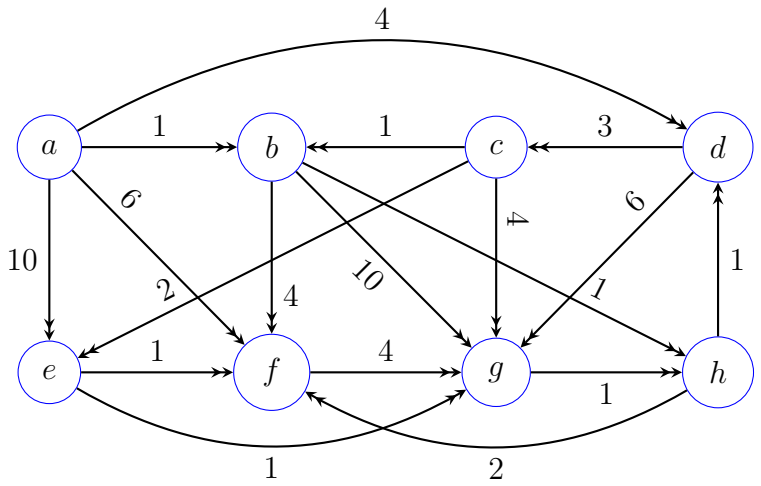
CECS 528 TuTh, LO Assessment 2, Fall 2026, Dr. Ebert

IMPORTANT: For each problem (LO1,LO2,GK,Advanced, All LO Makeups) write your solution using a **SINGLE** and **SEPARATE SHEET OF PAPER (FRONT AND BACK)**. Write your **FIRST NAME**, **ALPHABETIZING NAME** and **PROBLEM CODE** on each sheet.

LO3 Do the following.

a. Consider the following instance of the `Single-Source Distances` problem where vertex a is the source vertex.



Make a table whose rows are the vertices and whose eight columns shows the neighboring distance of each vertex from the source, and the parent of the vertex along the path that creates the neighboring distance. For each of the eight rounds, circle the neighboring distance that wins the round and use the parent information to add the winning vertex to the Dijkstra Distance Tree. "Dash out" the remaining table cells for the winning vertex. (12 pts)

b. For the weighted graph with edges

$$(a, f, 6), (b, e, 5), (c, d, 1), (c, e, 3), (c, f, 4), (e, f, 2),$$

Show how the forest of disjoint-set data structure trees changes when processing each edge in Kruskal's sorted list of edges. When merging two trees, use the convention that the root of the merged tree should be the one having *lower* alphabetical order. For example, if two trees, one with root a, the other with root b, are to be merged, then the merged tree should have root a. Redraw the forest each time an edge is processed. (13 pts)

LO4 Solve the following problems.

a. The dynamic-programming algorithm that solves the `0-1 Knapsack` optimization problem defines a recurrence for the function $p(i, c)$. In words, what does $p(i, c)$ equal? Hint: do *not* write the recurrence (see Part b). (5 pts)

b. Provide the dynamic-programming recurrence for $p(i, c)$. (10 pts)

c. Apply the recurrence from Part b to the instance `0-1 Knapsack` instance having capacity 13 and items with (weights, profits)

$$x_1 = (2, 17), x_2 = (1, 8), x_3 = (3, 22), x_4 = (5, 35), x_5 = (6, 45), x_6 = (4, 30).$$

Show the matrix of subproblem solutions and use it to provide an optimal solution. (10 pts)

GK a. What does it mean for an algorithm to be "greedy"? When is a greedy algorithm considered a heuristic? (12 pts)

b. Define what it means to be an instance of the `Runaway Traveling Salesperson (RTS)` problem and what needs to be optimized? Give an example of a greedy heuristic that would seem appropriate for the `RTS` problem. (13 pts)

ADV Write a *recursive* algorithm using pseudocode and that computes the void function `print_opt_path`, on inputs $d^{(0)}$, $d^{(n)}$, i, and j, and has the side effect of printing the sequence of vertices for a minimum-cost path from vertex i to vertex j, $1 \leq i, j \leq n$, where $d^{(0)}$ is the $n \times n$ cost-adjacency matrix, and $d^{(n)}$ is the final matrix obtained from the Floyd-Warshall algorithm.

```
void print_opt_path(dn,d0,int n, int i, int j);
```

Assume the existence of a `print` function that is capable of printing integers (including proper whitespacing). For example, `print`(j) has the side effect of printing the value assumed by variable j and surrounding it whitespace. Hint: make use of optimal subpaths. (25 pts)

LO MakeUps: Pass/No Pass

LO1 a. Use the Master Theorem to determine the growth of $T(n)$ if it satisfies the recurrence $T(n) = 8T(n/2) + n^2 \log n$. Defend your answer.

b. Use the Substitution method to establish that, if $T(n) = 2T(n/2) + n \log n$, then $T(n) = \mathrm{O}(n \log^2 n)$.

LO2 a. Demonstrate the `Median-of-Five Find Statistic Algorithm` when applied to the array

$$a = 86, 96, 35, 16, 36, 94, 55, 50, 77, 8, 44, 15, 19, 32, 55$$

with $k = 8$, lower $= 0$, and upper $= 14$. Compute the pivot M, and the arrays a_{left} and a_{right} that result after applying the partitioning algorithm from Hoare's `Quicksort`. Was the problem solved at the root level of recursion? If not, which array will be searched next and what will be the new lower and upper indices? Hint: although you do not need to show the step-by-step details of the partitioning step, an incorrect partitioning will result in a no pass.

b. What is the divide-and-conquer recurrence for the divide-and-conquer algorithm that solves the `Minimum Positive Subsequence Sum` problem? Defend your answer by desribing the algorithm using sufficient detail.