CSCI 3104 Spring 2022
Instructors: Profs. Chen and Layer

Problem Set 8

Due Date .... March 29
Name .... **Julia Troni**
Student ID **109280095**
Collaborators .... **:)(:**

Contents

1 Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.

2 Standard 22 - Dynamic Programming: Bactracking to Find Solutions

Problem 1. Consider the following input for the Knapsack problem with a capacity $W = 11$:

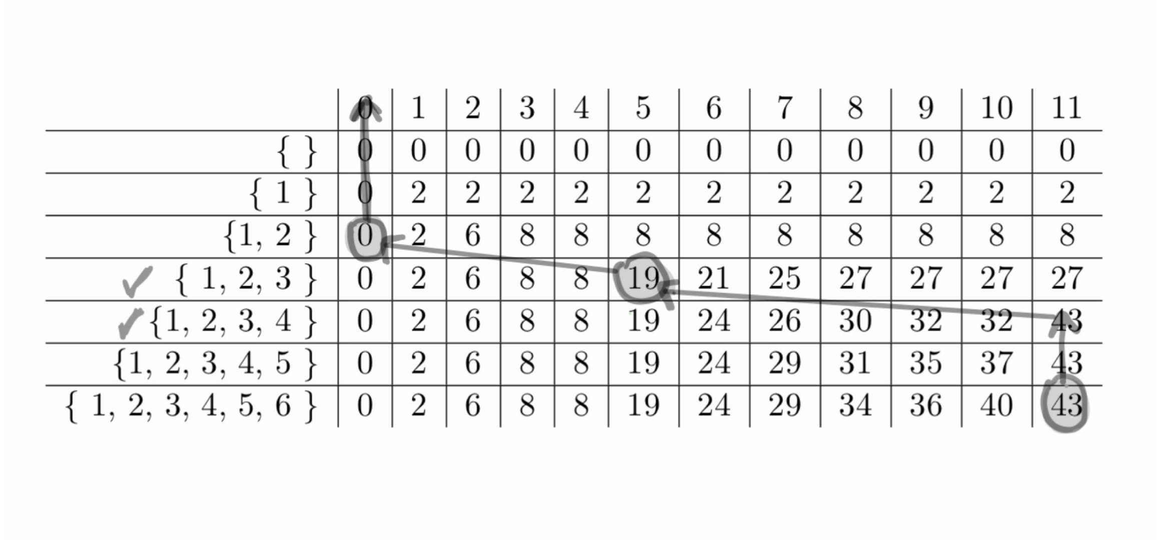
item i	1	2	3	4	5	6
value v_i	2	6	19	24	29	34
weight w_i	1	2	5	6	7	8

and the corresponding table for the optimal values/profits:

	0	1	2	3	4	5	6	7	8	9	10	11
{ }	0	0	0	0	0	0	0	0	0	0	0	0
{ 1 }	0	2	2	2	2	2	2	2	2	2	2	2
{1, 2 }	0	2	6	8	8	8	8	8	8	8	8	8
{ 1, 2, 3 }	0	2	6	8	8	19	21	25	27	27	27	27
{1, 2, 3, 4 }	0	2	6	8	8	19	24	26	30	32	32	43
{1, 2, 3, 4, 5 }	0	2	6	8	8	19	24	29	31	35	37	43
{ 1, 2, 3, 4, 5, 6 }	0	2	6	8	8	19	24	29	34	36	40	43

Draw the backward path consisting of backward edges to find the subset of items that has the optimal value/profit. Besides indicating the backward path, you must also give the optimal-value subset of items. Clearly explain your work.

Answer. The backward path is seen below:



	0	1	2	3	4	5	6	7	8	9	10	11
{ }	0	0	0	0	0	0	0	0	0	0	0	0
{ 1 }	0	2	2	2	2	2	2	2	2	2	2	2
{1, 2 }	0	2	6	8	8	8	8	8	8	8	8	8
✓ { 1, 2, 3 }	0	2	6	8	8	19	21	25	27	27	27	27
✓ {1, 2, 3, 4 }	0	2	6	8	8	19	24	26	30	32	32	43
{1, 2, 3, 4, 5 }	0	2	6	8	8	19	24	29	31	35	37	43
{ 1, 2, 3, 4, 5, 6 }	0	2	6	8	8	19	24	29	34	36	40	43

Since the table is filled in using this recurrence:
$Ifw < w_i : OPT(i, w) = OPT(i-1, w)$
Otherwise: $OPT(i, w) = max(OPT(i-1, w), v_i + OPT(i-1, w-w_i)$
Then the optimal subset of items is obtained through the following steps:

1. Begin in the bottom right most cell of the table, at position OPT[i,w] which denotes the max profit.
2. Trace up column w, until you reach a row r such that the value in the column directly above is different. That is when $OPT[r-1, w] \neq OPT[r, w]$. In this case, add the rth item to the subset.
3. Now subtract the value of the rth item from the value in cell $OPT[r, w]$.
4. Trace up and to the left until you find that corresponding value

5. Now repeat steps 2-5 until you reach OPT[0,0].

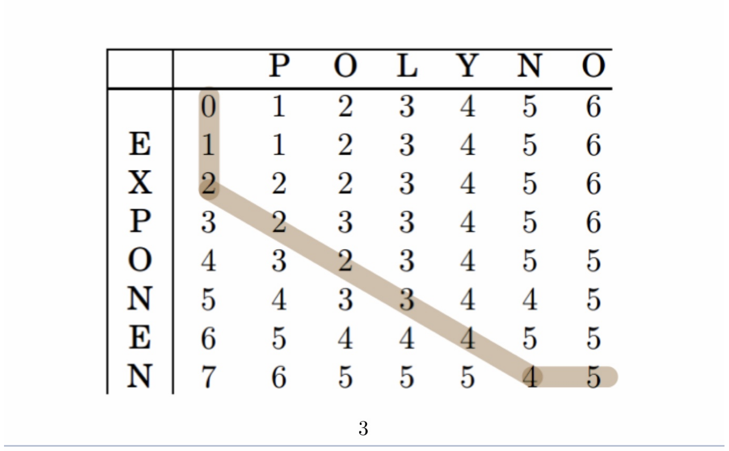
Thus the optimal-value subset of items is $OPT = \{4, 3\}$

$\square$

Problem 2. Recall the sequence alignment problem where the cost of *sub* and the cost of *indel* are all 1. Given the following table of optimal cost of aligning the strings EXPONEN and POLYNO, draw the backward path consisting of backward edges to find the minimal-cost set of edit operations that transforms EXPONEN to POLYNO. Besides indicating the backward path, you must also give the minimal-cost set of edit operations. Clearly explain your work.

		P	O	L	Y	N	O
	0	1	2	3	4	5	6
E	1	1	2	3	4	5	6
X	2	2	2	3	4	5	6
P	3	2	3	3	4	5	6
O	4	3	2	3	4	5	5
N	5	4	3	3	4	4	5
E	6	5	4	4	4	5	5
N	7	6	5	5	5	4	5

Answer. The backward path is seen below:



		P	O	L	Y	N	O
	0	1	2	3	4	5	6
E	1	1	2	3	4	5	6
X	2	2	2	3	4	5	6
P	3	2	3	3	4	5	6
O	4	3	2	3	4	5	5
N	5	4	3	3	4	4	5
E	6	5	4	4	4	5	5
N	7	6	5	5	5	4	5

3

The minimal-cost set of edit operations is obtained through the following steps: To extract the minimum-cost alignments, we examine the sequences of choices made to arrive at $S(7,6) = 5$. Specifically, find the minimum cost path backwards through the DAG to S(0,0). Since We know the table was filled in based on the following recurrence relation, we can extract the minimal cost set of edit operations.

$$\text{cost}(i,j) = \min \begin{cases} \text{cost}(i-2,j-2) + c(\mathit{swap}) \\ \text{cost}(i-1,j-1) + c(\mathit{sub}) \\ \text{cost}(i-1,j) + c(\mathit{indel}) \\ \text{cost}(i,j-1) + c(\mathit{indel}) \end{cases}$$

- A down move represents an *insertion* operation, as it corresponds to the case when $cost(i,j) = cost(i-1,j)+1$ or $cost(i,j) = cost(i,j-1)+1$
- A left move represents a *deletion* operation, as it is when $cost(i,j) = cost(i-1,j)+1$ or $cost(i,j) = cost(i,j-1)+1$
- A single diagonal move is a *substitution*, which is when $cost(i,j) = cost(i-1,j-1)+1$
- Double diagonal moves are a *swap* as it corresponds to $cost(i,j) = cost(i-2,j-2)+2$

The below figure shows this path and the corresponding alignment is

```
- - POLYNO
EXPONEN -
i i | | s s | d
```

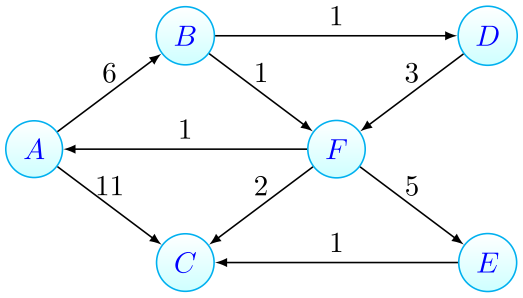
for a total cost of 3 *indels* and 2 *subs*, or 5 overall.

$\square$

3 Standard 23- Dynamic Programming: Bellman-Ford Algorithm

Problem 3. Consider the weighted directed graph $G(V, E, w)$ pictured below. Work through the Bellman-Ford algorithm with the destination vertex C.

- Clearly specify the cost $d(v)$ of the current best path from each node $v \in V$ to C as well as the corresponding successor node at each iteration/pass. You may want to make a table to store the costs and successors.
- Give the shortest path tree, i.e., the union of all the shortest paths to C from all other vertices. For your convenience, you may want to modify the "latex code" for the given graph to draw the shortest path tree.



Answer. Apply Bellman-Ford to the graph:
Let $c(u, v)$ denote cost of the edge from vertex u to vertex v. Let $D(v)$ denote the cost of the current path from vertex v to the destination vertex C.

1. 1st iteration: begin by examining source vertex C incoming edges.
   - Poll edge (A,C): $D(A) = \infty > D(C) + c(A, C) = 0 + 11$ so set $D(A) = 11$ and $successor[A] = C$
   - Poll edge (F,C): $D(F) = \infty > D(C) + c(F, C) = 0 + 2$ so set $D(F) = 2$ and $successor[F] = C$
   - Poll edge (E,C): $D(E) = \infty > D(C) + c(E, C) = 0 + 1$ so set $D(E) = 1$ and $successor[E] = C$
2. 2nd iteration:examine the incoming edges for each of the updated vertices, which were A, F, C
   - Poll edge (F,A): $D(F) = 2 < D(A) + c(F, A) = 11 + 1 = 12$ so no change
   - Poll edge (F,E): $D(F) = 2 < D(E) + c(F, E) = 5 + 1$ so no change
   - Poll edge (D,F): $D(D) = \infty > D(F) + c(F, F) = 2 + 3$ so set $D(D) = 5$ and $successor[D] = F$
   - Poll edge (B,F): $D(B) = \infty > D(F) + c(E, C) = 2 + 1$ so set $D(B) = 3$ and $successor[B] = F$
3. 3rd iteration:examine the incoming edges for each of the updated vertices, which were D,B
   - Poll edge (A,B): $D(A) = 11 > D(B) + c(A, B) = 3 + 6 = 9$ so set $D(A) = 9$ and $successor[A] = B$
   - Poll edge (B,D): $D(B) = 3 < D(D) + c(B, D) = 5 + 1 = 6$ so no change
4. 4th iteration:examine the incoming edges for each of the updated vertices, which was A
   - Poll edge (F,A): $D(F) = 2 < D(A) + c(F, A) = 11 + 1 = 12$ so no change
5. 5th iteration:no updates made

At the end of the algorithm the paths are the following

- A-B-F-C with cost of 9
- B-F-C with cost 3
- F-C with cost 2

- E-C with cost 1
- D-F-C with cost 5

See the shortest path tree below

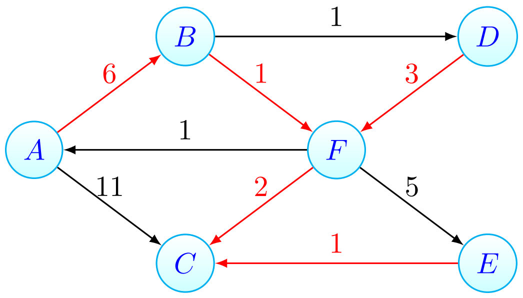


Table 1: Initialization

Node	Cost	Successor
A	∞	null
B	∞	null
C	0	C
D	∞	null
E	∞	null
F	∞	null

Table 2: 1st Iteration

Node	Cost	Successor
A	11	C
B	∞	null
C	0	C
D	∞	null
E	1	C
F	2	C

Table 3: 2nd Iteration

Node	Cost	Successor
A	11	C
B	3	F
C	0	C
D	5	F
E	1	C
F	2	C

$\square$

Table 4: 3rd Iteration

Node	Cost	Successor
A	9	B
B	3	F
C	0	C
D	5	F
E	1	C
F	2	C

Table 5: 4th Iteration

Node	Cost	Successor
A	9	B
B	3	F
C	0	C
D	5	F
E	1	C
F	2	C

Table 6: 5th Iteration

Node	Cost	Successor
A	9	B
B	3	F
C	0	C
D	5	F
E	1	C
F	2	C