(x_i, x_j): 0 \leq i < j \leq N\}$, where the link (x_i, x_j) corresponds to the group $\{n_{i+1}, n_{i+2}, \dots, n_j\}$.
- Each k -step x_0x_N -path give a k -grouping and vice-versa.
- Let $c(x_i, x_j) = (s_{i,j} - T/k)^2$; a shortest k -step x_0x_N -path gives an optimal k -grouping.



For $L = \langle 3, 2, 1, 1, 2 \rangle$ and the 3-step x_0x_5 -path $\langle x_0, x_1, x_2, x_5 \rangle$, the associated 3-grouping is $\{\langle 3 \rangle, \langle 2 \rangle, \langle 1, 1, 2 \rangle\}$, with cost $0+1+1 = 2$.

Question

- ? Show the weighted digraph \vec{G}_5 for $L = \langle 3, 2, 1, 1, 2 \rangle$ and the table of $S^n(0, j)$, $n \leq j$ and $1 \leq n \leq 3$.
- ? Give a formula to compute $S^n(0, j)$'s for the digraph \vec{G}_N that would lead to an efficient computation of $S^k(0, N)$.
- ? Verify your formula for $k = 3$ and $L = \langle 3, 2, 1, 1, 2 \rangle$. Mark the items $S^n(0, j)$ that will be computed in the process.
- ? Give the exact number of additions involving one or more $c(x_i, x_j)$ in computing $S^k(0, N)$; show sufficient details.
- ? What is the (total) complexity of this method in terms of k and N ? Why can we call this method a D.P. (show states and their relevant relationship)?

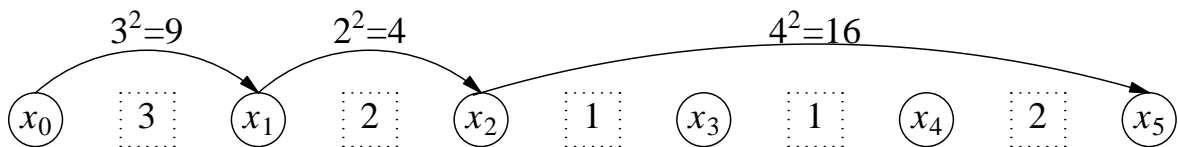
AN ALTERNATIVE LINK-COSTS FOR HISTOGRAM-EQUALIZATION

A Link-Cost Independent of k and T/k :

- Consider a k -steps x_0x_N -path $\pi = \langle x_0, x_{i_1}, x_{i_2}, \dots, x_{i_{k-1}}, x_N \rangle$. Let $s_1 = s_{0,i_1}$, $s_2 = s_{i_1,i_2}$, \dots , $s_k = s_{i_{k-1},N}$ and $a = T/k$. The cost of π is

$$\begin{aligned} & (s_1 - a)^2 + (s_2 - a)^2 + \dots + (s_k - a)^2 \\ &= [s_1^2 + s_2^2 + \dots + s_k^2] - 2a[s_1 + s_2 + \dots + s_k] + ka^2 \\ &= [s_1^2 + s_2^2 + \dots + s_k^2] - 2aT + ka^2 \end{aligned}$$

- Minimizing the cost of π is the same as minimizing the associated $[s_1^2 + s_2^2 + \dots + s_k^2]$.
- This means we can replace the link-cost $(s_i - a)^2$ simply by s_i^2 , which is independent of k and T/k .



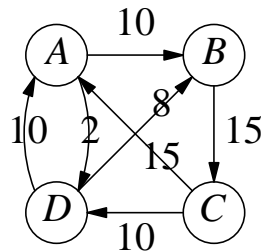
Some of the new link-costs for $L = \langle 3, 2, 1, 1, 2 \rangle$; the new cost of path $\langle x_0, x_1, x_2, x_5 \rangle$ is 29 and the old cost = 29 - 2 * 3 * 9 + 3 * 3^2.

Computing Link-Costs: //some not used in finding $S^k(0, N)$

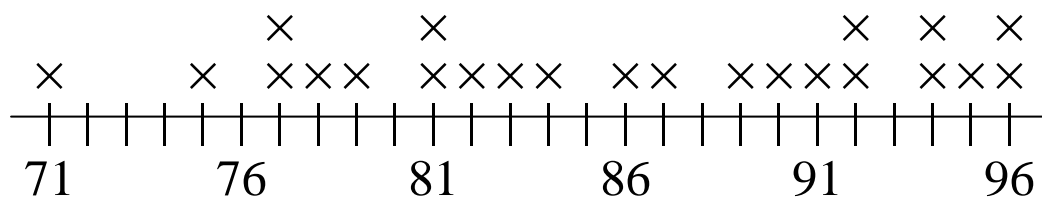
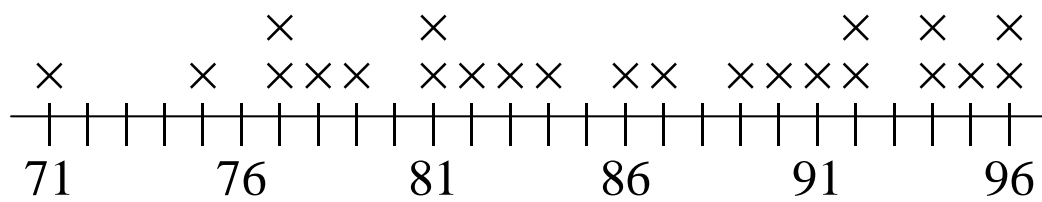
```
for (i=0; i<numItems; i++)
  for (sum=0, j=i; j<numItems; j++) {
    sum += items[j];
    linkCosts[i][j+1] = sum*sum;
  }
```

EXERCISE

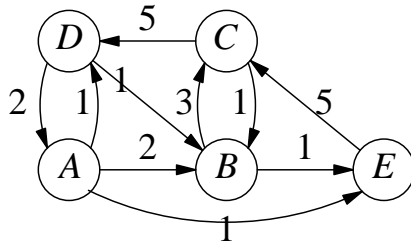
1. **Travelling salesman problem (TAP):** Given a digraph with link costs $c(x, y) \geq 0$, find a hamiltonian cycle (i.e., a cycle that goes through each node exactly once) which has the smallest cost. For the digraph below, <



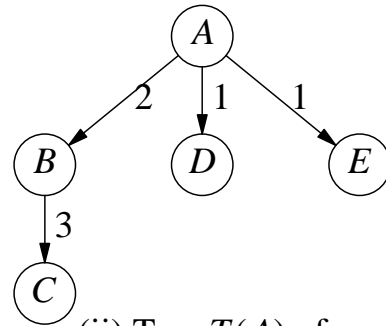
- (a) Why is it that we can drop the assumption $c(x, y) \geq 0$? (In other words, the condition $c(x, y) \geq 0$ does not make it easier or more difficult to solve TSP.)
- (b) Why can't we use the shortest-path algorithm to solve the TSP?
- (c) Why can't we use the k -step shortest-path algorithm to solve the TSP?



SHORTEST PATHS FROM A START-NODE TO ONE OR MORE OTHER NODES



(i) A slightly modified form of \vec{G} page 3.4; $c(E, C) = 5$.



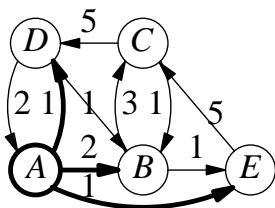
(ii) Tree $T(A)$ of shortest paths from A.

Method

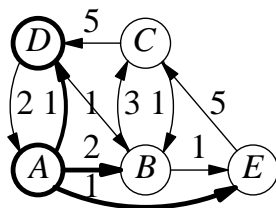
- Exploits input-property "each $c(x, y) \geq 0$ " and the output-property "tree-structure of shortest-paths from start-node s ".
- Maintains a tree of currently best known paths $\pi(s, x)$ from s for various x ; $d(x) = |\pi(s, x)|$ and $\text{parent}(x) = \text{parent of } x$.
- Extends $\pi(s, x)$ by adding links (x, y) from x *only* if $\pi(s, x) = \pi_m(s, x)$, i.e., $x = \text{a terminal node}$ and $d(x)$ is minimum among terminal nodes. This step is called *closing* of x .

Successive States of the Tree of Current $\pi(s, x)$'s:

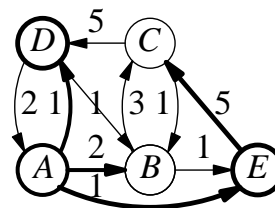
- The tree links are shown in bold.
- The closing of B gives the final tree $T(A)$.



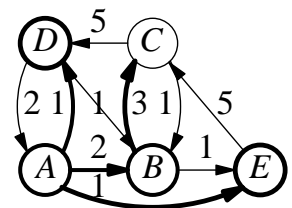
(i) Closing $s = A$.



(ii) Closing D.



(iii) Closing E.



(iv) Closing B; $\text{parent}(C)$ changed.

DIJKSTRA'S ALGORITHM

Terminology ($\text{OPEN} \cap \text{CLOSED} = \emptyset$):

$\text{CLOSED} = \{x: \pi_m(s, x) \text{ is known}\}$.

$\text{OPEN} = \{x: \text{some } \pi(s, x) \text{ is known but } x \notin \text{CLOSED}\}$.

Algorithm DIJKSTRA (shortest paths from s):

Input: AdjList(x) for each node x in \vec{G} , each $c(x, y) \geq 0$, a start-node s , and possibly a goal-node g .

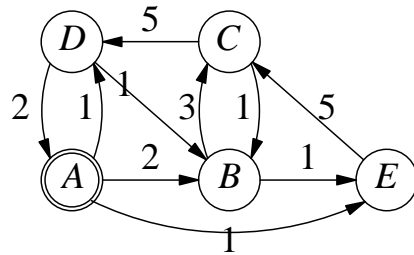
Output: A shortest sg -path (or sx -path for each x reachable from s).

1. [Initialize.] $d(s) = 0$, mark s OPEN, $\text{parent}(s) = s$ (or NULL), and all other nodes are unmarked.
2. [Choose a new closing-node.] If (no OPEN nodes), then there is no sg -path and stop. Otherwise, choose an OPEN node x with the smallest $d(\cdot)$, with preference for $x = g$. If $x = g$ or all but one node are closed, then stop.
3. [Close x and Expand $\pi_m(s, x)$.] Mark x CLOSED and for (each $y \in \text{adjList}(x)$ and y not marked CLOSED) do:
 - if (y not marked OPEN or $d(x) + c(x, y) < d(y)$) then let $\text{parent}(y) = x$, $d(y) = d(x) + c(x, y)$, and mark y OPEN.
4. Go to step (2).

Complexity: $O(N^2)$.

- A node x is marked CLOSED at most once and hence a link (x, y) is processed at most once.
- Each iteration of steps (2) and (3) takes $O(N)$ time.

ILLUSTRATION OF DIJKSTRA'S ALGORITHM



- A digraph \vec{G} with each $c(x, y) \geq 0$.
- Start-node $s = A$.

"?,?" indicates unknown values, "... " indicates no changes, and "-" indicates path-length not computed (would not have changed any way).

Open Nodes	Node Closed	Links processed	$d(x)$ and $\text{parent}(x)$				
			A	B	C	D	E
{A}	\emptyset		0, A	?, ?	?, ?	?, ?	?, ?
{B}	A	(A, B)		2, A			
{B, D}		(A, D)				1, A	
{B, D, E}		(A, E)					1, A
{B, E}	D	(D, A) (D, B)	-	...			
{B, C}	E	(E, C)			6, E		
{C}	B	(B, C) (B, E)			5, B		-
\emptyset	C	(C, B) (C, D)		-		-	

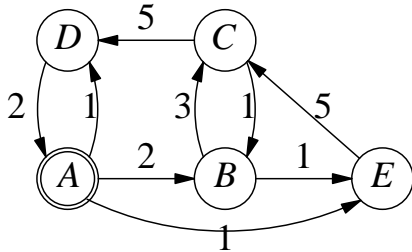
Question:

- ? List all sx -paths that are looked at (length computed) above. Are all these also looked at by Floyd? Is there an sx -paths looked at by Floyd but not by Dijkstra?
- ? What might happen if some $c(x, y) < 0$?
- ? Why is this not a D.P. method (but a special form of Search)?

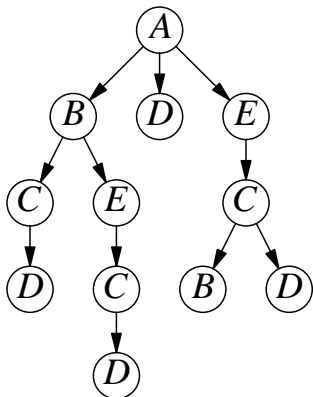
EXERCISE

1. Is it possible that we might find a shortest sg -path before we process all links to g (cf. the longest-path algorithm)?
2. Give an algorithm to compute a best sg -path based on $C_{\min}(\pi)$ as in Problem 2, page 3.15. It should work even if \vec{G} is not symmetric. Illustrate the algorithm by computing a best AE -path after changing $c(D, E) = c(E, D) = 0.1$.
3. Suppose that $c(y, z)$ is reduced by c_0 after computing the tree $T(s)$ of shortest-paths from s . Verify your answer for the following questions by considering the digraph on page 3.4 and $s = B$, $y = C$, and $z = D$ (if necessary, create your own digraph to illustrate your answer).
 - (a) Assume first $(y, z) \in T(s)$. Is it true that for each node w in the subtree $T_z(s)$ of $T(s)$ at z , the shortest-path $\pi_m(s, w)$ remains the same though its length $|\pi_m(s, w)|$ is reduced by c_0 ? What may happen to shortest paths and their lengths for other nodes?
 - (b) What happens if $(y, z) \notin T(s)$?
 - (c) Repeat (a)-(b) for the case of $c(y, z)$ going up by c_0 .
4. Which of the answers in (a)-(b) may not hold if c_0 is so large that the reduced cost of (y, z) has become negative?
5. State the algorithms to update the shortest sx -paths for $c_0 > 0$ (but the new cost of (y, z) still positive) and for $c_0 < 0$, i.e., the cost of (y, z) going up and $y = \text{parent}(z)$. In each case, show proper initializations of CLOSED/OPEN nodes and $d(\cdot)$'s, without changing their meaning.

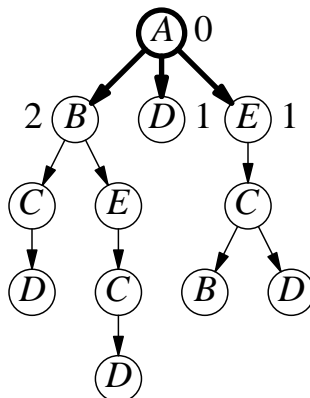
PATH ELIMINATION IN DIJKSTRA'S ALGORITHM



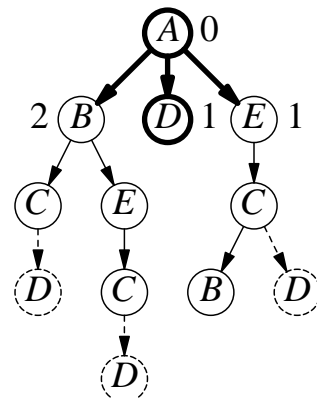
- Start-node $s = A$.
- The link (D, B) is removed to keep the tree of acyclic paths from A small.



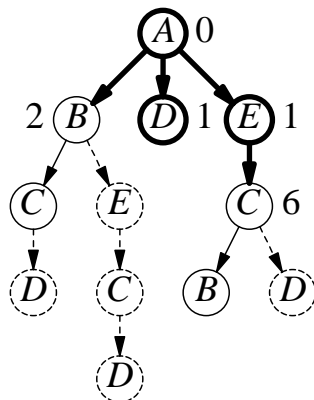
(i) Tree of acyclic paths from A .



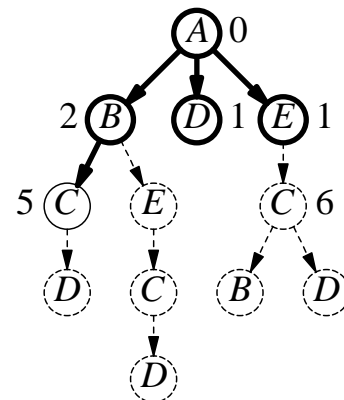
(ii) After closing root A with $d(A) = 0$.



(iii) After closing D with $d(D) = 1$; no other path to D or their extensions evaluated.



(iv) After closing E with $d(E) = 1$.



(v) After closing B with $d(B) = 2$.

- At each stage, there is at most one copy of a node x with a known distance $d(x)$.
- On closing x , the subtrees at all but one copy of x (with minimum $d(x)$) are eliminated along with the associated paths.
- We can view this as a D.P. formulation, where the state-space = nodes of the path-tree (which depends on the input \vec{G}). Computations proceed with elimination of some states; the goal-state(s) are not known ahead of time. Put another way, they are updated during the computation.

EXERCISE

1. Assume that each $c(x, y) \geq 0$ and let $d(s, y) = |\pi_m(s, y)|$ for a *fixed* start-node s . Which of the following are true?
 - (a) $d(s, y) = \min\{d(s, z) + c(z, y) : z \neq y\}$
 - (b) $d(s, y) = \min\{c(s, z) + d(z, y) : z \neq y\}$
 - (a') $d(s, y) = \min\{d(s, z) + c(z, y) : \text{all } z\}$
 - (b') $d(s, y) = \min\{c(s, z) + d(z, y) : \text{all } z\}$

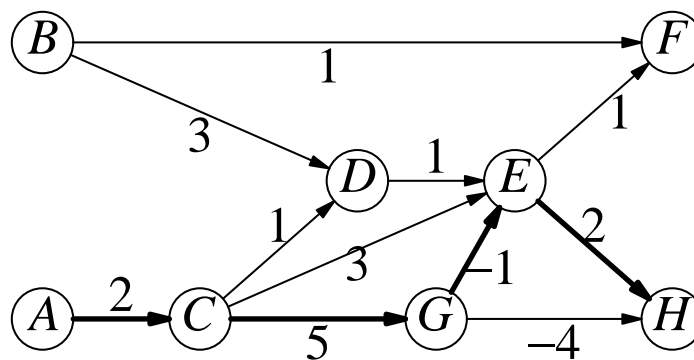
Verify your answer using the digraph on page 3.4 and $s = A$. Do we need to assume "each $c(x, y) \geq 0$ " – explain?

2. Why can't we use (a) (or b) above to compute $d(s, y)$'s?
3. Suppose we label the nodes in \vec{G} as y_1, y_2, \dots, y_N such that $d(s, y_1) \leq d(s, y_2) \leq \dots \leq d(s, y_N)$. Here, $y_1 = s$. Give an alternative (and computationally more useful) form of (a) based on the labeling y_i .
4. What is the connections between Dijkstra's algorithm and the alternative form for $d(s, y)$'s in Problem 3?
5. Give a similar modified form for (b) above, if any.
6. If \vec{G} is a rooted tree, $s = \text{root}$, and the links are directed from parent to child, give a linear algorithm to find $|\pi_m(s, x)|$ for all nodes x . Explain your algorithm using a small tree.
7. Give a shortest-path formulation (keep all $c(x, y) \geq 0$) for the LIS problem. Show \vec{G} for $\text{nums}[1..9] = [4, 5, 2, 9, 7, 6, 8, 1, 3]$. How much time is needed to construct \vec{G} and find an LIS from \vec{G} ? Could we eliminate some links in \vec{G} during its construction and still obtain an LIS? Would it be worthwhile?

LONGEST PATHS IN WEIGHTED ACYCLIC DIGRAPH

Longest Path:

$\pi_M(x, y)$ = an xy -path which has the largest length among all possible xy -path. If there is no xy -path, let $|\pi_M(x, y)| = -\infty$.



A longest AH -path: $\pi_M(A, H) = \langle A, C, G, E, H \rangle$,
 $|\pi_M(A, H)| = 2+5-1+2 = 8$.

Question:

- ? For an acyclic digraph \vec{G} with N nodes, what is the maximum possible number of xy -paths and when does it happen?
- ? Formulate the LIS problem for an input $\text{nums}[1..N]$ as a longest-path problem by constructing an acyclic weighted digraph \vec{G}_{nums} such that each increasing subsequence corresponds to an xy -path (for some node-pair x and y) and vice-versa. Keep each weight non-negative. Show the digraph for $\text{nums}[] = [4, 5, 2, 9, 7, 6, 8, 1, 3]$.
- ? What is the complexity of an algorithm to build your \vec{G}_{nums} ?
- ? Give a shortest-path formulation (with non-negative weights) for the LIS problem. Show the digraph for $\text{items}[]$ as above.

FINDING THE LARGEST NUMBER OF EVENTS THAT ONE CAN ATTENDED

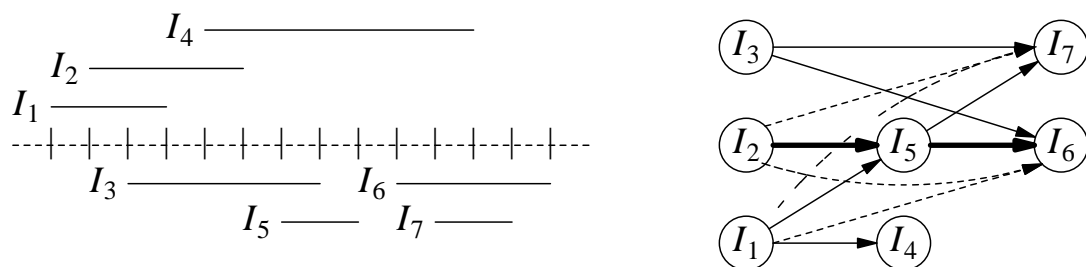
Problem:

- Let $I = \{I_1, I_2, \dots, I_n\}$ be a set of intervals, where I_i = duration of the event e_i in a festival.
- Find $I' \subseteq I$ such that the intervals I' are mutually *disjoint* (so that one can attend those events) and $|I'|$ is maximum.

Precedence-Digraph \vec{G}_I :

- $V = I$, $\vec{E} = \{(I_i, I_j) : I_i < I_j\}$, and $c(I_i, I_j) = 1$ for each link.
- If $I_i = [a, b]$ and $I_j = [a', b']$, then I_i *precedes* I_j , denoted by $I_i < I_j$, if $b \leq a'$. The relationship " $<$ " defines a partial order on the intervals.

Example. Let $I_1 = [1, 4]$, $I_2 = [2, 6]$, $I_3 = [3, 8]$, $I_4 = [5, 12]$, $I_5 = [7, 9]$, $I_6 = [10, 14]$, and $I_7 = [11, 13]$.

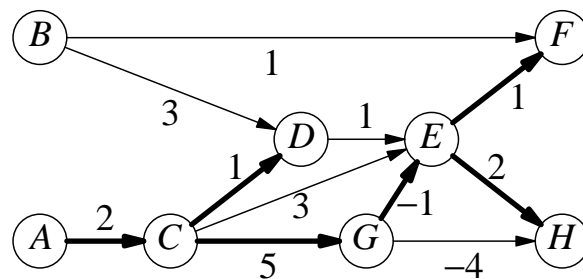


- We can eliminate the dashed links, which are implied by the transitivity of " $<$ ", without affecting the longest path.
- A longest path $\langle I_2, I_5, I_6 \rangle$ in \vec{G}_I gives a largest number of disjoint intervals. (There are several other longest paths.)

BELLMAN'S OPTIMALITY PRINCIPLE FOR LONGEST PATHS

Question:

- ? State Bellman's principle of optimality for longest acyclic paths and give an example digraph \vec{G} to show that it does not hold (keep \vec{G} small). (This means finding a longest acyclic $x_i x_j$ -path will be considerably more difficult in a general \vec{G} .)
- ? Show that the optimality principle holds for acyclic \vec{G} . Also, show that for each node y reachable from x we can choose a longest xy -path $\pi_M(x, y)$ such that the paths $\pi_M(x, y)$ form a tree $T_M(x)$ rooted at x . ($T_M(x)$ suggests that there might be an efficient algorithm for finding $\pi_M(x_i, x_j)$'s.)



The bold links show the tree $T_M(A)$.

- ? Can the following Floyd-like equations be used for computing $|\pi_M(x_i, x_j)|$, $1 \leq i \neq j \leq N$, if there is no positive cost cycle; here, $c(x_i, x_j) = -\infty$ (a large negative number) if $(x_i, x_j) \notin \vec{G}$ and $1 \leq k \leq N$? What are some $L^k(A, H)$?

$$L^0(i, j) = c(x_i, x_j) \text{ (assume } c(x_i, x_i) = 0 \text{ for } 1 \leq i \leq N)$$

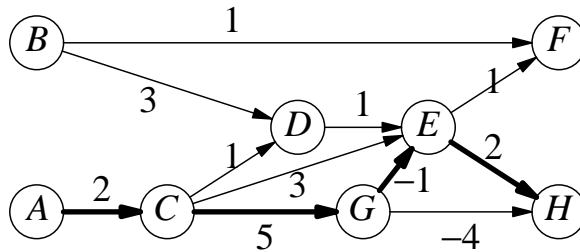
$$L^k(i, j) = \max \{L^{(k-1)}(i, j), L^{(k-1)}(i, k) + L^{(k-1)}(k, j)\}$$

$$L^N(i, j) = |\pi_M(x_i, x_j)|$$

LONGEST PATHS IN ACYCLIC DIGRAPHS

Optimality Principle:

- If $\pi = \langle x_1, x_2, \dots, x_n \rangle$ is a longest $x_1 x_n$ -path, then $\pi_{ij} = \langle x_i, x_{i+1}, \dots, x_j \rangle$ is a longest $x_i x_j$ -path for $1 \leq i < j \leq n$.



Some longest paths:

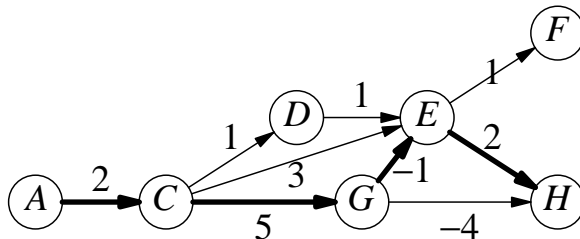
$$\pi_M(A, H) = \langle A, C, G, E, H \rangle$$

$$\pi_M(C, H) = \langle C, G, E, H \rangle$$

$$\pi_M(C, E) = \langle C, G, E \rangle$$

Method for computing $\pi_M(x, y)$:

- Eliminate links (u, v) which are not on any path from x (e.g., if $u \neq x$ and $\text{indeg}(u) = 0$).
- Expand an xz -path $\pi(x, z)$ only if it is a longest xz -path.
- $|\pi_M(x, y)| = \max \{ |\pi_M(x, z)| + c(z, y) : \text{all links } (z, y) \text{ to } y \}$.
In particular, all links to y must be processed before we can find $\pi_M(x, y)$.



- Reduced digraph for for $x = A$ and $y = H$.
- Could we reduce it further?
- $|\pi_M(A, H)| = 8 = \max\{7 - 4, 6 + 2\}$

Question:

- Is $|\pi_M(x, z)| = \max \{ c(x, y) + |\pi_M(y, z)| : \text{all } (x, y) \}$? Verify it for $|\pi_M(C, H)|$ and $|\pi_M(A, H)|$ using the digraph above.

ALGORITHM FOR LONGEST PATH

Notations:

- L : $\{z: \pi_M(x, z)$ is found, but it is not yet expanded $\}$.
 parent(z): the node preceding z on the current longest xz -path.
 $d(z)$: the current longest xz -path length.

Algorithm LONGESTPATH:

Input: An *acyclic* weighted \vec{G} , the start-node x , and the goal-node y ($\text{outdeg}(x) > 0$ and $\text{indeg}(y) > 0$).

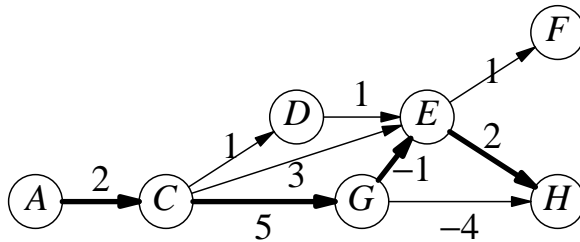
Output: A longest xy -path, if any.

1. Compute $\text{indeg}(z)$ of each node z .
2. [Reduce G .] Successively eliminate nodes z and all links (z, w) from z updating $\text{indeg}(w)$, if $\text{indeg}(z) = 0$.
3. [Initialize.] Let $L = \langle x \rangle$, $\text{parent}(x) = x$, $d(x) = 0$, and $d(z) = -\infty$ (a large negative number) for $z \neq x$.
4. Repeat the following until (L is empty or y is added to L):
 - (4.1) Let $z =$ an item in L ; remove (closing node) z from L .
 - (4.2) [Expand $\pi_M(x, z)$.] For (each link (z, w) from z) do:
 - (a) Reduce $\text{indeg}(w)$ by 1 and if $\text{indeg}(w) = 0$, then add w to L .
 - (b) If $(d(w) < d(z) + c(z, w))$, then let $d(w) = d(z) + c(z, w)$ and $\text{parent}(w) = z$.

Complexity: $O(|E|)$

- Each link is processed at most once using $O(1)$ time.

ILLUSTRATION OF LONGEST-PATH COMPUTATION



- Reduced digraph for for $x = A$ and $y = H$.
- $\pi_M(A, H) = \langle A, C, G, E, H \rangle$

Example. Parent(z) and $d(z)$ are final when z is added to L .

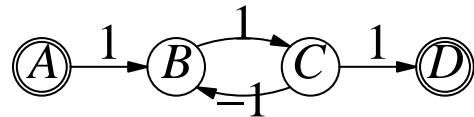
L	z	w	indeg(u) in parentheses, $d(u)$, parent(u), and path looked at							
			A	C	D	E	F	G	H	
$\langle A \rangle$			(0) 0, A	(1) $-\infty, ?$	(1) $-\infty, ?$	(3) $-\infty, ?$	(1) $-\infty, ?$	(1) $-\infty, ?$	(2) $-\infty, ?$	
$\langle C \rangle$	A	C		(0) 2, A						
$\langle D \rangle$	C	D			(0) 3, C					
$\langle D \rangle$		E				(2) 5, C				
$\langle D, G \rangle$		G						(0) 7, C		
$\langle G \rangle$	D	E				(1) 5, C				
$\langle E \rangle$	G	E				(0) 6, G				
		H							(1) 3, G	
$\langle F \rangle$	E	F					(0) 7, E			
		H							(0) 8, E	

Question:

- ? What is the maximum number of xz -paths looked at?
- ? Show all paths whose lengths are computed. Show an AH -path not looked at.

EXERCISE

1. If there is no xy path in G , how would we know this after step (2) of LONGESTPATH? If we do not reduce the digraph in step (2) of LONGESTPATH, then what would go wrong – explain using the example digraph and $x = A$ and $y = H$.
2. Explain what happens if we apply the algorithm LONGESTPATH to the digraph below for $x = A$ and $y = D$.



3. Can we find a longest path by changing each cost $c(u, v)$ to $-c(u, v)$ and apply Floyd or Dijkstra's algorithm? (Will it work, i.e., find the solution? Will it become less efficient, if it worked correctly?) Why is it that we do not perform node elimination in Dijkstra's algorithm like step (2) in LONGESTPATH?
4. Formulate the LIS problem as a longest-path problem in a digraph \vec{G} with $c(x, y) \geq 0$. Show \vec{G} for $\text{nums}[1..9] = [4, 5, 2, 9, 7, 6, 8, 1, 3]$ and a longest-path for an LIS.
5. Does the algorithm LONGESTPATH use any of the input-structure and output-structure? Explain. In what way, can we call LONGESTPATH a D.P. type algorithm?
6. Let $\text{numLongestPaths}(x_i, x_j) = \#(\text{longest } x_i x_j\text{-paths})$. Show the modifications to LONGESTPATH to compute $\text{numLongestPaths}(x, y)$ while computing $\pi_M(x, y)$.

LCS: LONGEST COMMON SUBSTRING

Substring:

- $C = c_1c_2\cdots c_p$ is a *substring* of $A = a_1a_2\cdots a_m$ if each $c_i = a_{j_i}$ such that $j_1 < j_2 < \cdots < j_p$. (It is same as a subsequence.)
- $C = abca$ is a *longest common substring (LCS)* of $A = abcab$ and $B = babbca$; $C = acb$ is a substring of only A .
- $\#(s, A) = \#(\text{occurrences of a symbol } s \text{ in } A)$; $\#(a, abcab) = 2$.

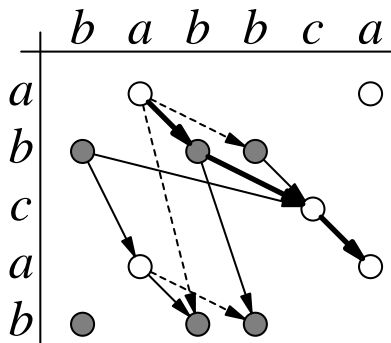
LCS Problem:

- Given strings $A = a_1a_2\cdots a_m$ and $B = b_1b_2\cdots b_n$ ($m \leq n$), find a longest common substring $LCS(A, B) = c_1c_2\cdots c_p$ ($p \leq m$).

Longest-Path Formulation in an Acyclic $\vec{G}(A, B)$:

Nodes: $\langle i, j \rangle$ if $a_i = b_j$; $\#(\text{nodes}) = \sum \#(s, A) \times \#(s, B)$. The $2 \times 3 = 6$ nodes for $s = b$ are shown below as shaded circles.

Links: $(\langle i, j \rangle, \langle i', j' \rangle)$ if $i < i'$ and $j < j'$. This ensures $\vec{G}(A, B)$ is acyclic, with paths going downwards and to the right (see below). Each path corresponds to a common substring and vice-versa. Each link-weight is 1.



- The transitive links are not shown; their removal does not affect the longest paths and hence the *LCS*'s.
- The dashed lines are some other links whose removal do not affect $|LCS|$.
- $LCS = abca$ (the path in thick lines)

Question: For $A = a^m$ and $B = a^n$, $m \leq n$, how many links does $\vec{G}(A, B)$ have if we do not eliminate any link?

SIMPLIFICATION OF $\vec{G}(A, B)$

- For each $\langle i, j \rangle$ and for each symbol s , eliminate all links to the nodes $\{\langle i', j' \rangle: a_{i'} = s = b_{j'}\}$ except for $\langle i_s, j_s \rangle$ where
 - $i_s = i' > i$ is the first occurrence of s in A after a_i and $j_s = j' > j$ is the first occurrence of s in B after b_j .

For $\vec{G}(A, B)$ on the previous page, this eliminates the links from $\langle 1, 2 \rangle$ to $\langle 2, 4 \rangle$ and $\langle 5, 3 \rangle$ (shown as dotted lines) and to $\langle 5, 4 \rangle$ (a transitive link and not shown) for $s = b$.

It doesn't eliminate the transitive link from $\langle 1, 2 \rangle$ to $\langle 4, 6 \rangle$.

- $\langle i, j \rangle$ is now adjacent to at most 1 node in a row/column.

Question:

- ? Let $m_s = \#(s, A)$ and $n_s = \#(s, B)$. How many links $\vec{G}(A, B)$ has among the nodes $\{\langle i, j \rangle: a_i = s = b_j\}$ if we do not eliminate any link and if we eliminate links using the above rule?
- ? How long will it take to construct the links of the reduced $\vec{G}(A, B)$ using the above rule?
- ? State a similar rule to eliminate links to a node in $\vec{G}(A, B)$. Show the reduced form of $\vec{G}(A, B)$ using this rule and also using both this and the rule above; do not remove transitive links if they are not eliminated by one of the rules.
- ? Does this reduced digraph contain a path for an $LCS(A, B)$ (every $LCS(A, B)$)? What do you conclude from this?
- ? Give example strings A and B to show that we cannot reduce $\vec{G}(A, B)$, where a node $\langle i, j \rangle$ is adjacent to only the nearest $\langle i', j' \rangle$, $i' > i$ and $j' > j$.

D.P. METHOD FOR LCS

Notations: $A = a_1 a_2 \cdots a_m$ and $B = b_1 b_2 \cdots b_n$.

$A_i = a_1 a_2 \cdots a_i$, $i \leq m$, and $B_j = b_1 b_2 \cdots b_j$, $j \leq n$.

$LCS_{i,j} = LCS(A_i, B_j)$ and $|LCS_{i,j}| = \text{length of an } LCS_{i,j}$.

Equations for Computing $|LCS_{i,j}|$:

(1) $|LCS_{0,0}| = 0 = |LCS_{0,j}| = |LCS_{i,0}|$ for $i, j \geq 1$.

(2.1) $a_i = b_j = s$.

The last symbol in $LCS_{i,j}$ must be s . If a_i is matched with $b_{j'}$, $j' < j$, then we can also obtain an $LCS_{i,j}$ by matching a_i and b_j . Similarly for the case if $a_{i'}$, $i' < i$, is matched with b_j . Thus, $|LCS_{i,j}| = 1 + |LCS_{i-1,j-1}|$.

(2.2) $a_i \neq b_j$.

At most one of a_i and b_j can be the last symbol in $LCS_{i,j}$; hence, $|LCS_{i,j}| = \max \{|LCS_{i-1,j}|, |LCS_{i,j-1}|\}$

(3) $|LCS(A, B)| = |LCS_{m,n}|$. (To compute an $LCS(A, B)$, keep track of a "path" for $LCS_{m,n}$ as in Floyd's algorithm.)

Example. Let $A = cbabd$ and $B = aabcbd$.

		<i>a</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>d</i>
	0	0	0	0	0	0	0
<i>c</i>	0	0	0	0	1	1	1
<i>b</i>	0	0	0	1	1	2	2
<i>a</i>	0	1	1	1	1	2	2
<i>b</i>	0	1	1	2	2	2	2
<i>d</i>	0	1	1	2	2	2	3

Complexity: $O(mn)$, better than the longest-path approach.

FINDING LCS FROM THE TABLE

Example. Let $A = cbabd$ and $B = aabcd$.

			<i>a</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>d</i>
		0	0	0	0	0	0	0
<i>c</i>	0	0	0	0	0	1	1	1
<i>b</i>	0	0	0	1	1	2	2	2
<i>a</i>	0	1	1	1	1	2	2	2
<i>b</i>	0	1	1	2	2	2	2	2
<i>d</i>	0	1	1	2	2	2	2	3

- The links to the node $|LCS_{i,j}|$ shows how it is obtained from one or more of $|LCS_{i-1,j}|$, $|LCS_{i-1,j-1}|$, and $|LCS_{i,j-1}|$.
- The bold links show increase in $|LCS_{i,j}|$ from its three preceding neighbors.
- Although $8 = \#(\text{paths to node } |LCS_{5,6}| = 3)$, there are only 7 ways of matching the symbols of A and B to give $|LCS| = 3$.

How do you describe the paths that correspond to the different ways giving an LCS ? Many such path may give the same LCS ; the 3 different LCS above are:

abd (4 ways of getting this), *bbd* (1 way of getting this),
and *cbd* (2 ways of getting this)

COUNTING THE NUMBER OF LCS

$$\text{count}(i, j) = \#(\text{LCS between } A_i \text{ and } B_j).$$

Equations for Computing $\text{count}(i, j)$:

- $\text{count}(1, j) = 1$ for $1 \leq j \leq n$ (one $\text{LCS}_{1,j}$ of length 0 or 1).
 $\text{count}(i, 1) = 1$ for $1 \leq i \leq m$ (one $\text{LCS}_{i,1}$ of length 0 or 1).
- $a_i = b_j$:
 $\text{count}(i, j) =$
 $\text{count}(i-1, j-1)$ (extension of $\text{LCS}_{i-1,j-1}$ uses a_i and b_j)
 $+ \text{count}(i-1, j)$, if $\text{LCS}_{i-1,j} = \text{LCS}_{i,j}$ (uses only b_j)
 $+ \text{count}(i, j-1)$, if $\text{LCS}_{i,j-1} = \text{LCS}_{i,j}$ (uses only a_i)
- $a_i \neq b_j$:
 $\text{count}(i, j) = 0$
 $+ \text{count}(i, j-1)$, if $\text{LCS}_{i,j-1} = \text{LCS}_{i,j}$ (not use b_j)
 $+ \text{count}(i-1, j)$, if $\text{LCS}_{i-1,j} = \text{LCS}_{i,j}$ (not use a_i)
 $- \text{count}(i-1, j-1)$, if $\text{LCS}_{i-1,j-1} = \text{LCS}_{i,j}$ (not use a_i or b_j)

Example. Let $A = cbab$ and $B = aabcb$. Shown below are the values of $\text{count}(i, j)$ in parentheses next to $\text{LCS}_{i,j}$.

	a	a	b	c	b	d
c	0 (1)	0 (1)	0 (1)	1(1)	1 (1)	1(1)
b	0 (1)	0 (1)	1 (1)	1(2)	2 (1)	2(1)
a	1 (1)	1 (2)	1 (3)	1(4)	2 (1)	2(1)
b	1 (1)	1 (2)	2 (2)	2(2)	2 (7)	2(7)
d	1 (1)	1 (2)	2 (2)	2(2)	2 (7)	3(7)

ANOTHER MORE EFFICIENT D.P. METHOD FOR COMPUTING $LCS(A, B)$

Complexity: $O(n + mp)$, where $|A| = m \leq n = |B|$ and $p = |LCS(A, B)|$.

Key Idea:

- Match each symbol of A with a symbol of B as early as possible.
- For each $A_j = a_1 a_2 \cdots a_j$, let

$$\text{match}[i] = \min \{k : |LCS(A_j, B_k)| = i\}.$$

As j increases to $j + 1$,

- the value of one or more $\text{match}[i]$ may be lowered
- the array $\text{match}[]$ may be extended by adding an items (obtain a longer common subsequence).
- For $j = m$, the last item of $\text{match}[]$ corresponds to an $LCS(A, B)$ and the associated $i = |LCS(A, B)|$.

We keep track of an actual $LCS(A_j, B_k)$ associated with each $\text{match}[i]$.

COMPLEXITY ANALYSIS

Cost of processing a_j : $O(p)$ For each symbol s , let $\text{FirstOccur}[s, i] = \min \{j: j > i \text{ and } b_j = s\}$.

	FirstOccur[s, i]											
	1	2	3	4	5	6	7	8	9	10	11	12
s	a	d	d	a	b	c	a	d	c	c	a	d
a	1	4	4	4	7	7	7	11	11	11	11	–
b	5	5	5	5	5	–	–	–	–	–	–	–
c	6	6	6	6	6	6	9	9	9	–	–	–
d	2	2	3	8	8	8	8	8	12	12	12	12

- Total cost of the algorithm is $O(n + mp)$.

EXERCISE

1. Compare the steps of this algorithm with those of longest-path computation in the digraph formulation and explain how this achieves better efficiency than a straightforward longest path computation.

D.P. METHOD FOR COMPUTING EDIT-DISTANCE BETWEEN TWO STRINGS

Problem:

For $A = a_1a_2\cdots a_m$ and $B = b_1b_2\cdots b_n$, $n \geq m$, find the edit-distance $ed(A, B)$ = the minimum number of single character insert(I)/delete(D)/replace(R) operations to convert A to B .

Equations For Computing $ed(A, B)$:

- Let $ed(i, j) = ed(A_i, B_j)$, where $A_i = a_1a_2\cdots a_i$ and $B_j = b_1b_2\cdots b_j$.
- (1) $ed(0, 0) = 0$, $ed(1, 0) = 1$ (delete a_1), and $ed(0, 1) = 1$ (insert b_1).
 - (2) $a_i = b_j$: $ed(i, j) = ed(i - 1, j - 1)$
 - (3) $a_i \neq b_j$: $ed(i, j) = \min \begin{cases} ed(i - 1, j) + 1, & \text{delete } a_i \\ ed(i - 1, j - 1) + 1, & \text{replace } a_i \text{ by } b_j \\ ed(i, j - 1) + 1, & \text{insert } b_j \end{cases}$

Example: Let $A = abc$ and $B = badc$.

$ed(i, j)$	b	a	d	c
a	1	1	2	3
b	1	2	2	2
c	2	2	3	2

$I(b)$ (insert b at the beginning of A): $abc \rightarrow babc$

$R(b, d)$ (replace next b by d): $babc \rightarrow badc$

Complexity: $\Theta(mn)$.

EXERCISE

1. If $a_i = b_j$, then explain why creating b_j by insertion or replacement from some other a_k ($k < i$) or equating with a_k provided $a_k = b_j$ (and then deleting the items a_{k+1}, a_{k+1}, \dots) does not give a lower value for $ed(i, j)$ than $ed(i - 1, j - 1)$, which corresponds to creating b_j from a_i .
2. If $c_I(a) = \text{cost}(\text{insertion of } a)$, $c_D(a) = \text{cost}(\text{deletion of } a)$, and $c_R(a, b) = \text{cost}(\text{replacement of } a \text{ by } b)$, then show the new equations for $ed(i, j)$. Show the table of $ed(i, j)$ for $A = abc$ and $B = badc$ using $I(a) = 1$, $D(a) = 2$, and $R(a, b) = 3$ for all a, b .
3. To simplify counting the number of edit-operation sequences for converting A to B , assume that each symbol of B is created in the left to right order by insertion of a symbol of B in A , or replacing a symbol of A to match a symbol of B (trivial replacement of a by a included), or a deletion of a symbol of A . Shown below are some valid and invalid operation-sequence for obtaining b from aa ; here, \bar{s} means addition of s and \underline{s} means deletion of s .

Valid: $\bar{b}\underline{a}\underline{a}$	Valid: $\underline{a}\bar{b}\underline{a}$	Invalid
1. $aa \rightarrow \bar{b}aa$	1. $\underline{a}a \rightarrow a$	1. $aa \rightarrow a\bar{b}a$
2. $b\underline{a}a \rightarrow ba$	2. $a \rightarrow \bar{b}a$	2. $ab\underline{a} \rightarrow ab$
3. $b\underline{a} \rightarrow b$	3. $b\underline{a} \rightarrow b$	3. $ab \rightarrow b$

For our purpose, we do not count them as distinct. To be precise, a consecutive sequence of deletions and insertions (without a replace operation) is considered to be equivalent to the one where all the deletions are done first and then all the insertions, and they are not counted as distinct.

Show that the places where the replacement operations are applied uniquely gives the whole operation sequence for converting s_1 to s_2 . Use this to determine $N(m, n) = \#(\text{operation-sequences for converting } s_1 \text{ to } s_2, \text{ where } |s_1| = m \leq n = |s_2|)$. Verify your formula by showing all operation-sequences for $s_1 = ab$ and $s_2 = eaf$. Note that $|s_1| + |s_2| = \#(\text{insertions}) + \#(\text{deletions}) + 2 \times \#(\text{replacements})$.

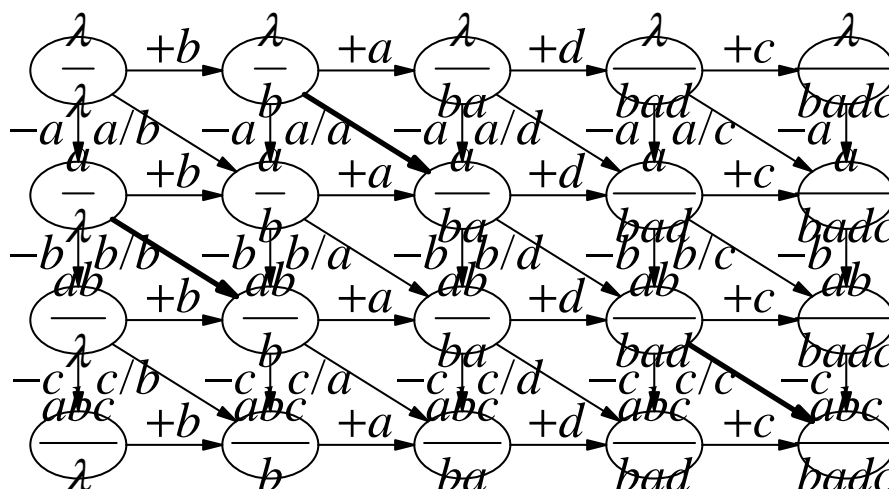
COMPUTATION OF EDIT-DISTANCE AS A SHORTEST-PATH

Notations: $op = +a$: Insert symbol a at the end of first string
 $op = -a$: Delete symbol a from the end of first string
 $op = a/b$: Replace symbol a at the end of first string

$\begin{matrix} s_1 \\ \hline s_2 \end{matrix} \xrightarrow{op} \begin{matrix} s'_1 \\ \hline s'_2 \end{matrix}$ if for a sequence of operations f, f' then $[f \cdot op](s'_1) = s'_2$, with op applied to s'_1

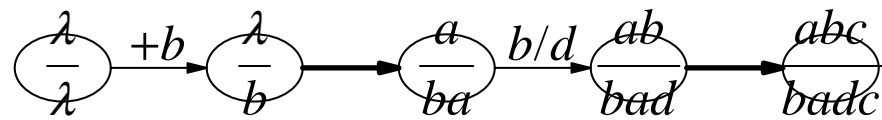
Digraph:

- The nodes are $(s_1(i), s_2(j))$, where $s_1(i)$ is the prefix of s_1 of length i whose symbols have been used so far for deletion and replacement and $s_2(j)$ is the prefix of s_2 of length j whose symbols have been used for insert and replacement.
- From each node, we have at most three links (see below) for the three kinds of operations. The thick links below have cost 0 corresponding to the operation a_i/b_j with $a_i = b_j$; all other links have cost 1.



The digraph for computing $ed(s_1, s_2)$ for $s_1 = abc$ and $s_2 = badc$

The unique shortest path is given by:



Theorem: $\|s_1\| - \|s_2\| \leq ed(s_1, s_2) \leq \min \{ \max(\|s_1\|, \|s_2\|), \|s_1\| + \|s_2\| - 2 \cdot |LCS(s_1, s_2)| \}$, where $LCS(s_1, s_2)$ is a longest common substring of s_1 and s_2 and $|LCS(s_1, s_2)|$ is its length.

Example. The lower bound occurs when s_1 is a substring of s_2 . The upper bound occurs for $s_1 = aaabb$ and $s_2 = bbcc$.

Example. In general, matching the parts of an LCS (i.e., using the maximum number of diagonal 0-cost links) may not give a shortest-path. For $s_1 = abbba$ and $s_2 = cccaacc$, $d(s_1, s_2) = 7$ and the matching of the LCS = aa gives a path of length 9 (= 6 insertions of c 's and 3 deletions of b 's).

EXERCISE

1. Show the digraph for $s_1 = abb$ and $s_2 = bab$ and show a path corresponding to $ed(s_1, s_2)$. Show two shortest paths for $s_1 = abbba$ and $s_2 = cccaacc$.

EXERCISE

1. Show that the algorithm for the edit-distance $ed(s_1, s_2)$ between two strings s_1 and s_2 satisfy the above criteria for D.P.. Do the same for the algorithm for an optimal scheme for a matrix multiplication $M_1 \times M_2 \times \cdots \times M_N$.
2. Can we view Dijkstra's shortest-path algorithm (from a node s to all nodes or from a node s to a node g) in the same way?
3. Can we view the depth-first algorithm for visiting the nodes of a graph as a DP by defining the state $s_i =$ the set of first i nodes visited (including their parents in the df-tree and other suitable information)? Is $s_i =$ the subproblem of finding a connected subgraph with i nodes containing the start-node is any good? (If v_i is the node added to s_{i-1} to obtain s_i , then $\text{parent}(v_i) = v_j$, with $j < i$ and as large as possible; this property is specific to the depth-first method.)
4. Argue that the insertion-sort method can be viewed as a DP. Show the states (and their meaning), the method for computing their values, the size of L_k , and the levels used in computing the values of states in L_k .