HKN Review Session! Sunday 11-1 ECEB 1015

Postmortem comments in green

CS/ECE 374 A ✧ Spring 2018

Fake Midterm 2

April 9, 2018

Real name:	Jeff Erickson
NetID:	jeffe

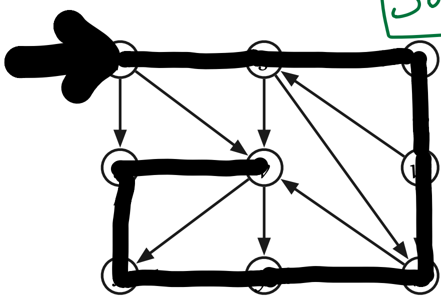
Gradescope name:	Flash
Gradescope email:	Flash@gordon.mil

- ***Don't panic!*** — This is harder than the real exam!
- **All problems are described in more detail in a separate handout.** If any problem is unclear or ambiguous, please don't hesitate to ask us for clarification.
- If you brought anything except your writing implements, your **hand-written** double-sided 8½" × 11" cheat sheet, and your university ID, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.
- Please clearly print your real name, your university NetID, your Gradescope name, and your Gradescope email address in the boxes above. However, if you are using your real name and your university email address on Gradescope, you do ***not*** need to write everything twice. **We will not scan this page into Gradescope.**
- Please also print **only the name you are using on Gradescope** at the top of every page of the answer booklet, except this cover page. These are the pages we will scan into Gradescope.
- Please do not write outside the black boxes on each page; these indicate the area of the page that the scanner can actually see.
- If you run out of space for an answer, feel free to use the scratch pages at the back of the answer booklet, but **please clearly indicate where we should look.**
- Except for greedy algorithms, proofs are required for full credit if and only if we explicitly ask for them, using the word ***prove*** in bold italics.
- Please return ***all*** paper with your answer booklet: your question sheet, your cheat sheet, and all scratch paper.

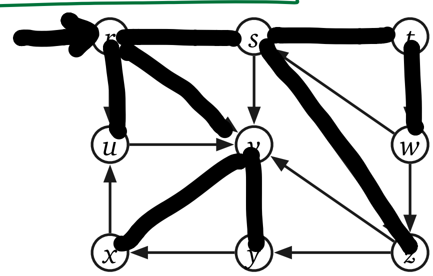
CS/ECE 374 A ✧ Spring 2018 Fake Midterm 2 Problem 1	Gradescope name: Flash

Clearly indicate the following structures in the directed graph below, or write NONE if the indicated structure does not exist. Don't be subtle; to indicate a collection of edges, draw a heavy black line along the entire length of each edge.

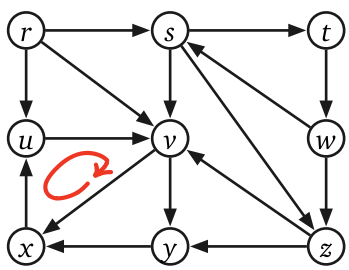
Solutions are not unique!



(a) A depth-first spanning tree rooted at r



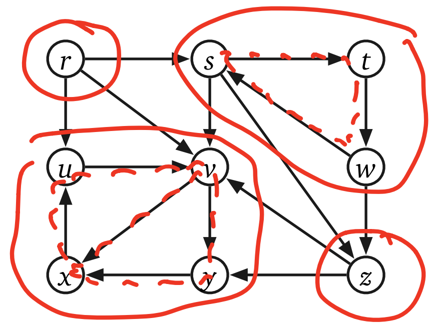
(b) A breadth-first spanning tree rooted at r



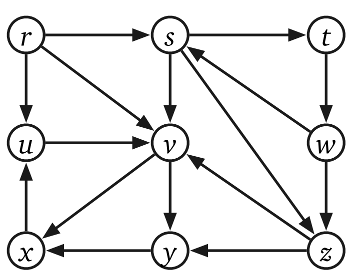
(c) A topological order (list vertices below)

NONE

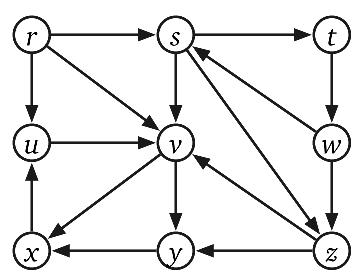
because G has cycle(s)



(d) The strongly connected components (circle each component)



[scratch]

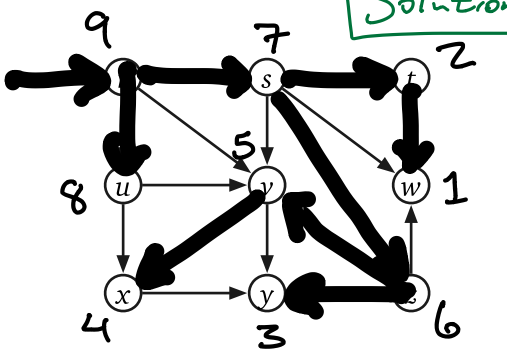


[scratch]

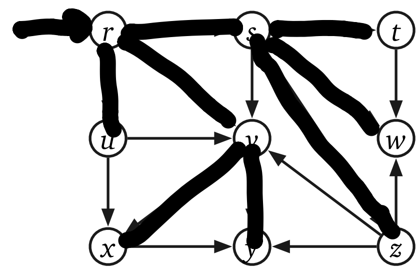
CS/ECE 374 A ✧ Spring 2018 Fake Midterm 2′ Problem 1	Gradescope name:

Clearly indicate the following structures in the directed graph below, or write NONE if the indicated structure does not exist. Don't be subtle; to indicate a collection of edges, draw a heavy black line along the entire length of each edge.

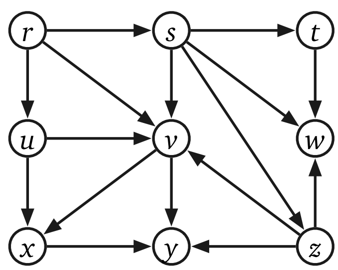
Solutions are not unique!



(a) A depth-first spanning tree rooted at r

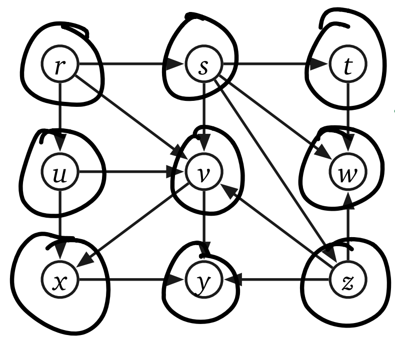


(b) A breadth-first spanning tree rooted at r



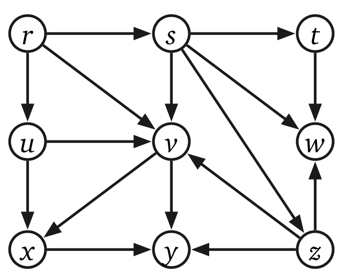
(c) A topological order (list vertices below)

r u s z v x y t w

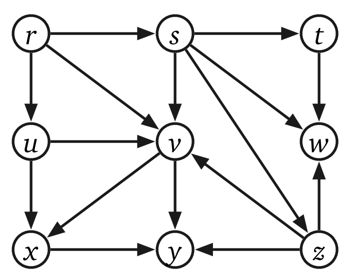


isolated vertices because G is a dag

(d) The strongly connected components (circle each component)



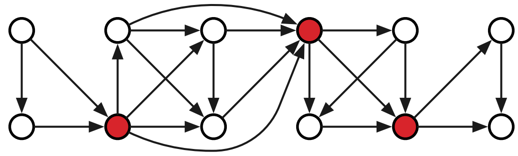
[scratch]



[scratch]

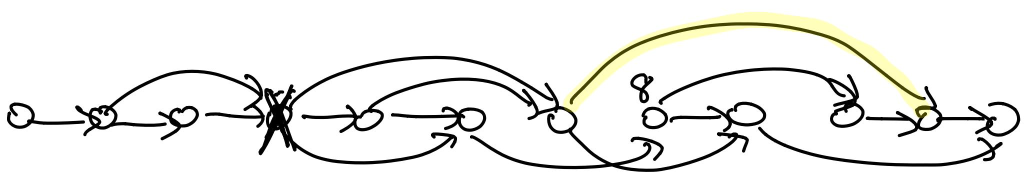
CS/ECE 374 A ✧ Spring 2018 Fake Midterm 2 Problem 2	Gradescope name:

A vertex v in a (weakly) connected graph G is called a ***cut vertex*** if the subgraph $G - v$ is disconnected. For example, the following graph has three cut vertices, which are shaded in the figure.



Suppose you are given a (weakly) connected *dag* G with one source and one sink. Describe and analyze an algorithm that returns TRUE if G has a cut vertex and FALSE otherwise.

Topological sort G



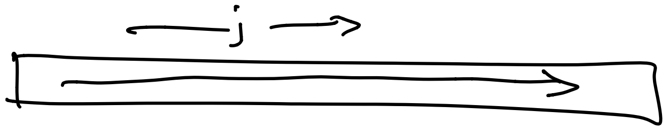
want a vertex that no edge "skips over."

j $i \to k$ $i < j < k$

~~Naive:~~
~~unmark everything~~
~~for all $i \to k$ in E~~
~~for $j \in i+1$ to $k-1$~~
~~mark j~~
~~if any marked~~
~~return T~~
~~$O(VE)$ time~~

Let Furthest(j) = rightmost vertex at head of edge whose tail is $1..j$

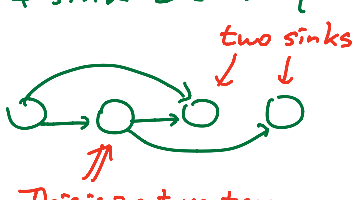
$$\text{Furthest}(j) = \begin{cases} 0 & \text{if } j = 0 \\ \max \begin{Bmatrix} \text{Furthest}(j-1) \\ \max\{k \mid j \to k \in E\} \end{Bmatrix} \end{cases}$$



$O(V+E)$ time

return TRUE iff Furthest($j-1$) $\leq j$ for any j.
except $j = 1$ or n

This solution only works because source + sink are unique!



CS/ECE 374 A ✧ Spring 2018 Fake Midterm 2 Problem 3	Gradescope name:

The City Council of Sham-Poobanana needs to partition Purple Street into voting districts. A total of n people live on Purple Street, at consecutive addresses $1, 2, \ldots, n$. Each voting district must be a contiguous interval of addresses $i, i+1, \ldots, j$ for some $1 \le i < j \le n$. By law, each Purple Street address must lie in exactly one district, and the number of addresses in each district must be between k and $2k$, where k is some positive integer parameter.

Every election in Sham-Poobanana is between two rival factions: Oceania and Eurasia. A majority of the City Council are from Oceania, so they consider a district to be *good* if more than half the residents of that district voted for Oceania in the previous election. Naturally, the City Council has complete voting records for all n residents.

For example, the figure below shows a legal partition of 22 addresses into 4 good districts and 3 bad districts, where $k = 2$. Each O indicates a vote for Oceania, and each X indicates a vote for Eurasia.

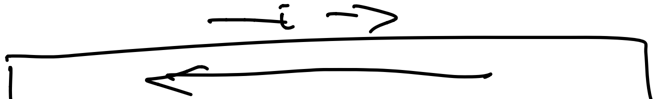
Describe an algorithm to find the largest possible number of *good* districts in a legal partition. Your input consists of the integer k and a boolean array GoodVote$[1..n]$ indicating which residents previously voted for Oceania (True) or Eurasia (False). You can assume that a legal partition exists. Analyze the running time of your algorithm in terms of the parameters n and k.

DP?

MaxGood(i) = max # good districts for addresses i..n

$$\text{MaxGood}(i) = \begin{cases} -\infty & \text{if } i > n+1 \\ 0 & \text{if } i = n+1 \\ \max\left\{ \begin{array}{l} \text{IsGood}(i,..i+\ell-1) \\ +\text{MaxGood}(i+\ell) \end{array} \middle|\, k \le \ell \le 2k \right\} \end{cases}$$

— i —>



O(nk) calls to IsGood

$O(nk\cancel{k})$ time ~~naive~~ DP see below

To get IsGood in O(1) time:

Let GoodLeft(j) = # good voters in 1..j

Then IsGood(i,j):

good ← GoodLeft[j] − GoodLeft[i−1]
if 2·good > j − i + 1 return TRUE
else return FALSE

$$\text{GoodLeft}(j) = \begin{cases} 0 & \text{if } j = 0 \\ GL(j-1) & \text{if } j \text{ bad} \\ 1 + GL(j+1) & \text{if } j \text{ good} \end{cases}$$

O(n) time

CS/ECE 374 A ✧ Spring 2018 **Fake Midterm 2 Problem 4**	Gradescope name:

After graduation, you accept a job with Aviophiles-Я-Us, the leading traveling agency for people who love to fly. Your job is to build a system to help customers plan airplane trips from one city to another. Your customers love flying, but they absolutely despise airports. You know all the departure and arrival times of all the flights on the planet.

Suppose one of your customers wants to fly from city X to city Y. Describe an algorithm to find a sequence of flights that *minimizes the total time spent in airports*. Assume (unrealistically) that your customer can enter the starting airport immediately before the first flight leaves X, that they can leave the final airport at Y immediately after the final flight arrives at Y.

Shortest path?

We have list of n flights   airpt, time $\longrightarrow$ airpt, time

$V = \{(A,t) \mid$ A flight leaves or arrives at airport A at time $t\}$
$\cup \{X, Y\}$

$E = \{(A,t) \to (A',t') \mid$ flight leaves A at time t reaches A' at time $t'\}$ $\boxed{\text{cost } 0}$

$\cup \{(A,t) \to (A,t') \mid t, t'$ consecutive events at airport $A\}$ $\boxed{\text{cost } t'-t}$

$\cup \{X \to (X,t)$ for all $t\}$ $\boxed{\text{cost } 0}$

$\cup \{(Y,t) \to Y$ for all $t\}$

Shortest path from X to Y.

This is a dag!! ☹ $V \le 2n+2$ $E = O(n)$

$O(V+E) = O(n)$ time

after sorting

$O(n \log n)$

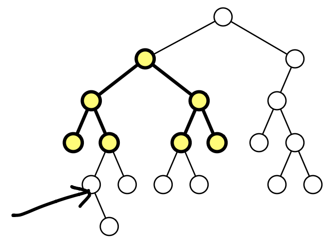
Or use Dijkstra, since we're already spending $O(n \log n)$ time sorting

If all flights are on same day and all arrival/departure times at exact minutes $\longrightarrow$ Bucketsort in $O(n)$ time

CS/ECE 374 A ✧ Spring 2018	Gradescope name:
Fake Midterm 2 Problem 5	

For this problem, a *subtree* of a binary tree means any connected subgraph. A binary tree is *complete* if every internal node has two children, and every leaf has exactly the same depth.

Describe and analyze a recursive algorithm to compute the ***largest complete subtree*** of a given binary tree. Your algorithm should return both the root and the depth of this subtree. For example, given the following tree T as input, your algorithm should return the left child of the root of T and the integer 2.



Recursion

Best(v) = max depth of complete subtree rooted at v.

$$\text{Best}(v) = \begin{cases} 0 & \text{if } v \text{ has less than 2 children} \\ 1 + \min\{\text{Best}(\text{left}(v)), \text{Best}(\text{right}(v))\} \end{cases}$$

Postorder O(v) time

return node v^* with max. Best(w) and Best(v^*)

keeptrack of this

Either memoize Best(w) at v
or maintain a global variable
or recursion returns Best(v) and BestBest(v)

$$\text{BestBest}(v) = \max\{\text{Best}(w) \mid w \text{ below } v\}$$