• Do not open this quiz booklet until you are directed to do so. Read all the instructions first.

• You have 70 minutes to earn up to 26 points, the test is graded out of 25.

• Write your name and user id (as indicated on T-square) on the top of every page

• This booklet contains 3 questions on 6 pages, including this one. You can use the back of the pages for scratch work.

• Write your solutions in the space provided. If you run out of space, continue your answer on the back of the same sheet and make a notation on the front of the sheet.

• You may use a sheet with notes on both sides.

• You may use a calculator, but not any device with transmitting functions, especially ones that can access the wireless or the Internet.

• You may use any of the theorems/facts/lemmas/algorithms/home-works that we covered in class without re-proving them unless explicitly stated otherwise.

• If necessary make reasonable assumptions but please be sure to state them clearly

• When we ask you to give an algorithm in this test, 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, twice a problem’s point value is an indication of how many minutes to spend on it.

• Good luck!

<table>
<thead>
<tr>
<th>Problem</th>
<th>Title</th>
<th>Points</th>
<th>Parts</th>
<th>Grade</th>
<th>Initial of Grader</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Name &amp; ID on top of every page</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Formulating Linear Programs</td>
<td>9</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>The Ford-Fulkerson Algorithm</td>
<td>10</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Sports Elimination</td>
<td>6</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
0 ( /1 point) **Write your name and user id on top of every page.**

1. ( /9 points) Formulating Linear Programs

An oil refinery has two sources of crude oil:

- light crude that costs $35/barrel
- heavy crude that costs $30/barrel

The refinery produces gasoline, heating oil, and jet fuel from crude in the following amounts:

- each barrel of light crude produces 0.3 barrel of gasoline, 0.2 barrel of heating oil, and 0.3 barrel of jet fuel.
- each barrel of heavy crude produces 0.3 barrel of gasoline, 0.4 barrel of heating oil, and 0.2 barrels of jet fuel.

(note that these numbers don’t add up to 1 due to processing inefficiencies)

The refinery needs to supply:

- 900,000 barrels of gasoline
- 800,000 barrels of heating oil
- 500,000 barrels of jet fuel

and wants to minimize the total cost purchasing cost of the two types of crude oil.

Formulate the problem of finding the amounts of light and heavy crude to purchase so as to be able to meet its obligations at minimum cost.

You may use any types of variables / constraints / objectives, as long as they are linear.

Note that the amount of barrels bought can’t be negative (there is no existing storage to sell), and you can exceed the amount contracted.

**SOLUTION:**

Let $x_1$ and $x_2$ be the amount of light and heavy crude bought respectively.

The LP is:

\[
\begin{align*}
\text{min} & \quad 35x_1 + 30x_2 \\
\text{subject to:} & \quad 0.3x_1 + 0.3x_2 \geq 900000 \\
& \quad 0.2x_1 + 0.4x_2 \geq 800000 \\
& \quad 0.3x_1 + 0.2x_2 \geq 500000 \\
& \quad x_1, x_2 \geq 0
\end{align*}
\]
Alternate solution: we can create 6 more variables, $g_1$, $h_1$, $j_1$ for the amount of gasoline, heating oil, and jetfuel coming from heavy crude, and $g_2$, $h_2$, $j_2$ for the amount of gasoline, heating oil, and jetfuel coming from light crude.

Then we get the program

$$\min \quad 35x_1 + 30x_2$$
subject to:

$$g_1 + g_2 \geq 900000$$
$$h_1 + h_2 \geq 800000$$
$$j_1 + j_2 \geq 500000$$
$$g_1 \leq 0.3x_1$$
$$h_1 \leq 0.2x_1$$
$$j_1 \leq 0.3x_1$$
$$g_2 \leq 0.3x_2$$
$$h_2 \leq 0.4x_2$$
$$j_2 \leq 0.2x_2$$
$$x_1, x_2 \geq 0$$

There are some flexibilities with the $g_i$, $h_i$, and $j_i$ constraints: them being $\geq 0$ is not necessary, and their interactions with $x_i$ can be equality (e.g. $g_1 = 0.3x_1$).

**GRADING:**
1 point each for:

(a) two variables, one for each amount produced
(b) cost function
(c) variables being non-negative.

2 points each for the three constraints, −1 per issue in them.

If the programs omitted the idea that part of a product may be from heavy crude, part from light crude, −2 points.

If the requirement inequalities are set to =, (e.g. need to produce exactly this much gasoline), −1 point.
2. (10 points) The Ford-Fulkerson Algorithm

Consider the maximum flow on the following graph $G$ with capacities on edges.

![Graph diagram]

along with the flow $f$:

![Flow diagram]

which is what happens after routing 2 units of flow along the path

$s \rightarrow a \rightarrow c \rightarrow d \rightarrow b \rightarrow t$.

(a) (3 points) give the residual graph at this point ($G_f$):

SOLUTION:
GRADING:
−0.5 per edge difference.
Capacities on edges of residual graphs are ignored, only the direction/existence of edges are marked.

(b) (  /3) Your residual graph from part a) should admit the path

\[ s \rightarrow c \rightarrow a \rightarrow b \rightarrow d \rightarrow t. \]

Consider routing 2 units of flow along this path, in addition to the flow \( f \) specified at the start of this problem. Give the resulting flow, and the updated residual graph (separately).

SOLUTION:

flow:

residual graph:

GRADING:
−0.5 per edge difference, for up to 1.5 points per part.
Capacities on edges of residual graphs are ignored, only the direction/existence of edges are marked.

(c) (  /4 points) provide a maximum flow and a minimum cut on this graph. Make sure to specify the cut via a set of vertices.
SOLUTION:
A flow of value 7 is:

\[
\begin{align*}
S \rightarrow a \rightarrow b \rightarrow t \\
S \rightarrow c \rightarrow d \rightarrow t
\end{align*}
\]

A cut with capacity 7 is given by \( \{S, a, b\} \).
3. (6 points) Sports Elimination

Consider the following situation in a sport tournament involving $n$ teams, numbered 1...$n$:

(a) Each team already has $x_i$ wins.

(b) There is a list of $m$ games to be played between these teams, each having the value $\{i,j\}$ for some teams $1 \leq i < j \leq n$. For simplicity, assume that each pair of teams have at most 1 game left, so $|m| \leq O(n^2)$.

We want to check whether another team, which could possibly get up to $z$ wins, can still win the tournament. That is, we want to know whether there exists an outcome of the remaining $m$ games so that no team wins more than $z - x_i$ games.

Of course, this means we need to have $z \geq x_i$ for every $i$.

For example, if teams $a$, $b$, and $c$ have 2, 1, and 1 wins respectively, and there are three games left between $ab$, $bc$, and $ac$, then it’s possible for each of them to have at most 3 wins: team $a$ could lose all of its remaining games, and team $b$ could win the game between $bc$, resulting in 2, 3, and 2 wins respectively.

(a) (3 points) By using either maximum flow or matching in black-box manners (you do not need to give details of those algorithms), give an $O(n^{10})$ time algorithm to determine whether it’s possible for all teams to finish with at most $z$ wins.

**SOLUTION:**

Construct a network with vertices

- Super source $s$
- super sink $t$
- vertices for games $V_{games} = \{g_1 \ldots g_m\}$
- vertices for teams $V_{teams} = \{t_1 \ldots t_n\}$.

Then we add capacitated edges:

- For each game $g_j$, an edge from $s$ to $g_j$ with capacity 1.
- For each team $t_i$, an edge from $t_i$ to $t$ with capacity $z - x_i$.
- For each game $g_j$, involving teams $i_1$ and $i_2$, add edges from $g_j$ to $t_{i_1}$ and $t_{i_2}$ with infinite capacities.

A flow on this network corresponds ‘assigning’ a game to one of the teams involved. So as long as we can route all $m$ units of flow, there is a way to assign the games so that each team does not get too many wins.

**GRADING:**

There is also another solution that just adds edges between the team vertices. That solution also receives full points.
(b) (3 points) Show (using either the maxflow-mincut theorem, or Hall’s theorem) that should it be impossible for all teams to finish with at most \( z \) wins, there exists a subset of teams \( S_{teams} \) so that the number of games left to be played between them is more than

\[
\sum_{i \in S_{teams}} (z - x_i),
\]

aka. the total number of wins that these teams could have with none of them going over \( z \).

Continuing with the example from part a), it’s not possible for all teams to have at most 2 wins because the total number of wins between them is at least

\[
2 + 1 + 1 + 3 = 7
\]

which averages out to \( 2.333 \ldots > 2 \).

SOLUTION:
By the max-flow min-cut theorem, such an assignment is not possible only if there is a cut \( S \) in the graph specified in part a) whose capacity is less than \( m \).

The source vertex \( s \) is in \( S \) and let its intersection with the games / teams vertices be \( S_{games} \) and \( S_{teams} \) respectively.

As all edges leaving the games vertices have infinite capacity, the only edges leaving \( S_{games} \) must be to \( S_{teams} \). Aka. the games selected in \( S_{games} \) must only be between teams in \( S_{teams} \).

The size of this cut is

\[
\sum_{t_i \in S_{teams}} (z - x_i) + |V_{games} \setminus S_{games}| = \sum_{t_i \in S_{teams}} (z - x_i) + r - |S_{games}|.
\]

For this to be less than \( r \), we must have

\[
\sum_{t_i \in S_{teams}} (z - x_i) < |S_{games}|,
\]

which means that there are at least this many games left to be played between the teams in \( S_{teams} \).

GRADING:
1 point for giving the correct set from \( S \). 1 point for stating no edges leave \( S_{games} \). 1 point for deriving the final bound.