# CSE 421 Section 10

## Final Review

# Announcements & Reminders

- **HW8**
  - Was due yesterday, Wednesday 3/12
- **Final review** with Nathan: G20 this Friday 4:30 PM
  - He will go over the practice final, so try it before the session if you can
- The **final exam** is on G20 2:30-4:20 PM, Monday 3/17
  - If you are sick the day of the exam, let us know and we will schedule a makeup
- **Course evaluations** are due 3/16 at 12 PM
  - **Section evaluations** are due 3/16 at 12 PM

# Final exam format

- Similar to midterm exam, but longer
  - A sample final is available on Ed
- 135 minutes
- You will be given a standard reference sheet
  - Is expanded from the midterm, attached to sample final on Ed
- You may bring one sheet of double sided 8.5x11" paper containing your own handwritten notes.
  - Must write name, student number, and UW NetID
  - Must turn in with exam
  - If you want to access your midterm notes sheet, go to Prof. Beame's OH

# Today's plan

1. (35 min) 6 stations around the room with practice problems (focused on second half of course, but exam is cumulative)
    - Station 1: Short answer
    - Station 2: Dynamic programming*
    - Station 3: Network flow
    - Station 4: Linear programming*
    - Station 5: Reduction
    - Station 6: Bonus problem
1. (10 min) Go over some of these problems

**the problem at this station was an extra problem on a previous section handout*

# Problems

1

2

3

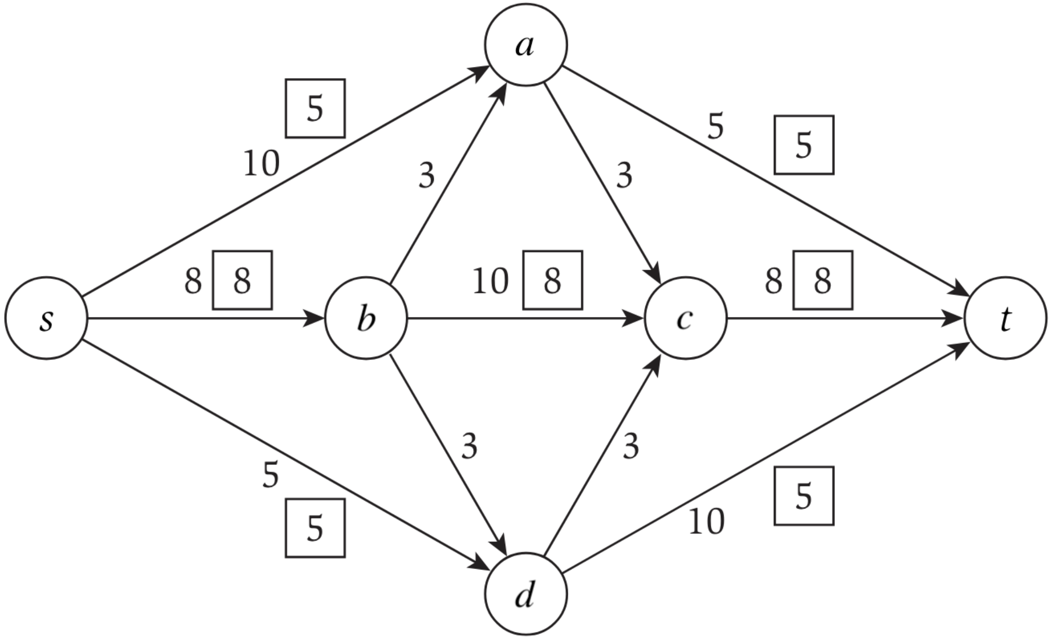
4

5

6

# Problem 1 – Short answer

In the network flow below, is the depicted flow a maximum flow?

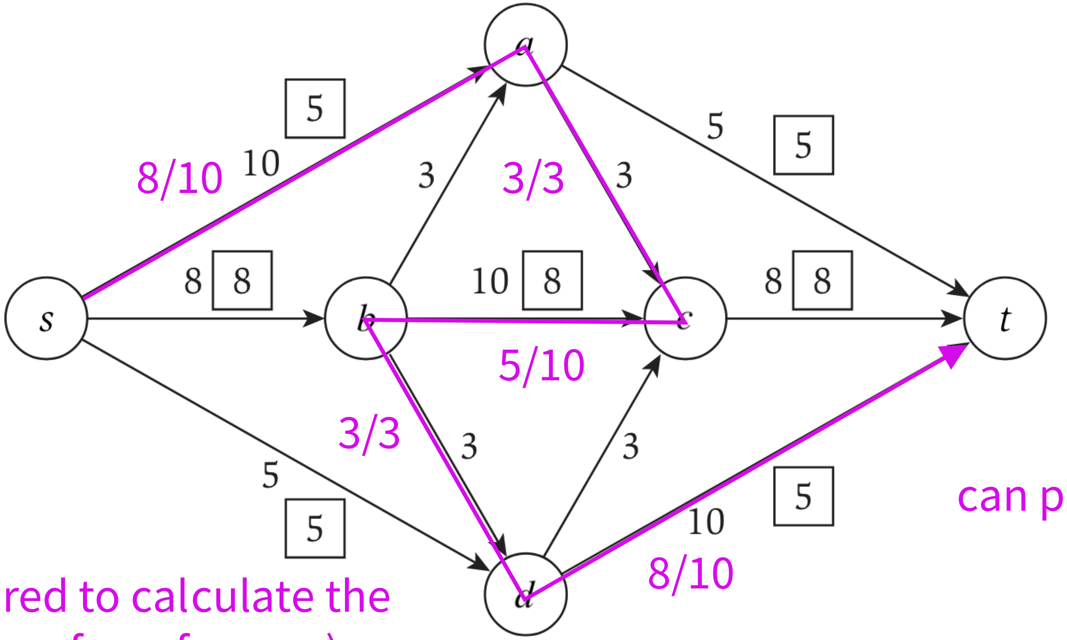


Solution

# Problem 1 – Short answer

In the network flow below, is the depicted flow a maximum flow?

Not maximum.




can push 3 more units

(you were not required to calculate the new flow, it's just here for reference)

# Problem 1 – Short answer

Recall Interval Scheduling: Given a collection of intervals and an integer $k$, determine if the collection contains at least $k$ nonoverlapping intervals.

i. Does Interval Scheduling $\leq_p$ Vertex Cover?

Solution

# Problem 1 – Short answer

Recall Interval Scheduling: Given a collection of intervals and an integer $k$, determine if the collection contains at least $k$ nonoverlapping intervals.

i. Does Interval Scheduling $\leq_p$ Vertex Cover?

Yes. Many possible reasons:

- Vertex Cover is NP-complete, in particular NP-hard, and Interval Scheduling is clearly in NP (the certificate is the list of $k$ nonoverlapping intervals). $A \leq_p B$ whenever $B$ is NP-hard and $A$ is in NP.
- Interval Scheduling is in P, as we solved it with a greedy algorithm earlier in this class. $A \leq_p B$ is always true when $A$ is in P.

# Problem 1 – Short answer

Recall Interval Scheduling: Given a collection of intervals and an integer $k$, determine if the collection contains at least $k$ nonoverlapping intervals.

ii. Does Independent Set $\leq_p$ Interval Scheduling ?

Solution

# Problem 1 – Short answer

Recall Interval Scheduling: Given a collection of intervals and an integer $k$, determine if the collection contains at least $k$ nonoverlapping intervals.

ii. Does Independent Set $\leq_p$ Interval Scheduling ?

Unknown. Because Independent Set is NP-complete and Interval Scheduling is in P, Independent Set $\leq_p$ Interval Scheduling would imply that an NP-complete problem is solvable in polynomial time, which is unknown.

# Problem 1 – Short answer

A greedy attempt at Set Cover is:

> **while** there exists an uncovered object **do**
>
> > choose a set that covers the most number of still-uncovered objects

Suppose you are given an instance where every set contains exactly 2 elements. Then this algorithm returns a set cover that is at most a factor 2 larger than the minimum.

Solution

# Problem 1 – Short answer

A greedy attempt at Set Cover is:

> **while** there exists an uncovered object **do**
>
> > choose a set that covers the most number of still-uncovered objects

Suppose you are given an instance where every set contains exactly 2 elements. Then this algorithm returns a set cover that is at most a factor 2 larger than the minimum.

True. If there are $n$ objects, the algorithm returns at most $n$ sets because every set chosen contains at least 1 new object. Since every object must be covered, and every set contains only 2 elements, we require $n/2$ sets. Thus the approximation ratio is 2.

Return to problem select

# Problem 2 – Dynamic programming

Given two strings, $s = s_1, \ldots, s_m$ with length $m$ and $t = t_1, \ldots, t_n$ with length $n$, find the length of their longest common subsequence.

Solution

# Problem 2 – Dynamic programming

Given two strings, $s = s_1, \ldots, s_m$ with length $m$ and $t = t_1, \ldots, t_n$ with length $n$, find the length of their longest common subsequence.

Let $\mathrm{OPT}(i, j)$ be the longest common subsequence between $s_1, \ldots, s_i$ and $t_1, \ldots, t_j$.

$$\mathrm{OPT}(i,j) = \begin{cases} 1 + \mathrm{OPT}(i-1, j-1) & \text{if } s_i = t_j \\ \max(\mathrm{OPT}(i-1,j), \mathrm{OPT}(i, j-1)) & \text{if } s_i \neq t_j \end{cases}$$

The base cases are $\mathrm{OPT}(i, 0) = \mathrm{OPT}(0, j) = 0$ for all $i$ and $j$.

Return to problem select

# Problem 3 – Network flows

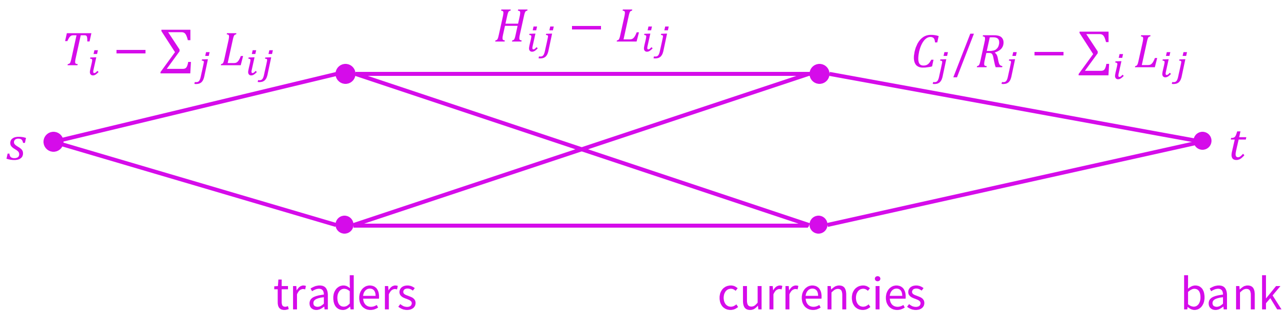
The bank has $C_j$ of currency $j$, and the exchange rate is $R_j$ of currency $j$ for every 1 Franc. Trader $i$ has $T_i$ Francs to convert and is willing to convert between $L_{ij}$ and $H_{ij}$ of their Francs to currency $j$. Determine if the bank can satisfy all requests, and if so, how to maximize the amount of Francs it collects.

Solution

# Problem 3 – Network flows

Determine if the bank can satisfy all requests, and if so, how to maximize the amount of Francs it collects.

First, give all traders their minimum request: check if $C_j/R_j \geq \sum_i L_{ij}$ for all $j$. Then,




Return to problem select

# Problem 4 – Linear programming

There are $k$ groups and $m_i$ voters in group $i$, of which $a_i$ are already voting for you. If you spend \$1000 advertising issue $j$, then $d_{ij}$ more voters in group $i$ will vote for you. Determine the minimum spending so that at least half of each group votes for you.

Solution

# Problem 4 – Linear programming

There are $k$ groups and $m_i$ voters in group $i$, of which $a_i$ are already voting for you. If you spend \$1000 advertising issue $j$, then $d_{ij}$ more voters in group $i$ will vote for you. Determine the minimum spending so that at least half of each group votes for you.

Let $x_j$ be the amount of money, in thousands, spent on issue $j$.

$$\begin{aligned} &\textbf{minimize } x_1 + \cdots + x_n \\ &\textbf{subject to } d_{i1}x_1 + \cdots + d_{in}x_n + a_i \geq \frac{m_i}{2} \qquad \text{for all } i \\ &\qquad\qquad\qquad\qquad\qquad\qquad\quad x_j \geq 0 \qquad \text{for all } j \end{aligned}$$

Solution

# Problem 4 – Linear programming

There are $k$ groups and $m_i$ voters in group $i$, of which $a_i$ are already voting for you. If you spend \$1000 advertising issue $j$, then $d_{ij}$ more voters in group $i$ will vote for you. Determine the minimum spending so that at least half of each group votes for you.

Let $x_j$ be the amount of money, in thousands, spent on issue $j$.

$$\begin{aligned}
\textbf{maximize} \quad & -x_1 - \cdots - x_n \\
\textbf{subject to} \quad & -d_{i1}x_1 - \cdots - d_{in}x_n \le a_i - \frac{m_i}{2} && \text{for all } i \\
& x_j \ge 0 && \text{for all } j
\end{aligned}$$

Return to problem select

# Problem 5 – Reduction

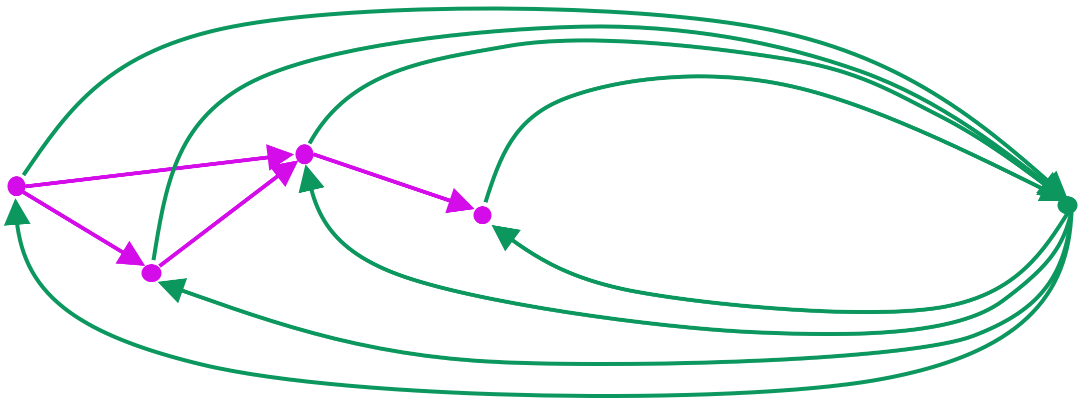
A Hamiltonian path/cycle is a path/cycle that visits every vertex exactly once. Suppose that HamiltonianPath is NP-hard. Show that HamiltonianCycle is NP-hard.

**Solution**

# Problem 5 – Reduction

A Hamiltonian path/cycle is a path/cycle that visits every vertex exactly once. Suppose that HamiltonianPath is NP-hard. Show that HamiltonianCycle is NP-hard.

We show HamiltonianPath $\leq_p$ HamiltonianCycle. Consider any input for HamiltonianPath. Create the following graph for HamiltonianCycle:

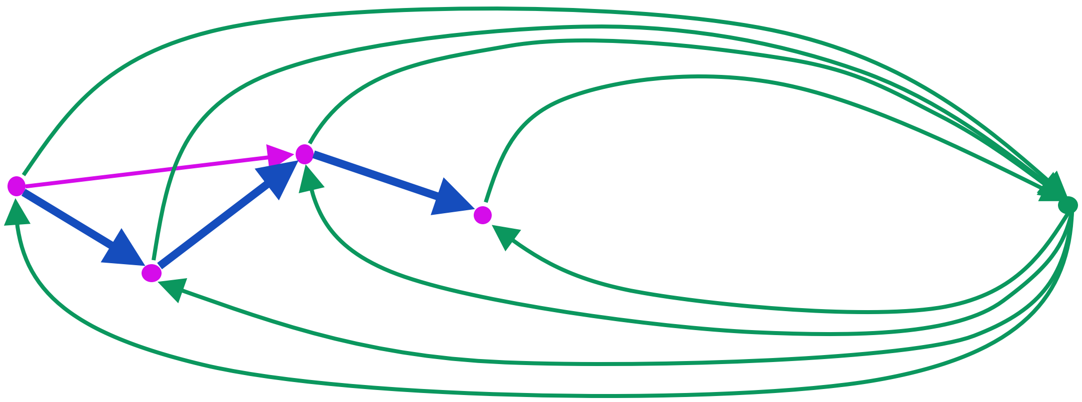


Solution

# Problem 5 – Reduction

A Hamiltonian path/cycle is a path/cycle that visits every vertex exactly once. Suppose that HamiltonianPath is NP-hard. Show that HamiltonianCycle is NP-hard.

To prove, convert certificate for HamPath to certificate for HamCycle.



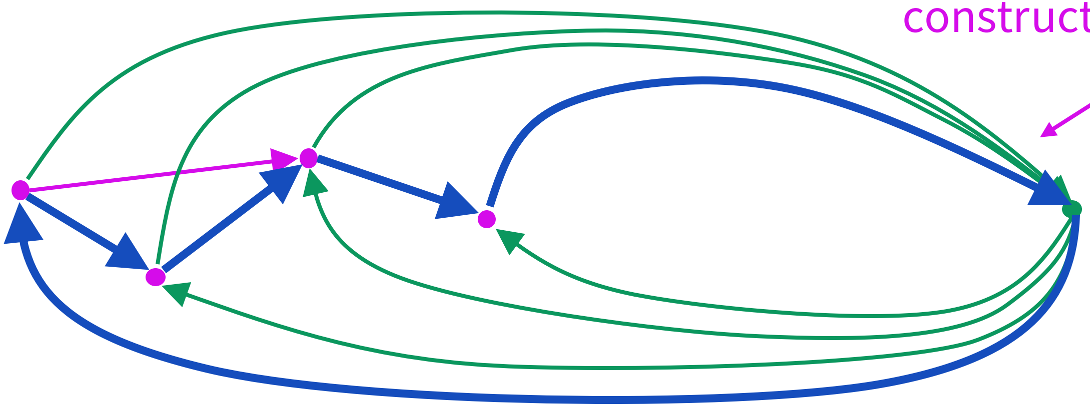
Solution

# Problem 5 – Reduction

A Hamiltonian path/cycle is a path/cycle that visits every vertex exactly once. Suppose that HamiltonianPath is NP-hard. Show that HamiltonianCycle is NP-hard.

To prove, convert certificate for HamPath to certificate for HamCycle.

two new blue edges exist by construction, form the cycle

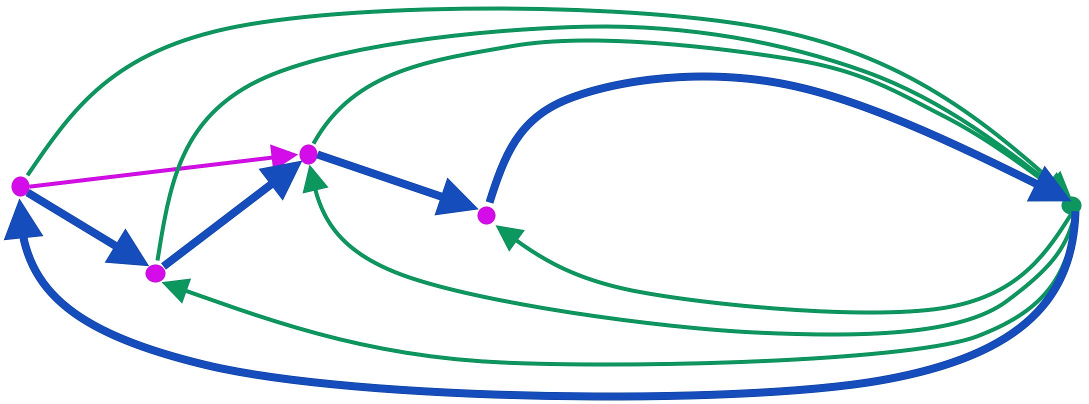


Solution

# Problem 5 – Reduction

A Hamiltonian path/cycle is a path/cycle that visits every vertex exactly once. Suppose that HamiltonianPath is NP-hard. Show that HamiltonianCycle is NP-hard.

To convert back, consider any Hamiltonian cycle in the graph we created.



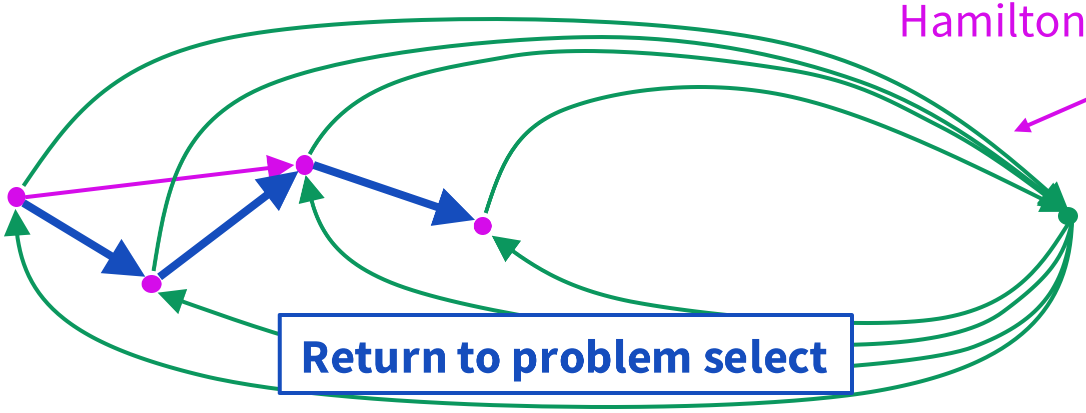
Solution

# Problem 5 – Reduction

A Hamiltonian path/cycle is a path/cycle that visits every vertex exactly once. Suppose that HamiltonianPath is NP-hard. Show that HamiltonianCycle is NP-hard.

To convert back, consider any Hamiltonian cycle in the graph we created.

removing these edges gives a Hamiltonian path



Return to problem select