1. [25 points] TRUE/FALSE OR PICK ONE . *No need for justification.*

(a) TRUE/FALSE

directed.

(b) TRUE/FALSE

(e) PICK ONE

Let $G = (V, E)$ be a directed graph with edge-weight function $w : E \to R$. Consider an adjacency matrix $A = (a_{ij})$ where $a_{ij} = w(i, j)$ or ∞. Let $d_{ij}^{(m)}$ denote the weight of a shortest path from i to j that uses at most m edges. Which of the following recurrences correctly formulate a dynamic programming solution for the all-pairs shortest path problem?

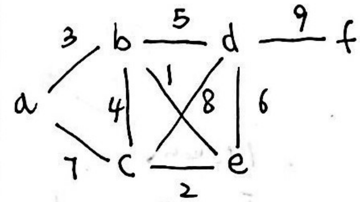
✓ (i) $d_{ij}^{(m)} = \min_{1 \le k \le n}\{d_{ik}^{(m-1)} + a_{kj}\}$ for $m = 1, 2, \ldots, n-1$

(ii) $d_{ij}^{(m)} = \min\{d_{ij}^{(m-1)} + a_{ij}\}$ for $m = 1, 2, \ldots, n$

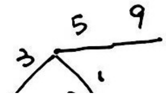
(iii) $d_{ij}^{(m)} = \min_m\{d_{ik}^{(m-1)} + a_{kj}\}$ for $k = 1, 2, \ldots, n$

(iv) $d_{ij}^{(m)} = \min_{i \le k \le j}\{d_{ik}^{(m-1)}\} + a_{kj}$ for $m = 1, 2, \ldots, n-1$

(v) none of the above



2. [30 points] SAFEST PATHS AND SPANNING TREES
For an undirected graph $G = (V, E)$, let V represent the campsites in the Everglades Park,



(a) Prove that a safest path between two campsites is always on a minimum spanning tree.

(b) Design and analyze an algorithm that computes the safest path for every pair of camp-

not in MST. Contradiction! Thus the shortest path detween two campsites, ie. the safest path is always on a MST.

Another way to proof: (if not by contradiction)

10

For pair (u, v), we can know the shortest edge on the path should be on MST (based on MST's property). If we remove that shortest edge, we can't find one path containing new shortest edge which is smaller than previous shortest edge. Thus the path should on MST.

(b) Algorithm:

① Use kruskal's method to build MST.

② For pair (u, v), travelsal starting from u, along the path in MST, stop at v.

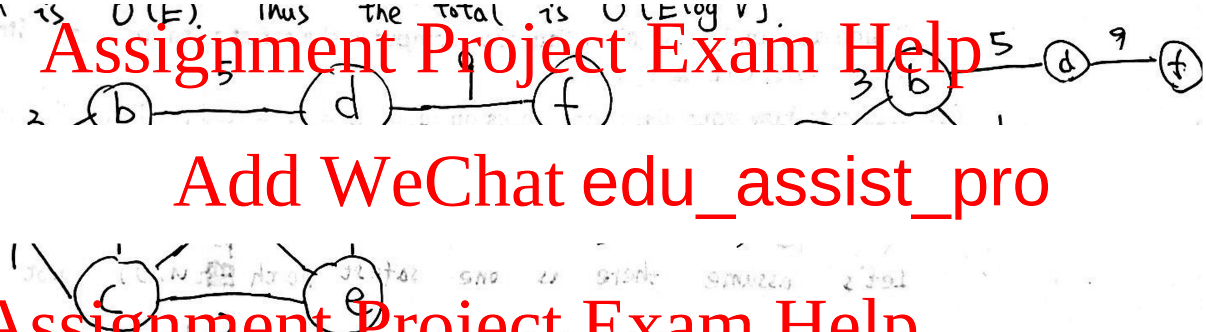
③ connection of these edges along the path is the safest path.

Analysis:

For the proof in (a) we know the shortest path must on the MST

path is O(E). Thus the Total is O(E log V).

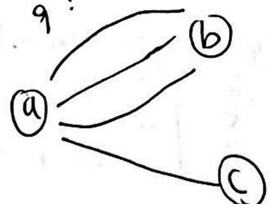
(3)

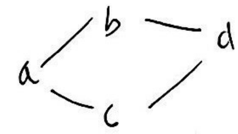


S(a, d) = a - b - d		
S(a, e) = a - b - e	1	
S(a, f) = a - b - d - f	3	
S(b, c) = b - e - c	1	
S(b, d) = b - d	5	
S(b, e) = b - e	1	
S(b, f) = b - d - f	5	
S(c, d) = c - e - b - d	1	
S(c, e) = c - e	2	
S(c, f) = c - e - b - d - f	1	
S(d, e) = d - b - e	1	S(e, f) = e - b - d - f 1
S(d, f) = d - f	9	

we can easily see all these danger level in the path contain smallest danger level edge.

9:





15/25

of whether we can add that edge to the set of edges which we have already determined the shortest travel time

for this problem, we can use ... Dynamic Programming to solve, since greedy method will cut off so many leaves, which will occur to miss the minima.

Algorithm: (u, v)

① Initial list T_a for storage of departure time, T for travel time of elements have checked, S for all the place vertices.

② Loop for start from u, find the time by $(t_d - t_a)$, ~~start~~ add to T, add t_a to T_a. Then check the next edge whether $t_a' > t_{d_1} + 0.5h$, if so, ~~check the~~ add t_2 to t, store in T.

Recursively. $O(|V||E|)$ complexity.

Initial Flow	Augmentation P	$c_f(p)$	Final Flow
0	s - a - d - t	5	5
5	s - b - d - t	2	7
7	s - c - d - t	4	11

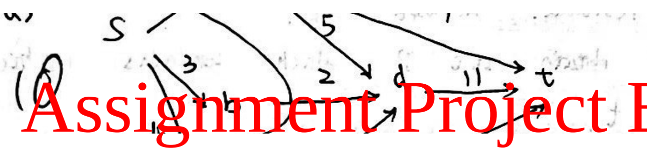
4. [10+5+5+10+10 = 40 ...

7) ... increase in the maximum flow. List all bottleneck edges in G.

(c) Can there be a flow network which has no bottleneck edges? Justify your answer.

(d) Design and analyze an efficient algorithm to identify all bottleneck edges in a network.

(a) 7



min cut: 5 + 2 + 4 = 11

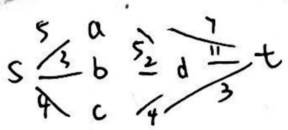
S = {s, b}

(b) 0 bottle neck edges in this graph:

(a, d), (b, d), (c, d)

(c). 5 Yes. There can be a flow network without bottle neck edges.

The simplest example can be $a \xrightarrow{1} b \xrightarrow{1} t$



(d) we can find the leftside of bottle neck edge should

② Build residual network G_f.

④ Build reverse residual network G_f.

⑤ Do a BFS to obtain set B which contains vertices which are reachable to t

Initial Flow	Augmentation path p	$C_f(p)$	Final Flow
0	s - a - t	5	5
5	s - b - d - t	2	7
7	s - c - d - t	4	11

③ Edmonds-Karp Algorithm.

Initial Flow	Augmentation path p	$C_f(p)$	Final Flow
0	s - a - t	5	5
5	s - c - t	3	8
8	s - b - d - t	2	10
10	s - c - d - t	1	11

Thus the augmentations of Edmonds-Karp is more than Ford-Fulkerson.