CS101 Algorithms and Data Structures
Fall 2022
Homework 10

Due date: 23:59, December 4th, 2022

1. Please write your solutions in English.
2. Submit your solutions to gradescope.com.
3. Set your FULL name to your Chinese name and your STUDENT ID correctly in Account Settings.
4. If you want to submit a handwritten version, scan it clearly. `CamScanner` is recommended.
5. When submitting, match your solutions to the problems correctly.
6. No late submission will be accepted.
7. Violations to any of the above may result in zero points.

1. (8 points) Multiple Choices

Each question has **one or more** answer(s). Select all the answer(s). For each question, you will get 0 points if you select one or more wrong answers, but you will get 1 point if you select a non-empty subset of the answers.

Write your answers in the following table.

(a)	(b)	(c)	(d)
AC	C	B	BC

(a) (2') Which of the followings are true?

A. One Floyd-Warshall algorithm's step is to find $d_{i,j}^{(k)}$: that is, the shortest path allowing intermediate visits to vertices $v_1, v_2, ..., v_{k-1}, v_k$.

B. Like Dijkstra's algorithm, Floyd-Warshall algorithm cannot work with any graph with negative weight edge.

C. Floyd-Warshall algorithm can find the shortest path between all pairs of nodes while Dijkstra's algorithm is used to find single-source shortest path.

D. Floyd-Warshall algorithm uses greedy idea.

(b) (2') Which of the followings are true?

A. For A* graph search algorithm, it will always return an optimal solution if it exists.

B. For A* graph search algorithm with admissible heuristic, it will always return an optimal solution if it exists.

C. For A* graph search algorithm with consistent heuristic, it will always return an optimal solution if it exists.

D. None of the above.

(c) (2') We run Floyd-Warshall algorithm on a graph with n vertices $v_1, v_2, ..., v_{n/2}, v_n$ (n is even). Suppose all three loops (k, i, j) are iterated from 1 to n. After running at least _____ iterations of the out-most loop k, it is ensured to find the shortest path between $v_{n/2}$ and v_n.

A. $\frac{n}{2} - 1$

B. $\frac{n}{2}$

C. $n - 1$

D. n

(d) (2') Which of the following statements about shortest path algorithms is/are true?

A. If we modify the outer loop of Dijkstra's algorithm to execute $|V| - 1$ iterations instead of $|V|$ iterations (i.e. only pop $|V| - 1$ times from the heap), the algorithm can still find the shortest path on a positive-weighted graph.

B. We can modify Bellman-Ford algorithm to detect whether there exists a negative cycle or not in a directed graph.

C. If we modify the outer loop of Bellman-Ford algorithm to execute $|V|$ iterations instead of $|V| - 1$ iterations (i.e. apply update rule to each edge for $|V|$ times), the algorithm can still find the shortest path on a positive-weighted graph.

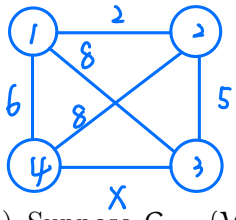
D. We can modify Floyd-Warshall algorithm to detect whether there exists a negative cycle or not in a directed graph.

2. (7 points) Shortest Path

(a) (3') Consider the weighted undirected graph with 4 vertices, where the weight of edge i, j is given by the entry $W_{i,j}$ in the matrix W,

$$W = \begin{bmatrix} 0 & 2 & 8 & 6 \\ 2 & 0 & 5 & 8 \\ 8 & 5 & 0 & x \\ 6 & 8 & x & 0 \end{bmatrix}$$

We want to find the largest possible integer value of x, for which at least one shortest path between some pairs of vertices will definitely contain the edge with weight x. What is this largest possible integer value of such x? Explain your reason briefly. When breaking tie, the path may be random.



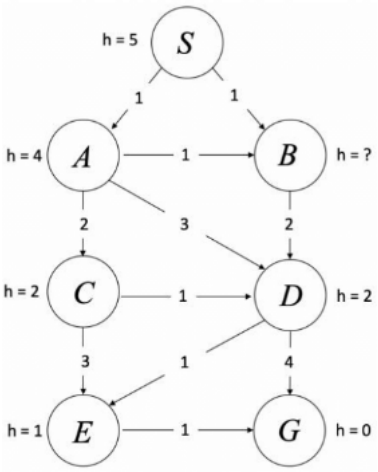
Let graph vertices are 1,2,3,4. To find the largest possible integer value of x, for which at least one shortest path between some pair of vertices will contain the edge with weight x. So, find the shortest path from 3 to 4 without including x(through 3-2-1-4)=5+2+6=13. So the largest possible value of x is 13.

(b) (4') Suppose $G = (V, E)$ is a weighted graph and T is its shortest-path from source s to t. If we increase all weights in G by the same amount, i.e., $\forall e \in E$, $w'_e = w_e + c$. Is T still the shortest-path from source s to t of the new graph? If yes, prove the statement. Otherwise, give a counter example.

Yes. Any linear transformation of all weights maintains all relative path lengths, and thus shortest paths will continue to be shortest paths, and more generally all paths will have the same relative ordering.

3. (10 points) A* algorithm

Consider A* algorithm on the following graph. Edges are labeled with their costs, and heuristic values h for nodes are labeled next to the nodes. S is the start node, and G is the goal node. Assume ties are broken in alphabetical order. Write your answer in the spaces provided.



(a) (3') With the heuristic values fixed for all nodes other than B, for which values of h(B) will A* graph search be guaranteed to return the optimal path? (All heuristics are consistent.) Either fill in the lower and upper bound in the spaces below or write "impossible" in two spaces.

2 ≤ h(B) ≤ 4

(b) (3') Given the above heuristics, and choose (one of) the heuristic you answered in (a), write down the heuristic h(B) you used. What is the order of the nodes that are going to be expanded in, assuming we run A* graph search on it.

h(B) = 2

Expand order:

S,A,B,D,E,G

(c) (4') Based on (b), what path is returned?

S – B – D – E – G

4. (10 points) Floyd-Warshall algorithm

(a) (4') Consider the following pseudo-code of the Floyd-Warshall algorithm. Assume $w_{ij} = \infty$ where there is no edge between vertex i and vertex, and assume $w_{ii} = 0$ for every vertex i.

Algorithm 1 Floyd-Warshall

```
for i = 1 to n do
  for j = 1 to n do
    A[i, j, 0] = w_ij
    P[i, j] = −1
  end for
end for
for k = 1 to n do
  for i = 1 to n do
    for j = 1 to n do
      A[i, j, k] = A[i, j, k − 1]
      if A[i, j, k] > A[i, k, k − 1] + A[k, j, k − 1] then
        A[i, j, k] = A[i, k, k − 1] + A[k, j, k − 1]
        P[i, j] = k
      end if
    end for
  end for
end for
```

Assume matrix P, the output of the above algorithm, is given. Design an algorithm for finding the shortest path from u to v by using matrix P. You can show your algorithm through natural language or pseudo-code.

Traverse through each edge and start comparing the weights after introducing the transition point. The cost of connecting the front edge is greater than the cost of adding the k th vertex as the transit point, indicating that the cost of introducing the k th vertex as the transit point between the i th vertex and the k vertex is smaller. At this time to modify the graph edge information, and modify the matrix corresponding unit information, into the k vertex information.

(b) (6') Consider the following implementation of the Floyd-Warshall algorithm. Assume $w_{ij} = \infty$ where there is no edge between vertex i and vertex j, and assume $w_{ii} = 0$ for every vertex i. Add some codes in the blank lines to detect whether there is **negative cycles** (the sum of weights of edges on the cycle is negative) in the graph. (You may not use all blank lines.)

```
bool detectNegCycle(int graph[][V])
{
    int dist[V][V], i, j, k;

    for (i = 0; i < V; i++)
        for (j = 0; j < V; j++)
            dist[i][j] = graph[i][j];

    for (k = 0; k < V; k++) {
        for (i = 0; i < V; i++) {
            for (j = 0; j < V; j++) {
                if (dist[i][k] + dist[k][j] < dist[i][j])
                        dist[i][j] = dist[i][k] + dist[k][j];
            }
        }
    }
    for (int i=0 ; i<V ; i ++)
    ------------------------------------
        if (dist [i][i] < 0)
    ------------------------------------
            return true;
    ------------------------------------
    ------------------------------------
    ------------------------------------

    return false;
}
```