• Don’t panic!

• 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.
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.

(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)

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

Solutions are not unique!
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)

$ru$ $s$ $z$ $v$ $x$ $y$ $t$ $w$

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

Isolated vertices because $G$ is a DAG

[scratch]
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 $\text{TRUE}$ if $G$ has a cut vertex and $\text{FALSE}$ otherwise.

Topological sort $G$

want a vertex that no edge “ships over.”

Let $\text{Furthest}(j) = \max \{ \text{Furthest}(j-1), \max \{ k | j \rightarrow k \in E \} \}$

return $\text{TRUE}$ iff $\text{Furthest}(j-1) \leq j$ for any $j$ except $j = 1$ or $n$
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 \(1 \leq i < j \leq n\). 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 \(\text{GoodVote}[1..n]\) 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 \(k\).

\[
\text{Max Good} (i) = \begin{cases} 
-\infty & \text{if } i > n+1 \\
0 & \text{if } i = n+1 \\
\max \left[ \text{Is Good}(i, i+l-1), \sum_{k=1}^{l} \text{Max Good}(i+k) \right] & \text{for } k \leq l \leq 2k \end{cases}
\]

To get \(\text{Is Good}\) in \(O(k)\) time:

Let \(\text{Good Left}(j) = \# \text{ good voters in } 1-j\)

Then \(\text{Is Good}(i,j): \)

\[
good \leftarrow \text{Good Left}[j] - \text{Good Left}[i-1] \\
\text{if } 2 \cdot \text{good} > j-i+1 \text{ return True} \\
\text{else return False}
\]

\(O(nk)\) calls to \(\text{Is Good}\) \(\text{O(nk)}\) time \(\text{DP}\) see below

\(\text{Good Left}(j) = \begin{cases} 
0 & \text{if } j = 0 \\
\text{Good Left}(j-1) & \text{if } j \text{ bad} \\
1 + \text{Good Left}(j+1) & \text{if } j \text{ good}
\end{cases}\)
After graduation, you accept a job with Aviophiles-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 flights $\text{airpt, time} \rightarrow \text{airpt, time}$

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

$E = \{ (A, t) \rightarrow (A', t') \mid \text{ flight leaves } A \text{ at time } t \text{ reaches } A' \text{ at time } t' \}$

Cost $0$

$U \{ (A, t) \rightarrow (A', t') \mid t, t' \text{ consecutive events at airport } A \}$

Cost $t' - t$

$U \{ X \rightarrow (X, t) \mid \text{ for all } t \}$

$U \{ Y, t) \rightarrow Y \mid \text{ for all } t \}$

Shortest path from $X$ to $Y$.

This is a DAG! $U \leq 2n + 2$ $E = O(n)$

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

Dr. 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 $\rightarrow$ BucketSort in $O(n)$ time
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

$$\text{Best}(v) = \max \text{ depth of complete subtree rooted at } v.$$  

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

Post-order $O(v)$ time

return node $v$ with max. $\text{Best}(v)$ and $\text{Best}(v^*)$

keep track of this

Either memoize $\text{Best}(v)$ at $v$

or maintain a global variable

or recursion returns $\text{Best}(v)$ and $\text{Best}(\text{Best}(v))$

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