Algorithms: Homework 3

Li-Yuan Wei

Wednesday 14th June, 2023

Problem 1

Give an $\mathcal{O}(n^2)$-time algorithm to find the longest monotonically increasing subsequence of n numbers.

Solution. Psuedocode for an $\mathcal{O}(n^2)$-time algorithm:

```
LONGEST-INCREASING-SUBSEQUENCE(A, n)
1:  B = MERGE-SORT(A) // B is a sorted copy of A in ascending order
2:  C[n+1, n+1] // create 2-D array C with val, dir fields
3:  for i = 0 to n
4:    for j = 0 to n
5:      C[i][j] = 0.0 // initialize each address, input can be real numbers
6:  for i = 1 to n
7:    for j = 1 to n
8:      if B[i] ≤ A[j]
9:        C[i][j].val = C[i - 1][j - 1].val + 1
10:       C[i][j].dir = ↖
11:     else // B[i] > A[j] no increasing
12:       if C[i - 1][j].val < C[i][j - 1].val
13:         C[i][j].val = C[i][j - 1].val
14:         C[i][j].dir = ←
15:       else
16:         C[i][j].val = C[i - 1][j].val
17:         C[i][j].dir = ↑
18: return C // C[n][n].val is the length of the longest increasing subsequence of n
```

Line 1 has time complexity of merge sort, which is $\mathcal{O}(n \lg n)$. Line 2 has constant time complexity. Line 3-5 has time complexity of $\mathcal{O}(n^2)$, same as line 6-17. Thus, the overall time comlexity is dominated by $\mathcal{O}(n^2)$. ■

Problem 2

Find an optimal solution to the following activity selection problem:

i	1	2	3	4	5	6	7	8	9	10
s_i	1	3	2	3	5	8	7	10	5	11
f_i	3	4	5	6	8	9	11	12	14	15

Solution. The optimal solution is to chose the following events: <1, 2, 5, 6, 8>. ■

Problem 3

Not just any greedy approach to the activity-selection problem produces a maximum-size set of mutually compatible activities. (a) Give an example to show that the approach of selecting the activity of least duration from those that are compatible with previously selected activities does not work. (b) Do the same for the approaches of always selecting the compatible activity that overlaps with the fewest other remaining activities by selecting the compatible remaining activity with the earliest start time.

i	1	2	3	4	5
s_i	1	2	3	4	5
f_i	2	4	5	7	8

Figure 1: select least duration

i	1	2	3	4	5
s_i	1	2	3	4	5
f_i	2	4	5	7	8

Figure 2: select fewest overlaps

Solution. ■

Problem 4

Give a dynamic-programming solution to the 0-1 knapsack problem that runs in $\mathcal{O}(nW)$ time, where n is number of items and W is the maximum weight of items that the thief can put in his knapsack. Note that W, as well as all the weights of the involved items are integers.

Solution. ■

Problem 5

Assume there are only 7 characters A, B, C, D, E, F and G in a document, with occurrences of 6, 4, 8, 3, 2, 1 and 5, respectively. Please calculate the total number of bits of this document after applying the Huffman coding.

C: 8 | 11
A: 6 | 01
G: 5 | 101
B: 4 | 100
D: 3 | 001
E: 2 | 0000
F: 1 | 0001

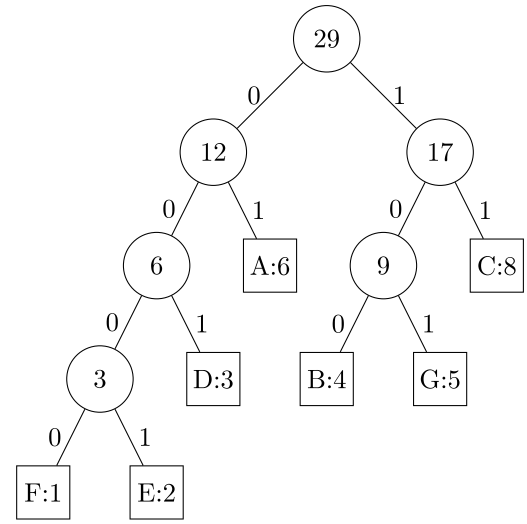


Figure 3: Huffman code tree

Solution. total bits: $1 \cdot 4 + 2 \cdot 4 + 3 \cdot 3 + 4 \cdot 3 + 5 \cdot 3 + 6 \cdot 2 + 8 \cdot 2 = 76$ ■

Problem 6

Represent the graph on the right by adjacency-matrix and adjacency-list.

	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	1	0
3	0	1	0	0	1
4	1	1	0	0	1
5	0	0	1	1	0

Figure 4: adjacency-matrix

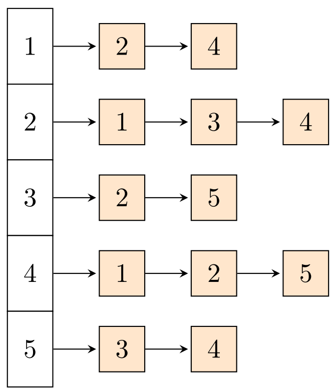
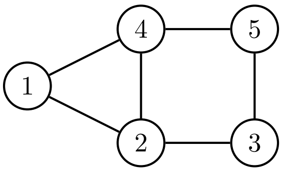


Figure 5: adjacency-list

Solution. ■

Problem 7

Show the d and π values that result from running the breadth-first search on the same graph using vertex 3 as the source.



v	1	2	3	4	5
d	2	1	0	2	1
π	2	3	**NIL**	2	3

Figure 6: Original Graph

Solution. v stands for verticies in the graph. d is the search distance, while π is vertex v's parent node. Vertex 3 is the starting point, thus, it has no parent node. ■

Problem 8

(a) Show how depth-first search works on the graph at the bottom. Assume that the for loop of lines 5—7 of the DFS procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically. Show the discovery and finishing time for each vertex, and show the classification of each edge.
(b) According to (a), show the corresponding topological sort of this graph.

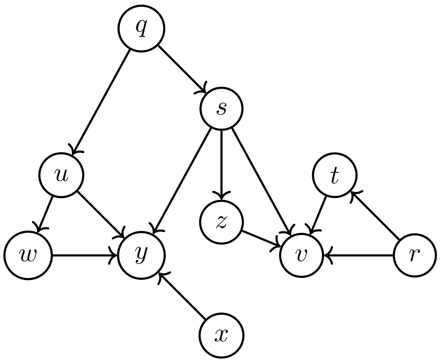


Figure 7: Original Graph

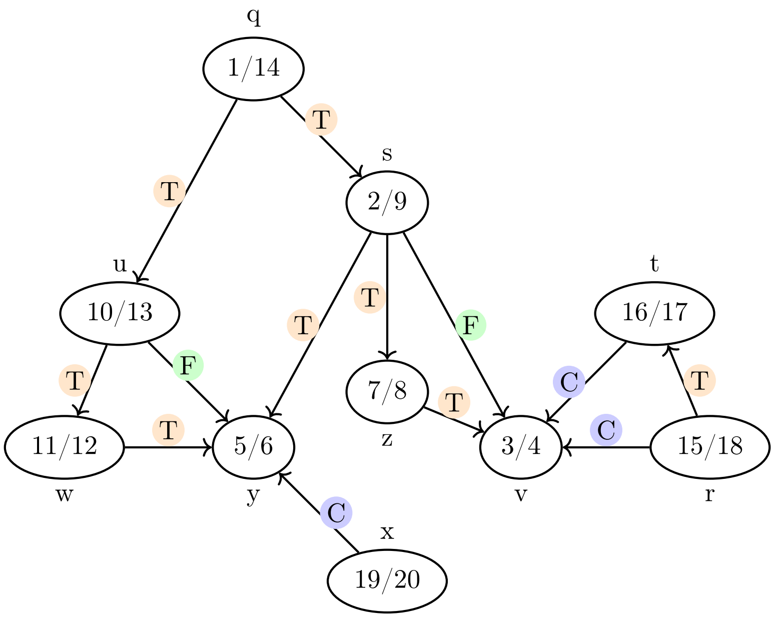


Figure 8: (a)

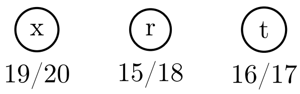


Figure 9: (b)

Solution. ■

Problem 9

Find the strongest components of the following graph:

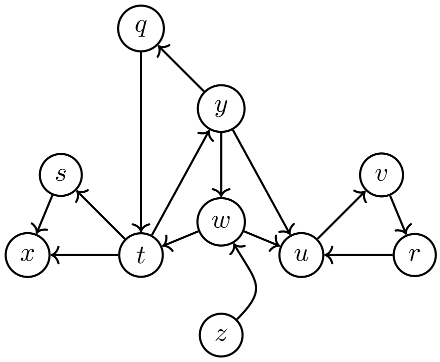


Figure 10: Original Graph

Solution. ■

Problem 10

Find the minimal spanning tree of the following graph, using (a) Kruskal's algorithm, (b) Prim's algorithm (using q as the starting vertex). Please give the selected edge in order during the spanning tree construction process

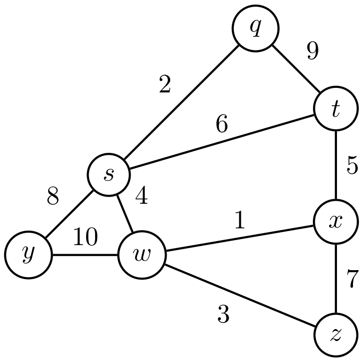


Figure 11: Original Graph

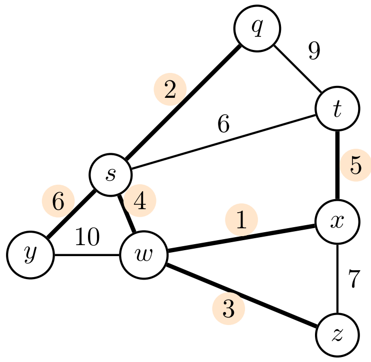


Figure 12: Minimal Spanning Tree: Kruskal's algorithm

Solution. Kruskal's algorithm: first sort all edges in the graph in increasing order, then pick the lowest cost edge every step until every vertex is connected to our tree. The final result is shown in Figure 12. ■