CSCI 3104 Spring 2022
Instructors: Profs. Chen and Layer

Problem Set 8

Due Date .... March 29
Name .... **Chengming Li**
Student ID **109251991**
Collaborators .... **N/A**

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.

$$OPT(i,w) = \begin{cases} 0 & i = 0 \\ OPT(i-1,w) & w_i > w \\ max\{OPT(i-1,w), v_i + OPT(i-1, w-w_i)\} & otherwise \end{cases}$$

- For item 6:

  OPT(6,11) = 43, we choose third condition because $w_6 = 8 < 11$. And in this condition,we have $max\{OPT(5,11) = 43, 34 + OPT(5,3) = 34 + 8 = 42\}$. So **we choose OPT(5,11) = 43**
  OPT(5,11) = 43, we choose third condition because $w_5 = 7 < 11$. And in this condition,we have $max\{OPT(4,11) = 43, 24 + OPT(4,4) = 29 + 8 = 37\}$. So **we choose OPT(4,11) = 43**
  OPT(4,11) = 43, we choose third condition because $w_4 = 6 < 11$. And in this condition,we have $max\{OPT(3,11) = 27, 19 + OPT(3,5) = 24 + 19 = 43\}$. So **we choose** $w_4 + OPT(3,5) = 24 + 19$.
  OPT(3,5) = 19, we choose third condition because $w_3 = 5 = 5$. And in this condition,we have $max\{OPT(2,5) = 8, 19 + OPT(2,0) = 19 + 0 = 19\}$. So **we choose** $w_3 + OPT(2,0) = 24 + 19$.
  OPT(2,0) = 0, we choose second condition because $w_2 = 2 > 0$. And in this condition,we have OPT(1,0) = 0. So **we choose OPT(1,0) = 0**.
  OPT(1,0) = 0, we choose second condition because $w_1 = 1 > 0$. And in this condition,we have OPT(0,0) = 0. So **we choose OPT(0,0) = 0**.
  The path is OPT(6,11) -> OPT(5,11) -> OPT(4,11) -> OPT(3,5) -> OPT(2,0) -> OPT(1,0) -> OPT(0,0)

- For item 5:
  OPT(5,11) = 43, we choose third condition because $w_5 = 7 < 11$. And in this condition,we have $max\{OPT(4,11) = 43, 24 + OPT(4,4) = 29 + 8 = 37\}$. So **we choose OPT(4,11) = 43**
  OPT(4,11) = 43, we choose third condition because $w_4 = 6 < 11$. And in this condition,we have $max\{OPT(3,11) = 27, 19 + OPT(3,5) = 24 + 19 = 43\}$. So **we choose** $w_4 + OPT(3,5) = 24 + 19$.
  OPT(3,5) = 19, we choose third condition because $w_3 = 5 = 5$. And in this condition,we have $max\{OPT(2,5) = 8, 19 + OPT(2,0) = 19 + 0 = 19\}$. So **we choose** $w_3 + OPT(2,0) = 24 + 19$.

OPT(2,0) = 0, we choose second condition because $w_2 = 2 > 0$. And in this condition,we have OPT(1,0) = 0. So **we choose OPT(1,0) = 0**.
OPT(1,0) = 0, we choose second condition because $w_1 = 1 > 0$. And in this condition,we have OPT(0,0) = 0. So **we choose OPT(0,0) = 0**.
The path is OPT(5,11) -> OPT(4,11) -> OPT(3,5) -> OPT(2,0) -> OPT(1,0) -> OPT(0,0)

- For item 4:
  OPT(4,11) = 43, we choose third condition because $w_4 = 6 < 11$. And in this condition,we have $max\{OPT(3,11) = 27, 19 + OPT(3,5) = 24 + 19 = 43\}$. So **we choose** $w_4 + OPT(3,5) = 24 + 19$.
  OPT(3,5) = 19, we choose third condition because $w_3 = 5 = 5$. And in this condition,we have $max\{OPT(2,5) = 8, 19 + OPT(2,0) = 19 + 0 = 19\}$. So **we choose** $w_3 + OPT(2,0) = 24 + 19$.
  OPT(2,0) = 0, we choose second condition because $w_2 = 2 > 0$. And in this condition,we have OPT(1,0) = 0. So **we choose OPT(1,0) = 0**.
  OPT(1,0) = 0, we choose second condition because $w_1 = 1 > 0$. And in this condition,we have OPT(0,0) = 0. So **we choose OPT(0,0) = 0**.
  The path is OPT(4,11) -> OPT(3,5) -> OPT(2,0) -> OPT(1,0) -> OPT(0,0)

- For item 3:
  OPT(3,8) = 27, we choose third condition because $w_3 = 5 < 8$. And in this condition,we have $max\{OPT(2,8) = 27, 19 + OPT(2,3) = 19 + 8 = 27\}$. So **we choose** $w_3 + OPT(2,3) = 19 + 8$.
  OPT(2,3) = 8, we choose third condition because $w_2 = 2 < 3$. And in this condition,we have $max\{OPT(1,3) = 2, 6 + OPT(1,1) = 6 + 2 = 8\}$. So **we choose** $w_2 + OPT(1,1) = 6 + 2$.
  OPT(1,1) = 2, we choose third condition because $w_1 = 1 = 1$. And in this condition,we have $max\{OPT(0,1) = 0, 2 + OPT(0,0) = 2 + 0 = 2\}$. So **we choose 2+OPT(0,0)= 2+0**.
  The path is OPT(3,8) -> OPT(2,3) -> OPT(1,1) -> OPT(0,0)

- For item 2:
  OPT(2,3) = 8, we choose third condition because $w_2 = 2 < 3$. And in this condition,we have $max\{OPT(1,3) = 2, 6 + OPT(1,1) = 6 + 2 = 8\}$. So **we choose** $w_2 + OPT(1,1) = 6 + 2$.
  OPT(1,1) = 2, we choose third condition because $w_1 = 1 = 1$. And in this condition,we have $max\{OPT(0,1) = 0, 2 + OPT(0,0) = 2 + 0 = 2\}$. So **we choose 2+OPT(0,0)= 2+0**.
  The path is OPT(2,3) -> OPT(1,1) -> OPT(0,0)

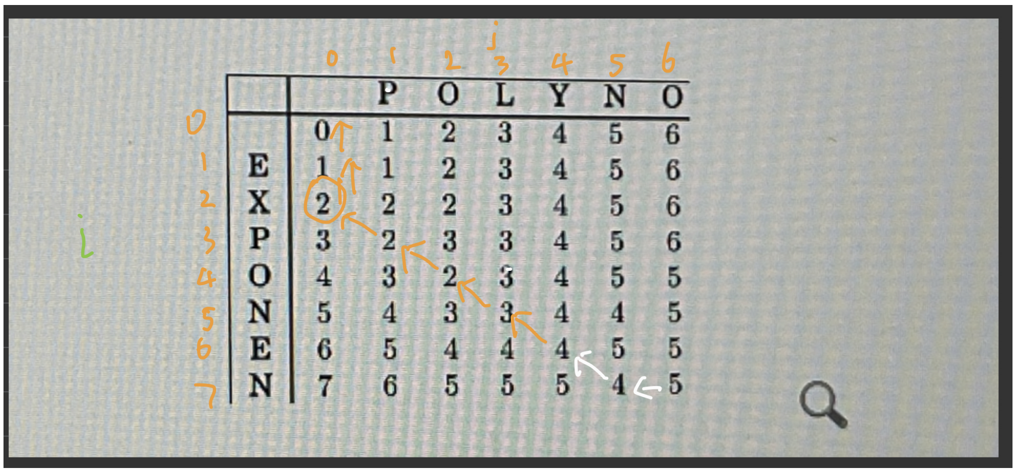
- For item 1:
  OPT(1,1) = 2, we choose third condition because $w_1 = 1 = 1$. And in this condition,we have $max\{OPT(0,1) = 0, 2 + OPT(0,0) = 2 + 0 = 2\}$. So **we choose 2+OPT(0,0)= 2+0**.
  The path is OPT(1,1) -> OPT(0,0)

$\square$

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.



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

$$cost(i,j) = min \begin{cases} cost(i-1, j-1) + c(sub) \\ cost(i-1, j) + c(indel) \\ cost(i, j-1) + c(indel) \\ cost(i-1, j-1) & if(x_i = y_j) \end{cases}$$

The path is $cost(7,6) -> cost(7,5) -> cost(6,4) -> cost(5,3) -> cost(4,2) -> cost(3,1) -> cost(2,0) -> cost(1,0) -> cost(0,0)$

cost(7,6) = cost(7,5)+1(indel) = 5 because it is the minimum cost and $x_i \neq y_j$

$$cost(7,6) = min \begin{cases} cost(6,5) + c(sub) = 6 \\ cost(6,6) + c(indel) = 6 \\ cost(7,5) + c(indel) = 5 \\ cost(6,5) = 5 & if(x_i = y_j) \end{cases}$$

cost(7,5) = cost(6,4)+0(sub) = 4 because it is the minimum cost and$x_i = y_j$

$$cost(7,5) = min \begin{cases} cost(6,4) + c(sub) = 5 \\ cost(6,5) + c(indel) = 6 \\ cost(7,4) + c(indel) = 6 \\ cost(6,4) = 4 & if(x_i = y_j) \end{cases}$$

cost(6,4) = cost(5,3)+1(sub) = 4 because it is the minimum cost and$x_i \neq y_j$

$$cost(6,4) = min \begin{cases} cost(5,3) + c(sub) = 4 \\ cost(5,4) + c(indel) = 5 \\ cost(6,3) + c(indel) = 5 \\ cost(5,3) = 3 & if(x_i = y_j) \end{cases}$$

cost(5,3) = cost(4,2)+1(sub) = 3 because it is the minimum cost and$x_i \neq y_j$

$$cost(5,3) = min \begin{cases} cost(4,2) + c(sub) = 3 \\ cost(4,3) + c(indel) = 4 \\ cost(5,2) + c(indel) = 4 \\ cost(4,2) = 2 & if(x_i = y_j) \end{cases}$$

cost(4,2) = cost(3,1)+0(sub) = 2 because it is the minimum cost and$x_i = y_j$

$$cost(4,2) = min \begin{cases} cost(3,1) + c(sub) = 3 \\ cost(3,2) + c(indel) = 4 \\ cost(4,1) + c(indel) = 4 \\ cost(3,1) = 2 & if(x_i = y_j) \end{cases}$$

cost(3,1) = cost(2,0)+0(sub) = 2 because it is the minimum cost and$x_i = y_j$

$$cost(3,1) = min \begin{cases} cost(2,0) + c(sub) = 3 \\ cost(2,1) + c(indel) = 3 \\ cost(3,0) + c(indel) = 4 \\ cost(2,0) = 2 & if(x_i = y_j) \end{cases}$$

cost(2,0) = cost(1,0)+1(indel) = 2 because it is the minimum cost and$x_i \neq y_j$

$$cost(2,0) = min \begin{cases} cost(1,-1) + c(sub) = NULL \\ cost(1,0) + c(indel) = 2 \\ cost(2,-1) + c(indel) = NULL \\ cost(1,-1) = NULL & if(x_i = y_j) \end{cases}$$

$\square$

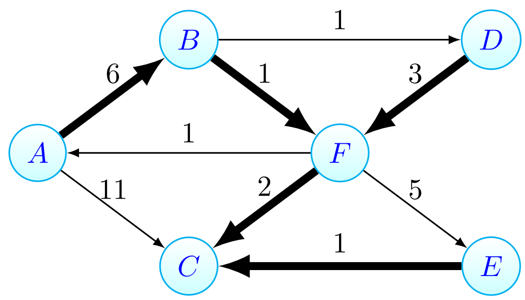
cost(1,0) = cost(0,0)+1(indel) = 1 because it is the minimum cost and$x_i \neq y_j$

$$cost(1,0) = min \begin{cases} cost(0,-1) + c(sub) = NULL \\ cost(0,0) + c(indel) = 2 \\ cost(1,-1) + c(indel) = NULL \\ cost(0,-1) = NULL & if(x_i = y_j) \end{cases}$$

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. Initialization:

Vertex	Dist from C	Prev-Vertex
A	∞	NULL
B	∞	NULL
C	0	C
D	∞	NULL
E	∞	NULL
F	∞	NULL

First iteration:

Vertex	Dist from C	Prev-Vertex
A	11	C
B	∞	NULL
C	0	C
D	∞	NULL
E	1	C
F	2	C

Second iteration:

Vertex	Dist from C	Prev-Vertex
A	11	C
B	3	F
C	0	C
D	5	F
E	1	C
F	2	C

Third iteration:

Vertex	Dist from C	Prev-Vertex
A	9	B
B	3	F
C	0	C
D	5	F
E	1	C
F	2	C

Fourth iteration:

Vertex	Dist from C	Prev-Vertex
A	9	B
B	3	F
C	0	C
D	5	F
E	1	C
F	2	C

Iteration stops because no values change anymore. □