**­­CSE 3318 – ExamQuiz 5** NAME: Nebojsa Kilibarda

Total points: 100 Topics: Greedy and Dynamic Programming, leetcode. Problems: Job Scheduling, 0/1 Knapsack, Edit Distance, LCS, LIS.

**Submit ONLY ONE pdf (or docx) document. Fill in electronically (NOT handwritten) your name and answers** for problems P1 to P5. Use **a different color** for the answers or highlight them. It makes the grading easier. It makes the grad. **Insert an image of the required screen shot(s) for P6**. Save the document in pdf or docx format (with any name you want). **Submit only the pdf/docx.**

**P1 (JS1)** (8 pts) The left table below is the preprocessed version of a Job Scheduling problem: the jobs have already been preprocessed as needed in order to apply dynamic programming.

(6 pts) Fill in the table as done in class. Put ***Y*** if a job was picked and ***N*** if it was not picked in the ***local*** choice.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | value | p(i) |  | idx | Y/N | sol(i) | Work (as done in class) |
| 0 | 0 | -1 |  | 0 | N | 0 |  |
| 1 | 4 | 0 |  | 1 | Y | 4 | Max {0, 4+sol (0)} = max (0, 4) = 4 |
| 2 | 3 | 1 |  | 2 | Y | 7 | Max (4, 3+sol (1)} = max (4, 7) = 7 |
| 3 | 6 | 0 |  | 3 | N | 7 | Max {7, 6+sol (0)} = max (7, 6) = 7 |
| 4 | 5 | 1 |  | 4 | Y | 9 | Max {7, 5+sol (1)} = max (7, 9) = 9 |
| 5 | 1 | 2 |  | 5 | N | 9 | Max {9, 1+sol (2)} max (9, 8) = 9 |
| 6 | 3 | 1 |  | 6 | N | 9 | Max {9, 3+sol (1)} = max (9, 7) = 9 |
| 7 | 3 | 5 |  | 7 | Y | 12 | Max {9, 3+sol (5)} = max (9, 12) = 12 |

(2pts) Total money made: \_12$\_ IDs of jobs picked for the final problem: \_7,4,1\_\_\_\_\_

**P2 (JS2)** (4pts) Given the jobs below, preprocess them as needed to be able to run dynamic programming. You only need to do the preprocessing part, not the dynamic programming part.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (start,end,value) |  | ID | value | p(i) |
| (10,11,1) |  | 1 | 6 | 0 |
| (4,8,2) |  | 2 | 2 | 0 |
| (13,17,5) |  | 3 | 3 | 1 |
| (6,9,3) |  | 4 | 1 | 3 |
| (2,5,6) |  | 5 | 7 | 2 |
| (8,14,7) |  | 6 | 5 | 4 |

**P3 (ED,LCS)**  (12 pts) Fill in the tables below to compute the Edit Distance (ED) and the Longest Common Subsequence (LCS) between STORM and TRAY.

1. (5 pts) Find the Edit Distance (ED). (**4** pts) Fill in the table. Record only the value. The choice/direction is not needed.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | S | T | O | R | M |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 |
|  | 0 | 0 | 1 | 2 | 3 | 4 | 5 |
| T | 1 | 1 | 1 | 1 | 2 | 3 | 4 |
| R | 2 | 2 | 2 | 2 | 2 | 2 | 3 |
| A | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Y | 4 | 4 | 4 | 4 | 4 | 4 | 4 |

(**1** pt) Final answer for ED: \_\_\_4\_\_\_\_\_\_\_

1. (7 pts) Find the Longest Common Subsequence.

(4 pts) Fill in the table. Record the direction as well using the symbols: “\” for diagonal, “-“ for left, “|” for up. E.g. “0-” in a cell means the value is 0 and the arrow is left.

**Whenever there is a tie between diagonal and anything else, choose diagonal. Use the algorithm for LCS\_length(X,Y) from page 26 of the slides. Whenever there is a tie between left and up, chose up. Penalty for the wrong choice in case of a tie: 1 point.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | S | T | O | R | M |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 |
|  | 0 | 0 | 0- | 0- | 0- | 0- | 0- |
| T | 1 | 0| | 0\ | 1\ | 1- | 1- | 1- |
| R | 2 | 0| | 0\ | 1| | 1\ | 2\ | 2- |
| A | 3 | 0| | 0\ | 1| | 1\ | 2| | 2\ |
| Y | 4 | 0| | 0\ | 1| | 1\ | 2| | 2\ |

(2 pts) LCS final answer: Length: \_2\_ Subsequence: \_T R\_\_\_\_

( 1pt) Highlight the cells that you visit when recovering the longest common subsequence

**P4 (LIS)**  (14 pts) Find the Longest Increasing Subsequence in the array {5,4,7,4,0,4,1,6} with and without duplicates, using LCS. Fill in below the second array to given to LCS and also the final answer to the problem, the actual longest increasing subsequence.

1. (4pts) LISunique( {5,4,7,4,0,4,1,6}) = LCS ({5,4,7,4,0,4,1,6}, {0,1,4,5,6,7}) = {0,4,6}
2. (4pts) LISduplicates({5,4,7,4,0,4,1,6}) = LCS ({5,4,7,4,0,4,1,6}, {0,1,4,4,4,5,6,7}) ={4,4,4,6}
3. (6pts) Fill in the table below to compute the Longest Increasing Subsequence in the array {5,4,7,4,0,4,1,6} when duplicates are allowed (LISduplicates({5,4,7,4,0,4,1,6}) in part b). The numbers shown in the table are the indexes for rows and columns. You should fill in the data corresponding for each row and column and the inner part of the table. Note that the table may have more rows and columns than you need. You should use as many as you need, and ignore the bottom row(s) and rightmost column(s) that you do not need. In the table record the choice/direction using: “\” for diagonal, “-“for left, “|” for up. E.g. “0-” in a cell means the value is 0 and the arrow is left. Use the algorithm from class. Whenever there is a tie between left and up, chose up.

The class code uses diagonal only when a match is found.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | 5 | 4 | 7 | 4 | 0 | 4 | 1 | 6 |  |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| 0 | 1 | 0 | 0| | 0| | 0| | 0| | 1\ | 1- | 1- | 1- |  |
| 1 | 2 | 0 | 0| | 0| | 0| | 0| | 1| | 1| | 2\ | 2- |  |
| 4 | 3 | 0 | 0| | 1\ | 1- | 1\ | 1| | 2\ | 2| | 2| |  |
| 4 | 4 | 0 | 0| | 1\ | 1| | 2\ | 2- | 2\ | 2| | 2| |  |
| 4 | 5 | 0 | 0| | 1\ | 1| | 2\ | 2| | 3\ | 3- | 3- |  |
| 5 | 6 | 0 | 1\ | 1| | 1| | 2| | 2| | 3| | 3| | 3| |  |
| 6 | 7 | 0 | 1| | 1| | 1| | 2| | 2| | 3| | 3| | 4\ |  |
| 7 | 8 | 0 | 1| | 1| | 2\ | 2| | 2| | 3| | 3| | 4| |  |
|  | 9 |  |  |  |  |  |  |  |  |  |  |

Exam continues on next page**P5 (KP)** (18 pts) The max capacity is 14, and there is ONLY ONE of each item shown in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item: | A | B | C | D |
| Weight: | 3 | 4 | 6 | 7 |
| Value: | 4 | 7 | 10 | 12 |

**a)** (10 pts) Solve this 0/1 Knapsack problem, using Dynamic programing as done in class.

*(4 pts) Fill in the table below*. Assume there is only of each item, and you CANNOT take fractions of an item (**0/1, and NOT fractional** Knapsack). *(3pts) Use a star to show if the current item was used or not in the solution.*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A, 3, 4 | 0 | 0 | 0 | 4\* | 4\* | 4\* | 4\* | 4\* | 4\* | 4\* | 4\* | 4\* | 4\* | 4\* | 4\* |
| B, 4, 7 | 0 | 0 | 0 | 4 | 7\* | 7\* | 7\* | 11\* | 11\* | 11\* | 11\* | 11\* | 11\* | 11\* | 11\* |
| C, 6, 10 | 0 | 0 | 0 | 4 | 7 | 7 | 10\* | 11 | 11 | 14\* | 17\* | 17\* | 17\* | 21\* | 21\* |
| D, 7, 12 | 0 | 0 | 0 | 4 | 7 | 7 | 10 | 12\* | 12\* | 14 | 17 | 19\* | 19\* | 22\* | 23\* |

*Final answer: (1pt) Value: \_\_\_23\_\_ (1pt) Items picked: \_\_\_D, B, A\_\_ (1pt) Highlighted cells visited to recover picked items.*

**b)** (4 pts) What items will a **Greedy** algorithm based on the **ratio value to weight,** choose for max **capacity 14** for the normal **0/1 Knapsack** problem (where you CANNOT take a fraction of an item)?

(2pts) Final answer: Value: \_\_23\_\_\_ Items picked: \_\_\_B, D, A\_\_\_\_\_\_

(2pts) Show your work:

A = 4/3 = 1.333 B = 7/4 = 1.75 C = 10/6 = 1.666 D = 12/7 = 1.714

The order according to ratio: B D C A -> 7(4) + 12(7) + skip C + 4(3) = 23(14)

**c)** (4 pts) What items will a **Greedy** algorithm based on **only value**, choose for max **capacity 14** for the normal **0/1 Knapsack** problem (where you CANNOT take a fraction of an item)?

(2pts) Final answer: Value: \_22\_\_ Items picked: \_\_\_\_\_\_\_D, C\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(2pts) Show your work:

The order according to value (decreasing): D C B A

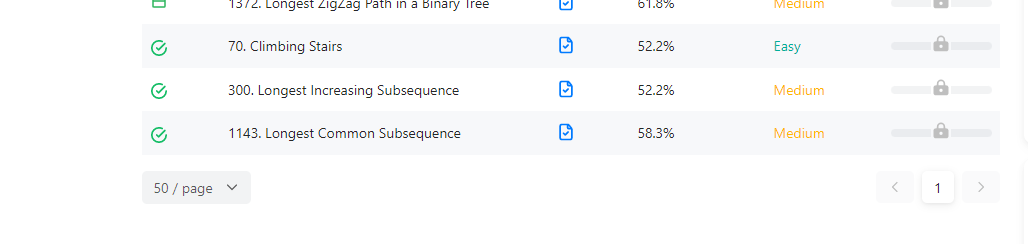
Items picked: 12(7) + 10(6) = 22(13)

**P6** (44 pts) Solve 3 leetcode problems. You are allowed to check the leetcode solutions, but it is better to ask and discuss with us any questions you have about solving these problems. The purpose of this quiz is to learn and understand these problems. Solve:

* [1143. Longest Common Subsequence](https://leetcode.com/problems/longest-common-subsequence/) (15 pts)
* [300. Longest Increasing Subsequence](https://leetcode.com/problems/longest-increasing-subsequence/) (15 pts)
* And 1 more problem from the list ( 14 pts):
  + [64. Minimum Path Sum](https://leetcode.com/problems/minimum-path-sum/)
  + [70. Climbing Stairs](https://leetcode.com/problems/climbing-stairs/)
  + [198. House Robber](https://leetcode.com/problems/house-robber/)
  + [1235. Maximum Profit in Job Scheduling](https://leetcode.com/problems/maximum-profit-in-job-scheduling/)

**Insert in this document one image of your screenshot** **of the ACCEPTED leetcode problems list** with: 1) the SOLVED status (and the check mark next to them) and 2) “Dynamic Programming” tag. If your list of problems is long, mark (circle, underline, or highlight) the problems that go towards P6. If you prefer, in addition to the one screen shot, you can include individual screenshots with the code for each problem.

**Insert here your leetcode screenshot image(s) with proof that you completed these problems:**

****