Name: Luna McBride
ID: 107607144

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 10a (25 points) **Fall 2019, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solution:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to Latex.
- You should submit your work through **Gradescope** only.
- If you don't have an account on it, sign up for one using your CU email. You should have gotten an email to sign up. If your name based CU email doesn't work, try the identikey@colorado.edu version.
- Gradescope will only accept **.pdf** files (except for code files that should be submitted separately on Gradescope if a problem set has them) and **try to fit your work in the box provided**.
- You cannot submit a pdf which has less pages than what we provided you as Gradescope won't allow it.

Name: Luna McBride
ID: 107607144

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 10a (25 points) **Fall 2019, CU-Boulder**

1. (11 pts) Solve the following based on the given intervals.

Event Index	Interval	Value
1	$[1, 10]$	10
2	$[11, 20]$	5
3	$[21, 30]$	15
4	$[9, 12]$	7
5	$[19, 22]$	8

(a) (2.5 pts) For the given set of requests with a start and finish time and a value, what is $p(j)$ for each request j?

Solution.
P(1)=0, P(2)=1, P(3)=2, P(4)=0, P(5)=1

(b) (1.5 pts) Assume you are given a greedy algorithm for the weighted interval scheduling problem that selects the requests in descending order of their value, i.e. highest valued requests are selected first. Provide an example set of intervals where this approach won't produce the optimal solution.

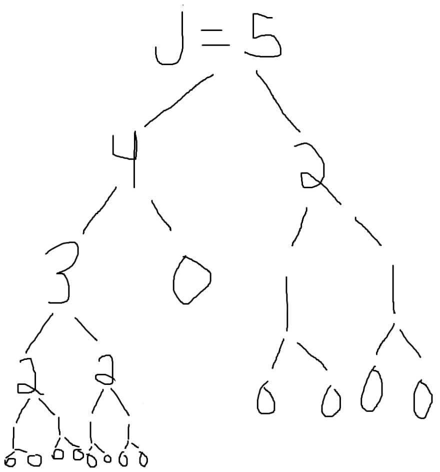
Event Index	Interval	Value
1	$[1, 10]$	13
2	$[11, 20]$	17
3	$[1, 20]$	29

Solution.
The greedy solution would pick the one of value 29 because that is the biggest, then block off the other intervals. What the algorithm doesn't know is that the other two, taking the same amount of space, but having an overall value of 30.

(c) (2 pts) In the following recursive formula for the weighted interval scheduling problem in Question 1, how many recursive calls are there to *WISNoMem*? Show the recursion tree to support your answer.

```
WISNoMem(j) {
    if (j == 0)
        return 0
    else
        return max(WISNoMem(j-1), v[j] + WISNoMem(p[j]))
}
```

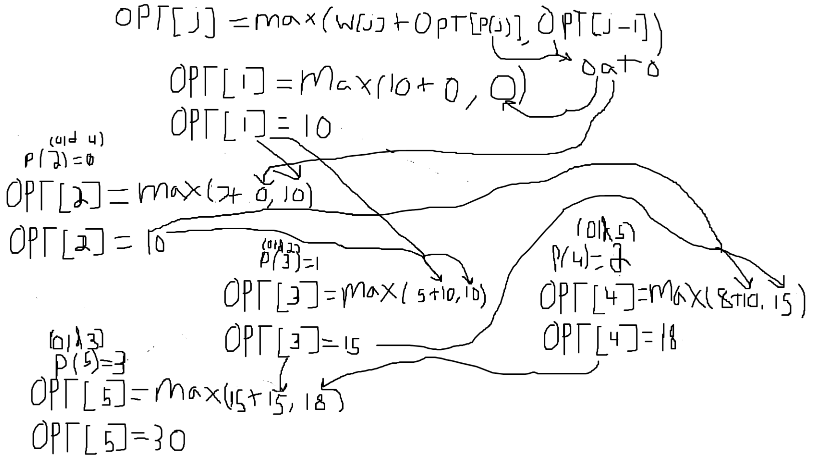
Solution.



Number of calls: 24

(d) (3 pts) Using the bottom-up iterative algorithm presented in lecture, show the $OPT(j)$ values for all requests. Also draw the arrows to show the previous subproblem you used to fill an entry.

Solution.



OPT[j] = max(w[j] + OPT[P(j)], OPT[j-1])

0 or 0

OPT[1] = Max(10 + 0, 0)

OPT[1] = 10

(012 4)
p(2) = 0

OPT[2] = max(7 + 0, 10)

OPT[2] = 10

(0123)
P(3) = 1

OPT[3] = max(5 + 10, 10)

OPT[3] = 15

(0125)
P(4) = 2

OPT[4] = Max(8 + 10, 15)

OPT[4] = 18

(0123)
P(5) = 3

OPT[5] = max(15 + 15, 18)

OPT[5] = 30

(e) (2 pts) Show which requests are selected in the optimal solution and the value of the solution. Provide a 2-3 sentence explanation of how you retrieved the solution using the arrows you drew.

Solution.

The optimal requests are 1,3, and 5 (in terms of the table and not the sorted by finish, requests 1,2, and 3). The OPT array shows the most optimal up to the point j. When the OPT is updated with the same value, that means the value did not need to be considered for the optimal (right side of the max). When the j is considered, the weight is added to P(j) and not the previous, cutting out ones that may have been optimal before, but are no longer because of the addition of the j value. So jumping back through the arrows, w[5]+OPT[3], w[3]+OPT[1],w[1]+OPT[0], which is w[5]+w[3]+w[1] when merging the OPTs.

Name: Luna McBride
ID: 107607144

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 10a (25 points) **Fall 2019, CU-Boulder**

2. (14 pts) Solve the following sequence alignment questions.

   (a) (2 pts) Consider the sub-problem represented by `i` and `j` i.e. `cost[i][j]` which represents the minimum cost of aligning sub-sequences in the source and target respectively.
   What are the smaller sub-problems needed to solve this sub-problem?
   What operation do each of the smaller sub-problems represent **with respect to the source sequence**?

   Solution.
   In terms of `cost[i][j]`, there are three subproblems to look at to decide the alignment: `cost[i-1][j]`, `cost[i][j-1]`, and `cost[i-1][j-1]`. Movement between `cost[i][j]` and `cost[i-1][j]` (left-right) means an insertion into the source, movement between `cost[i][j]` and `cost[i][j-1]` (up-down) indicates a deletion from the source to create the target, and movement between `cost[i][j]` and `cost[i-1][j-1]` (diagonal) indicates a substitution of a character in that location for what goes into the target.

   It is good to note, however, that such ops are not used to go to a cell of higher cost. In addition, if the next cell is the exact same cost as the previous, no matter the direction, this represents a 'no-op', where nothing is actually done and the cost to do so is 0.

(b) (4 pts) Consider the source sequence `s = BEAR`, the target sequence `t = BARE`, and the given cost table. **Draw the arrows that represent the optimal alignment** of `s` and `t`. You don't need to draw arrows which are not part of this optimal alignment.
(Note that you have to do the back tracing starting from the cell representing the solution to the full problem and traverse to the base case. You can infer the parent from observing the relevant sub-problems.)

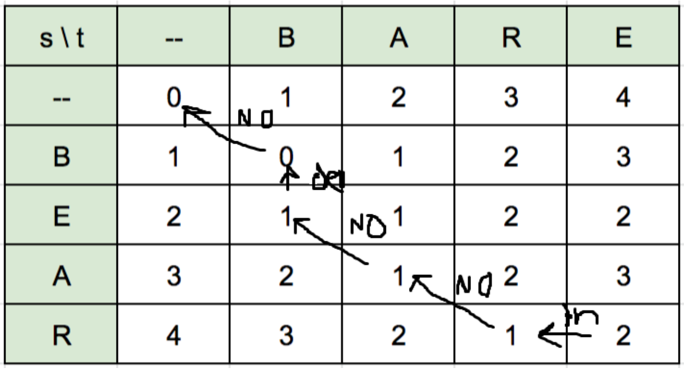
s\t	--	B	A	R	E
--	0	1	2	3	4
B	1	0	1	2	3
E	2	1	1	2	2
A	3	2	1	2	3
R	4	3	2	1	2

Also, draw the alignment corresponding to the arrows for the given sequences `s = BEAR` and `t = BARE` and **give the set of optimal operations** on the `s = BEAR` which will transform it to `t = BARE` (using the cost table with relevant arrows you drew earlier).
An example drawn alignment and set of operations for `s = STEP` and `t = APE` look like following. (Show the 'no-op' using a colon (:) to differentiate with a regular sub.)

Drawn alignment = **Ops =** `['sub', 'sub', 'no-op', 'delete']`

S T E P
| | : |
A P E _

Solution.



s \ t	--	B	A	R	E
--	0	1	2	3	4
B	1	0	1	2	3
E	2	1	1	2	2
A	3	2	1	2	3
R	4	3	2	1	2

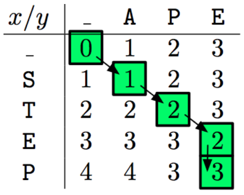
NO
del
NO
NO
in

Ops = [`'no-op'`, `'delete'`, `'no-op'`, `'no-op'`, `'insert'`]

B E A R _
: | : : |
B _ A R E

(c) (8 pts) Consider the source sequence `s = PLAIN`, the target sequence `t = PLANE`, `cost(insert) = 2`, `cost(delete) = 2`, and `cost(sub) = 2`. Your goal is to align (or transform) the source to the target with minimum cost. Align the source sequence to the target sequence using a bottom-up tabular approach like the one shown on Q2 part b.

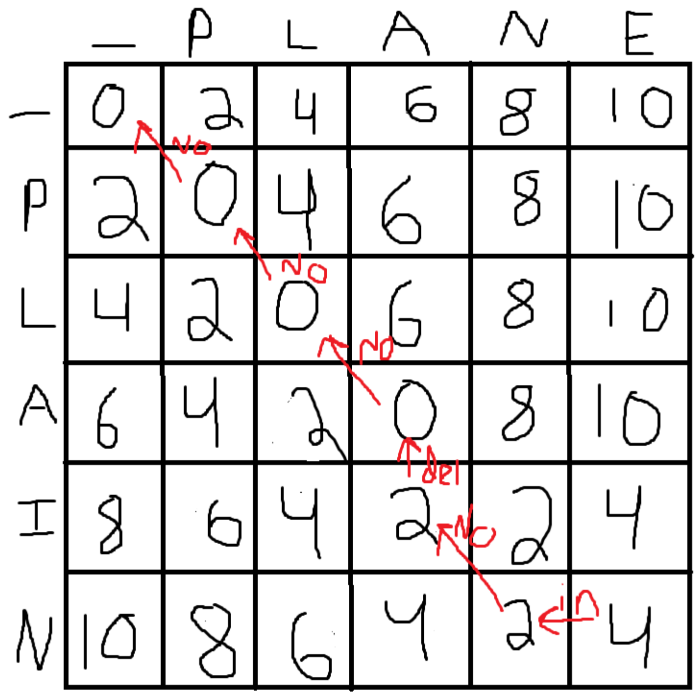
- Draw the DP table showing the cost of alignment.
- Show the arrows representing the optimal alignment. Note that there are more than one optimal alignment. Show **two** of the optimal alignments in separate tables using arrows.
  One such example for different sequences and costs -



x/y	_	A	P	E
_	0	1	2	3
S	1	1	2	3
T	2	2	2	3
E	3	3	3	2
P	4	4	3	3

- Draw the alignments corresponding to the **two** optimal alignments like you drew in the last part of Q2 part b.
- Provide the cost of the optimal alignment.

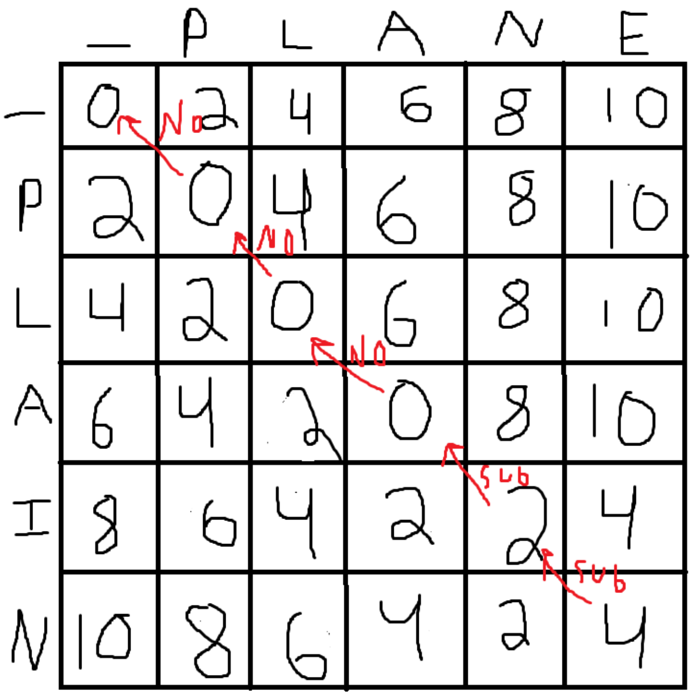
Solution.


NO
NO
NO
del
NO
in

	_	P	L	A	N	E
_	0	2	4	6	8	10
P	2	0	4	6	8	10
L	4	2	0	6	8	10
A	6	4	2	0	8	10
I	8	6	4	2	2	4
N	10	8	6	4	2	4

```
P L A I N _
: :  : | : |
P L A _ N E
```

Ops = [’no-op’, ’no-op’, ’no-op’, ’delete’, ’no-op’, ’insert’]
Cost: 4 (Note: all normal cost multiplied by 2 because all the costs are 2)


NO
NO
NO
sub
sub

	_	P	L	A	N	E
_	0	2	4	6	8	10
P	2	0	4	6	8	10
L	4	2	0	6	8	10
A	6	4	2	0	8	10
I	8	6	4	2	2	4
N	10	8	6	4	2	4

P L A I N
: : : | |
P L A N E

Ops = [’no-op’, ’no-op’, ’no-op’, ’sub’, ’sub’]
Cost=4