Missouri University of Science & Technology	Department of Computer Science
Spring 2024	**CS 2500: Algorithms (Sec: 102)**

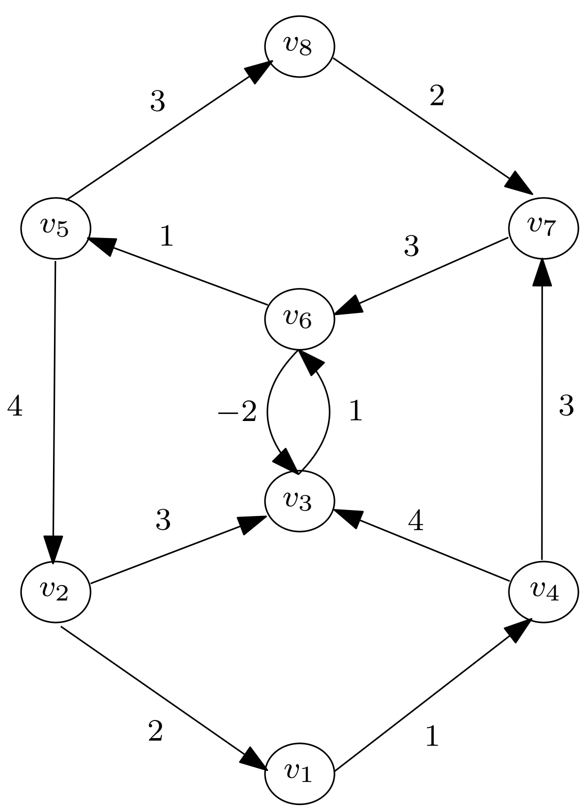
Solutions to HW 5: Dynamic Programming

Instructor: *Sid Nadendla* **Due:** *April 29, 2024*

Problem 1 Bellman-Ford Algorithm *5 points*

Demonstrate the value iteration at each subproblem within Bellman-Ford algorithm on the following graph, and clearly print the final output. Assume v_1 is the start node.

In each stage of the algorithm, clearly state the shortest distance estimate at each node from the source.

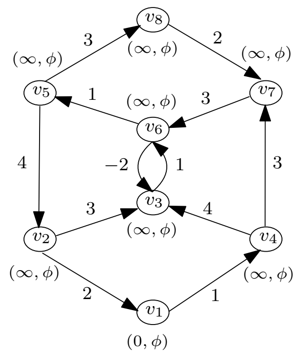


SOLUTION:

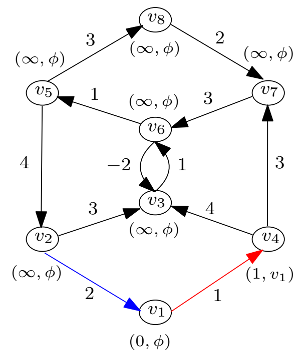
Edge relaxation order: $(v_1, v_4), (v_2, v_1), (v_2, v_3), (v_3, v_6), (v_4, v_3), (v_4, v_7), (v_5, v_2), (v_5, v_8), (v_6, v_3), (v_6, v_5), (v_7, v_6), (v_8, v_7)$

Node Attributes: $(distance\_estimate, parent)$

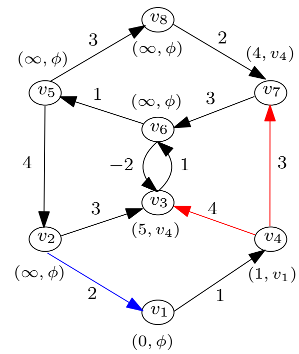
Successful edge relaxations in RED, failed edge relaxations in BLUE.



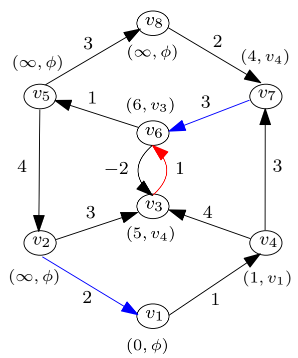
Initialization



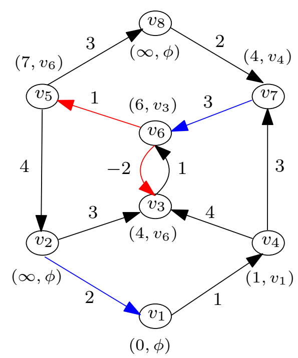
Iteration 1



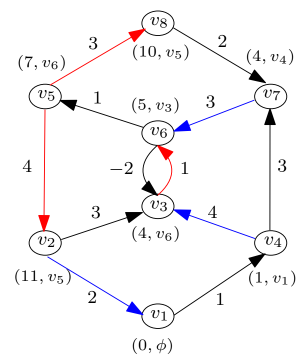
Iteration 2



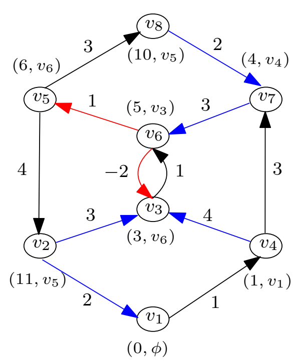
Iteration 3



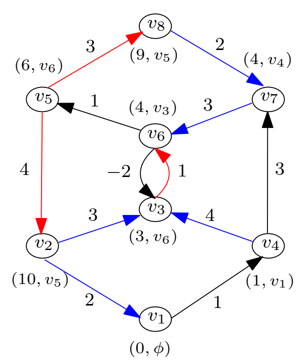
Iteration 4



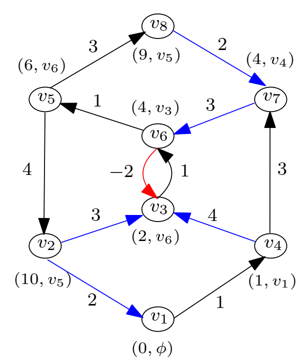
Iteration 5



Iteration 6



Iteration 7



Iteration 8

In the iteration $|V| = 8$, we observe an update of distance estimate due to edge relaxation.

In other words, we have a negative weight cycle.

Problem 2 String Edit Problem *5 points*

The *string edit* problem is to find the cheapest way to modify two strings so that they are the same. The permitted operations are *deletions*, *insertions* and *substitutions*.

Example: Consider two strings: ALKHWARIZMI and ALGORITHM. We need to perform the following sequence of operations in order to modify ALKHWARIZMI into ALGORITHM:

- Substitute K with G
- Substitute H with O
- Delete W
- Delete A
- Replace Z with T
- Insert H
- Delete I

Let the two strings be denoted as $a_1a_2 \cdots a_m$ and $b_1b_2 \cdots b_n$, where each a_i and each b_j are characters in the set S. If s_i and s_j are any two characters in S, let

- the cost of deleting s_i = $D_i > 0$
- the cost of inserting s_i = $I_i > 0$
- the cost of substituting s_i with s_j = $C_{ij} \geq 0$.

Assume $C_{ij} = C_{ji}$ for all i, j and $C_{ij} = 0$ if and only if $i = j$.

Then, present the following four stages of your design approach to this problem:

1. Model the above problem as a multi-stage decision problem, identify the state and decision variables, define the state transitions and derive the Bellman equation.
2. Using the Bellman equation, write a pseudocode to compute the optimal solution using dynamic programming approach.
3. Write down the pseudocode for the greedy solution to this problem.
4. Implement in Python, both the dynamic programming and greedy solutions to this problem and compare the value of the solutions returned for random pairs of strings.

SOLUTION:

1. Assume that the string-edit problem is solved in multiple-stages where, in each stage, the last symbol in the first sequence is modified using one of the three permitted operations,

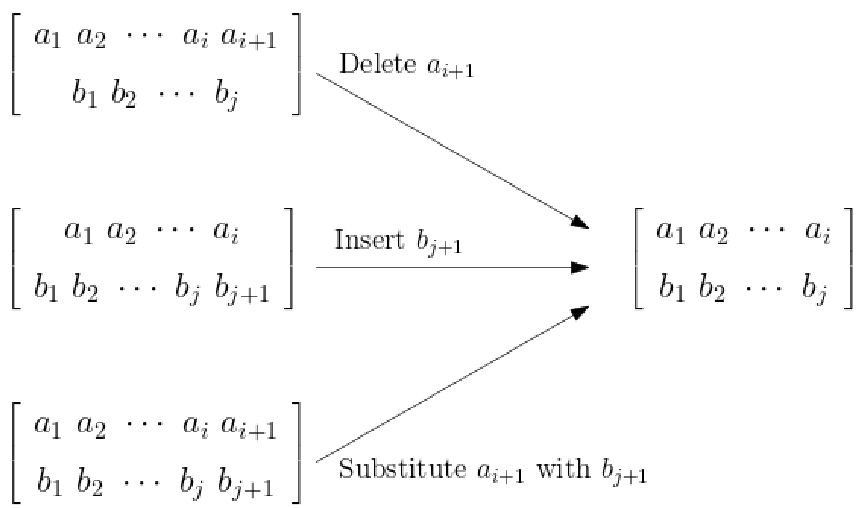
namely *deletion*, *insertion*, or *substitution*. At an arbitrary stage, we would typically have the following subproblem to be solved:

$$\begin{bmatrix} a_1\, a_2\, \cdots\, a_i \\ b_1\, b_2\, \cdots\, b_j \end{bmatrix}$$

Therefore, the **state** of the above arbitrary stage can be represented as (i, j). The **decision** variable at any stage can be modeled as

$$x[i,j] = \begin{cases} -1, & \text{if } a_i \text{ is deleted} \\ 0, & \text{if } b_j \text{ is inserted as } a_{i+1}. \\ 1, & \text{if } a_i \text{ is substituted by } b_j. \end{cases}$$

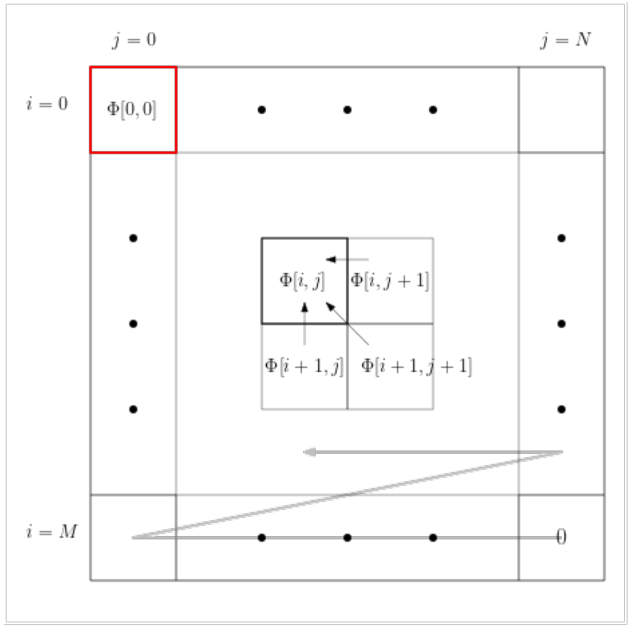
Note that the $(i, j)^{th}$ stage is manifested due to one of the three following decisions made in the prior stages:



The "*Delete* a_{i+1}" decision in the $(i + 1, j)^{th}$ stage leaves us with a residual problem of solving the problem in $(i, j)^{th}$ stage. Similarly, the "*Insert* b_{j+1}" decision in the $(i, j + 1)^{th}$ stage matches a_{i+1} with b_{j+1}, thus leaving us with a residual problem of solving the problem in $(i, j)^{th}$ stage. Finally, the "*Substitute* a_{i+1} with b_{j+1}" decision in the $(i + 1, j + 1)^{th}$ stage matches a_{i+1} with b_{j+1}, thus leaving us with a residual problem of solving the problem in $(i, j)^{th}$ stage. Note that one of these decisions will be result in the minimum cost. Therefore, if $\Phi[i, j]$ denotes the minimum cost to reach the $(i, j)^{th}$ stage, then the corresponding **Bellman recursion** will be given by

$$\Phi[i,j] = \min \begin{cases} \Phi[i+1,j] + D[a[i+1]], & \text{if } x[i+1,j] = -1, \\ \Phi[i,j+1] + I[b[j+1]], & \text{if } x[i+1,j] = -1, \\ \Phi[i+1,j+1] + C[a[i+1],b[j+1]], & \text{if } x[i+1,j] = -1. \end{cases}$$

2. Since the Bellman recursion shows that $\Phi[i,j]$ depends on $\Phi[i+1,j]$, $\Phi[i,j+1]$ and $\Phi[i+1,j+1]$, we design the recursion as shown in the figure below:



Note that the target is to reach $\Phi[0,0]$, since, upon matching the two strings $\boldsymbol{a}$ and $\boldsymbol{b}$, we are left with a subproblem that contains zero characters in $\boldsymbol{a}$ and $\boldsymbol{b}$ that need to be matched.

Therefore, the pseudocode for the dynamic programming algorithm to solve the String Edit problem is as follows:

STRINGEDIT-DP(a, b, D, I, C)

$\Phi = $ ZEROS$(M+1, N+1)$ // Initialize Φ as a $(M+1) \times (N+1)$ all-zero matrix.
for $i = M$ **to** 0
  for $j = N$ **to** 0
    if $i = M$ and $j \neq N$
      $\Phi[M,j] = \Phi[N,j+1] + I[b(j+1)]$
    else if $i \neq M$ and $j = N$
      $\Phi[i,N] = \Phi[i+1,N] + D[a(i+1)]$
    else

$$\Phi[i,j] = \min \begin{cases} \Phi[i+1,j] + D[a(i+1)], \\ \Phi[i,j+1] + I[b(j+1)], \\ \Phi[i+1,j+1] + C[a[i+1], b[j+1]] \end{cases}$$

return $\Phi[0,0]$

3. A simple greedy algorithm is as follows:

STRINGEDIT-GREEDY(a, b, D, I, C)

$K = \max\{a.\mathit{length}, b.\mathit{length}\}$
for $k = 1$ **to** K
    if $k \leq a.\mathit{length}$ and $k \leq b.\mathit{length}$
        $\Psi[k] = \Phi[k-1] + \min \begin{cases} D[a(k)], \\ I[b(k)], \\ C[a[k], b[k]] \end{cases}$
    else if $k > b.\mathit{length}$
        $\Psi[k] = \Phi[k-1] + D[a[k]]$
    else
        $\Psi[k] = \Phi[k-1] + I[b[k]]$
return $\Psi[K]$