Quiz 2

Friday, March 29, 2019 6:08 PM

quiz2

Design and Analysis of Algorithms
Massachusetts Institute of Technology
Profs. Erik Demaine, Srini Devadas, and Nancy Lynch

April 16, 2015
6.046J/18.410J
Quiz 2

Quiz 2

- Do not open this quiz booklet until you are directed to do so. Read all the instructions first.
- The quiz contains 6 problems, with multiple parts. You have 120 minutes to earn 120 points.
- This quiz booklet contains 10 pages, including this one.
- This quiz is closed book. You may use two double-sided letter ($8\frac{1}{2}'' \times 11''$) or A4 crib sheets. No calculators or programmable devices are permitted. Cell phones must be put away.
- Do not waste time deriving facts that we have studied. Just cite results from class.
- When we ask you to "give an algorithm" in this quiz, describe your algorithm in English or pseudocode, and provide a short argument for correctness and running time. You do not need to provide a diagram or example unless it helps make your explanation clearer.
- Do not spend too much time on any one problem. Generally, a problem's point value is an indication of how many minutes to spend on it.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Please be neat.
- Good luck!

Problem	Title	Points	Parts	Grade	Initials
1	True or False	40	10		
2	Who Charged the Electric Car?	20	3		
3	Planning Ahead	10	1		
4	Maze Marathoner	20	3		
5	6.046 Carpool	10	1		
6	Paths and/or Cycles	20	2		
Total		120			

106

Name: _____

120 =

.883

Problem 1. True or False. [40 points] (10 parts)

Circle **T** or **F** for each of the following statements to indicate whether the statement is true or false **and briefly explain why**.

FW uses matrix operations, not matrix multiplications

(a) T **F** [4 points]

Using similar techniques used in Strassen's matrix multiplication algorithm, the Floyd–Warshall algorithm's running time can be improved to $O(V^{\log_2 7})$.

APSP using BF to ... edges + Dijkstra. Strassen improvements on related.

Strassen: more efficient Matmul w/ DnC

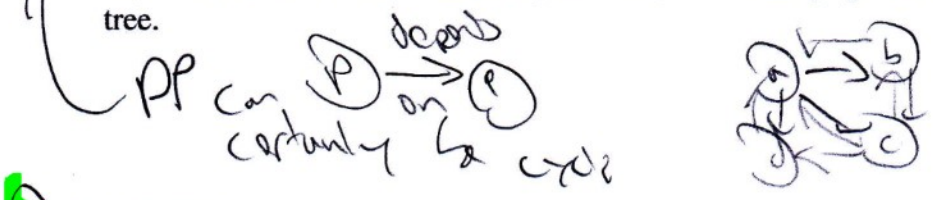
(b) **T** F [4 points]

For graphs $G = (V, E)$ where $E = O(V^{1.5})$, Johnson's algorithm is asymptotically faster than Floyd–Warshall.

Johnson: $O(EV)$ + $O(V(V \lg V))$ FW: $O(V^3)$

(c) T **F** [4 points]

Consider the directed graph where each vertex represents a subproblem in a dynamic program, and there is an edge from p to q if and only if subproblem p depends on (recursively calls) subproblem q. Then this graph is a directed rooted tree.



PP can certainly have cycles

(d) **T** F [4 points]

In a connected, weighted graph, every lowest weight edge is always in *some* minimum spanning tree.

consider MST, cut the edge it uses and put in the min weight edge

(e) T **F** [4 points]

For a connected, weighted graph with n vertices and exactly n edges, it is possible to find a minimum spanning tree in $O(n)$ time.

~~select/all~~ tree has n-1 edges. use DFS to find a cycle, kill max weight edge from cycle

(f) T F [4 points]

For a flow network with an integer capacity on every edge, the Ford–Fulkerson algorithm runs in time $O((V + E)\,|f|)$ where $|f|$ is the maximum flow.

Since, every iteration of FF guarantees a flow more of 1

(g) T F [4 points]

Let $C = (S, V \setminus S)$ be a minimum cut in a flow network. If we strictly increase the capacity of every edge across C, then the maximum flow of the network must increase.

→ there may be other min-cuts

(h) T F [4 points]

Every linear program has a unique optimal solution.

Alternative solution: False. There could be no solutions at all.

Alternative solution: False. There could be no solutions at all.

Consider a variant of LP w/ unique sln modified to have infinitely variable

(i) T F [4 points]

3SAT cannot be solved in polynomial time, even if P = NP.

3SAT is a reduction of many NP problems ☺

(j) T F [4 points]

Repeatedly selecting a vertex of maximum degree, and deleting the incident edges, is a 2-approximation algorithm for Vertex Cover.

X

at least 4 and...

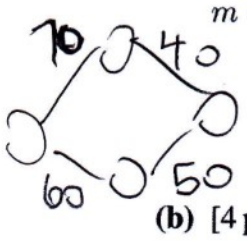
→ ever letter? could be

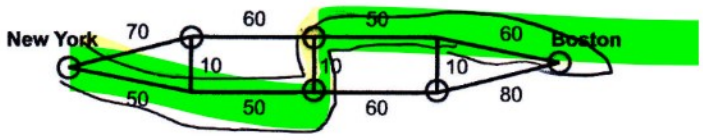
Problem 2. Who Charged the Electric Car? [20 points] (3 parts)

Prof. Musk is driving his Nikola electric car from Boston to New York. He wants to take the shortest path, but his car can only drive m miles before needing to charge. Fortunately, there are Furiouscharger charging stations on the way from Boston to New York, which instantaneously charge the battery to full.

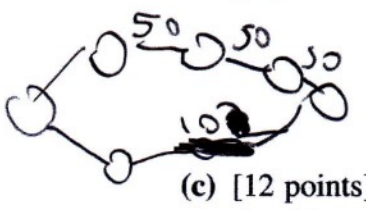
The road network is given to you as a weighted undirected graph $G = (V, E, w)$ along with the subset $C \subseteq V$ of vertices that have charging stations. Each weight $w(e)$ denotes the (positive) length of road e. The goal is to find a shortest path from node $s \in V$ to node $t \in V$ that does not travel more than m miles between charging stations. Assume that $s, t \in C$.

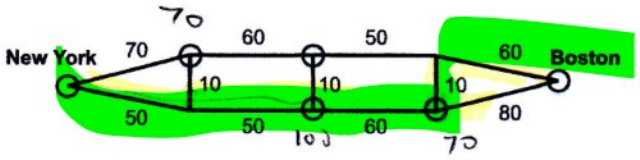
(a) [4 points] Draw the shortest path from Boston to New York in the following graph if $m = \infty$. Charging stations are marked as circles.



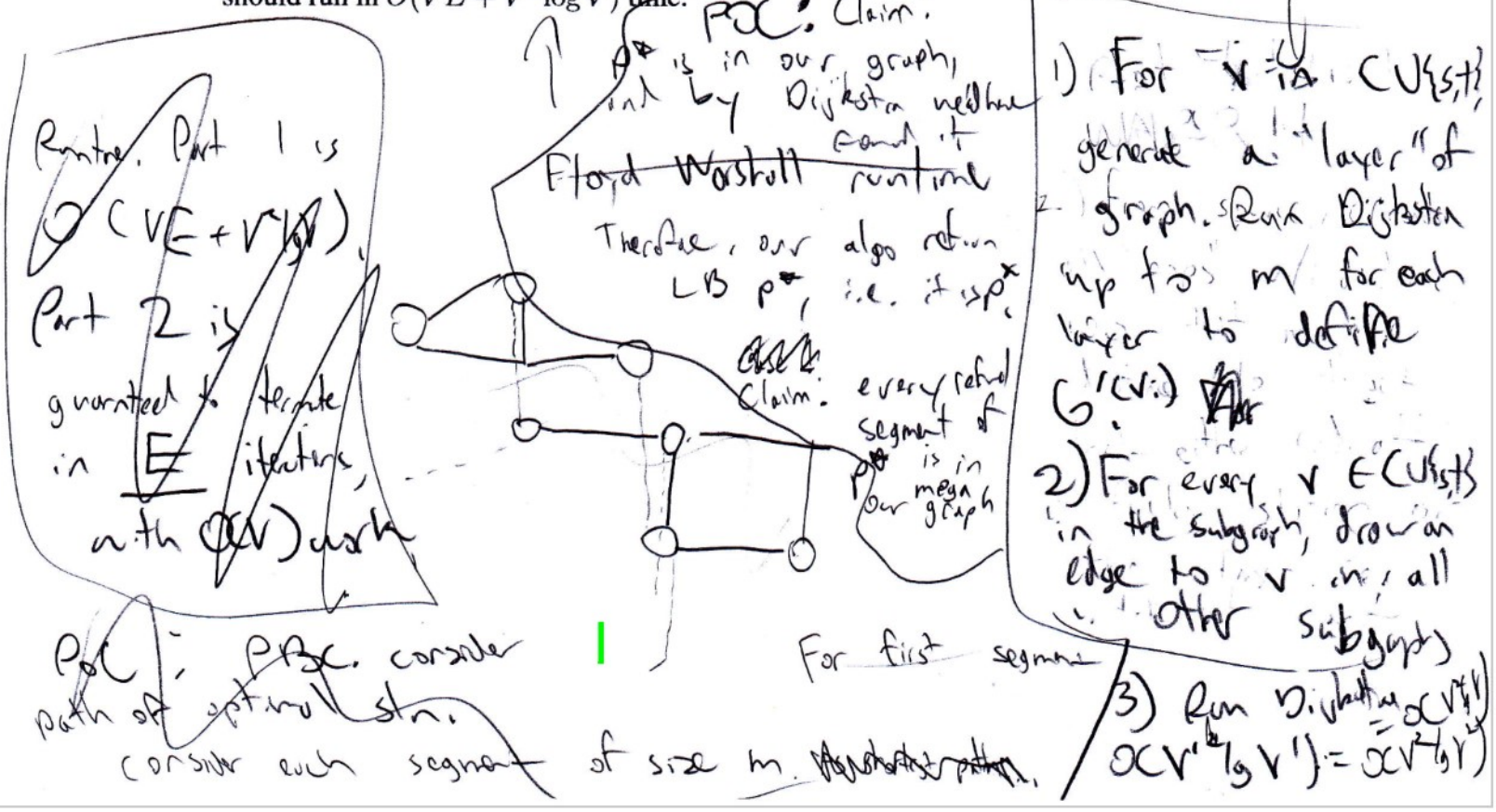


(b) [4 points] Draw the shortest path from Boston to New York in the following (identical) graph if $m = 100$.





(c) [12 points] Give an algorithm to solve the problem. For full credit, your algorithm should run in $O(VE + V^2 \log V)$ time.

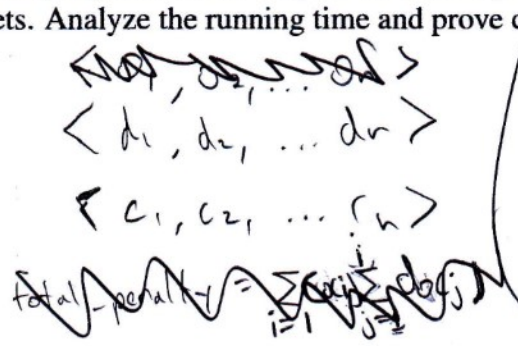


Problem 3. Planning Ahead [10 points] (1 part)

You have N psets due right now, but you haven't started any of them, so they are all going to be late. Each pset requires d_i days to complete, and has a cost penalty of c_i *per day*. So if pset i ends up being finished t days late, then it incurs a penalty of $t \cdot c_i$. Assume that once you start working on a pset, you must work on it until you finish it, and that you cannot work on multiple psets at the same time.

For example, suppose you have three problem sets: 6.003 takes 3 days and has a penalty of 12 points/day, 6.046 takes 4 days and has a penalty of 20 points/day, and 6.006 takes 2 days and has a peanlty of 4 points/day. The best order is then 6.046, 6.003, 6.006 which results in a penalty of $20 \cdot 4 + 12 \cdot (4+3) + 4 \cdot (3+4+2) = 200$ points.

Give a greedy algorithm that outputs an ordering of the psets that minimizes the total penalty for all the psets. Analyze the running time and prove correctness.



Assume sorted order opt on probs size $n+1$. BCCD is trivially true IS', PBC otherwise.

Suppose biggest not first, must be second: cost diff. This is positive if $c_2 d_1 - c_1 d_2$. This is positive if $c_2/d_2 > c_1/d_1$, c_i/d_i must be by points/day.

Algorithm sort psets in descending order

Return $\sum_{i=1}^{n} c_i^* \sum_{j=1}^{i} d_j^*$

Runtime: $O(n \lg n)$

↳ Sort; $O(n \lg n)$

↳ Evaluate penalty → nested summation $O(n^2)$

↳ $O(n)$ if you're smart

↳ GenerateArray DT where $DT_i = \sum_{j=1}^{i} d_j^*$

take this and ref dot product of $DT_i \cdot c_i$

Correctness (Optimality): GA returns result for n tasks, ∀ n

PBT: BC: PC(): GA = OPT on pick 1 (true).

no proof IS: Given GA = opt agree on problem of size n,

Consider size n+1.

later, time or -

Bubble-sort proofing.

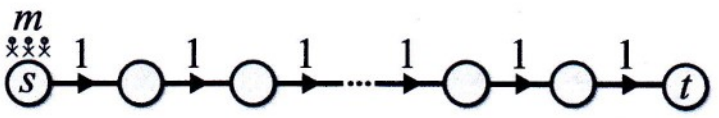
Any "out of order" element can be AUGMENTED via bubbleswap.
We repeat this until the optimal form by bubbleswap leaves us in… sorted form!

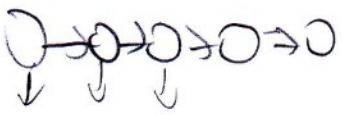
We get GUARANTEED improvement by reducing number of inversions

Problem 4. Maze Marathoner [20 points] (3 parts)

A group of m teens need to escape a maze, represented by a directed graph $G = (V, E)$. The teens all start at a common vertex $s \in V$, and all need to get to the single exit at $t \in V$. Every night, each teen can choose to remain where they are, or traverse an edge to a neighboring vertex (which takes exactly one night to traverse). However, each edge $e \in E$ has an associated capacity $c(e)$, meaning that at most $c(e)$ teens can traverse the edge during the same night. The goal is to minimize the number of nights required for all teens to escape by reaching the goal t.

(a) [3 points] First look at the special case where the maze is just a single path of length $|E|$ from s to t, and all the edges have capacity 1 (see below). Exactly how many nights are required for the teens to escape?





teens will exit at a rate of ~~|f|~~ |f| w/ an offset of |E| for the flow to initialize.

$$T_{min} = |E| + \frac{m}{|f|} = |E| + m$$

(b) [7 points] The general case is more complex. Assume for now that we have a "magic" algorithm that calculates whether the teens can all escape using $\leq k$ nights. The magic algorithm runs in polynomial time: $k^\alpha T(V, E, m)$ where $\alpha = O(1)$.

Give an algorithm to calculate the minimum number of nights to escape, by making calls to the magic algorithm. Analyze your time complexity in terms of V, E, m, α, and $T(V, E, m)$.

max nights is $|E|+m$. We can use our Magic Algorithm $\log(|E|+m)$ times to find the minimum nights by successive space pruning

min = 0, max = $|E|+m$

while min ≠ max:

  if M(⌊(max+min)/2⌋): max = ⌊(max+min)/2⌋

  else: min = ⌊(max+min)/2⌋

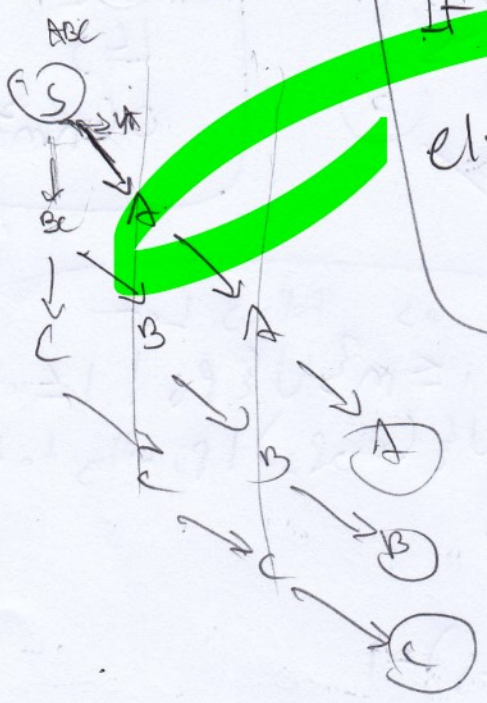
Time: $\log(|E+m)\, k^\alpha T(V, E, n)$

(c) [10 points] Now give the "magic" algorithm, and analyze its time complexity.

Hint: Transform the problem into a max-flow problem by constructing a graph $G' = (V', E')$ where $V' = \{(v, i) \mid v \in V, 0 \le i \le k\}$. What should E' be?

$E' = \{((u_i, u_{i+1}) \mid u \in V, 0 \le i < k\} \cup \{(u_i, v_{i+1}) \mid (u, v) \in E, 0 \le i < k\}$

$c'(u_i, v_i) = \begin{cases} c(u,v) & (u,v) \in E \\ \infty & \text{else} \end{cases}$



If $E = \text{dmaxflow}(G') \ge k$
return True
else:
return False.

Runtime: $O(V'E'^2)$
$= O(kV \cdot (kE + kV))^2$
$= O(k^3 VE^2)$

-3

Problem 5. 6.046 Carpool [10 points] (1 part)

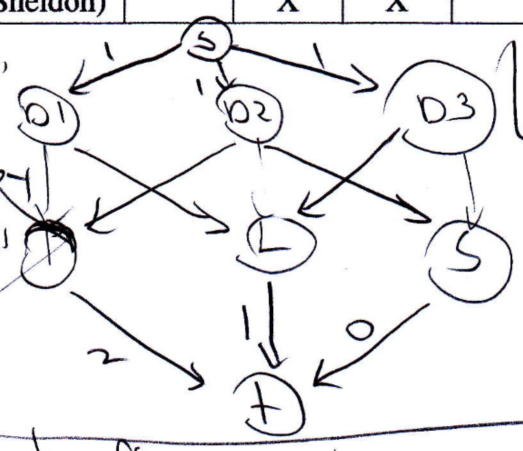
The n people in your dorm want to carpool to 34-101 during the m days of 6.046. On day i, some subset S_i of people actually want to carpool (i.e., attend lecture), and the driver d_i must be selected from S_i. Each person j has a limited number of days ℓ_j they are willing to drive.

Give an algorithm to find a driver assignment $d_i \in S_i$ for each day i such that no person j has to drive more than their limit ℓ_j. (The algorithm should output "no" if there is no such assignment.) *Hint:* Use network flow.

For example, for the following input with $n = 3$ and $m = 3$, the algorithm could assign Penny to Day 1 and Day 2, and Leonard to Day 3.

Person	Day 1	Day 2	Day 3	Driving limit
1 (Penny)	X	X	X	2
2 (Leonard)	X		X	1
3 (Sheldon)		X	X	0

Claim 1: valid driving configs bijectively map to a flow state
1 driver by day, flow only from valid drivers, driver limited by it's cap
Claim 2: max flow is OBJ by min (cut), if |f|=m, drivers covered on all days



V = O(m+n)
E = O(nm)

Runtime: O(VE²) via EK
LL)
O(n³m² + n²m³)

Generate network flow graph as follows. Let
Let $V = \{s, t\} \cup \{d_i \mid 1 \le i \le m\} \cup \{p_j \mid 1 \le j \le n\}$
Let $E = \{(s, d_i) \mid 1 \le i \le n\} \cup \{(d_i, p_j) \mid p_j \text{ willing to drive on } d_i\} \cup \{p_j, t \mid 1 \le j \le n\}$

Let $c(s, d_i) = 1$, $c(d_i, c_j) = 1$
$c(p_j, t) = \ell_j$

Solve IF using Edmonds-Karps.
If $|f| < m$: return no.
else: return $\{str(d_i) + \text{"drives on day \%s" \%(driver w/ nonzero flow from this day)} \mid 1 \le i \le m\}$

Problem 6. Paths and/or Cycles [20 points] (2 parts)

A ***Hamiltonian path*** on a directed graph $G = (V, E)$ is a path that visits each vertex in V exactly once. Consider the following variants on Hamiltonian path:

(a) [10 points] Give a polynomial-time algorithm to determine whether a directed graph G contains *either* a cycle or a Hamiltonian path (or both).

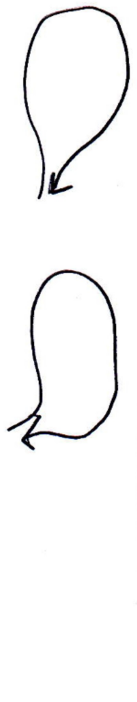
Run DFS w/ back-edge detection to find cycles. W/ no cycles, run 1 tree DFS and see if you hit all nodes.

~~Ham cycle finding is 2^V time.~~

for Ham path, enumerate all $|V|!$ sequences of vertices + evaluate if $(v_i, v_{i+1}) \in V$ in sequence.

Run DFS and check if the length ever reaches |V|

(b) [10 points] Show that it is NP-hard to decide whether a directed graph G' contains *both* a cycle and a Hamiltonian Path, by giving a reduction from the HAMILTONIAN PATH problem: given a graph G, decide whether it has a Hamiltonian path. (Recall from recitation that the HAMILTONIAN PATH problem is NP-complete.)



In NP: a certificate would be the vertex sequence of a cycle AND the vertex sequence of HP, both verifiable in $|V|$ time

NP Hard

~~NP Complete~~: Ham Path can be reduced to Ham Cycle by checking for cycle.

if cycle ⇒ do nothing

no cycle ⇒ add a self loop or similar

NP Complete

Convert graph to digraph. Spit it to oracle.
If oracle says yes, tehn it has a ham path and a cycle
RETURN YES
If oracle says no… check for a cycle

If there's a cycle, RETURN NO.

If there's no cycle….
We have TREE HAM PATH. Do a topo sort, run DFS, check if length ever hits |V|