APMA 1210 Recitation 5

October 2022

1 Questions

1.1 Duality of Max Flow

Consider the general maximum flow problem, with a source s, sink t, and n intermediate vertices. Let all intermediate nodes have no demand, and let q_{ij} be the capacity constraint on edge $i \to j, i = s, 1, 2, ..., n, j = 1, 2, ..., n, t$. Write out the max flow problem as an LP using these parameters, then write down a minimization problem that is dual to the max flow problem. **Without using the result that max flow and min cut are dual problems,** how might you go about showing that the result of this minimization problem is the minimum cut on the graph? (What do the constraints and variables represent? What do these have to do with the min cut problem?) Please note that you **DO NOT** have to construct a formal proof of your minimization problem yielding the minimum cut - the goal is to build intuition about this dual relationship.

1.2 The Gold Mine Problem

Given a gold mine modeled as an m-by-n grid M, each grid cell in this mine is associated with a positive integer g_{ij} $(i = 1, ..., m, j = 1, ..., n)$ which is the amount of gold in tons that can be mined in that cell. Initially, the miner is at the first column but can be at any row. From the miner's current position, he can move only right or right up or right down from the current cell, i.e., the miner can move to the cell diagonally up towards the right or horizontally towards the right or diagonally down towards the right. As the miner moves across the grid from the first column to the n-th column, he mines all of the gold in each cell he passes through.

(a) Devise a dynamic programming strategy to figure out the maximum amount of gold the miner can collect. Make sure to identify what the vertices would be, what the partial ordering you need is, and what the set of recursive formulas are to compute the results. Watch out for edge cases!

(b) Use your strategy to solve the problem for the following 4-by-3 grid:

1	5	12
2	4	4
0	6	4
3	0	0

2 Solutions

2.1 Duality of Max Flow

In the general max flow problem, the flows out of s and into t must be equal in magnitude; we will assume that this is accounted for. Then the maximum flow can be identified as the sum of all flows out from the source s. Let x_{ij} be our decision variables, representing the flow from vertex i to vertex j, and let E represent the set of directed edges in the network. Then the max flow problem can be written as:

$$\begin{aligned}
\text{Maximize: } & z = \sum_{(s,i)\in E} x_{si} \\
\text{Subject to: } & \sum_{(j,i)\in E} x_{ji} - \sum_{(i,k)\in E} x_{ik} = 0, i = 1, ..., n \\
& x_{ij} \leq q_{ij}, i = s, 1, ..., n, j = 1, ..., n, t, (i,j) \in E \\
& x_{ij} \geq 0
\end{aligned}$$

Note that since the flow balance constraints sum to 0, they do not produce decision variables that will go into the objective function of the dual problem. Instead, the objective function will get exactly 1 decision variable for each capacity constraint, i.e. each edge. We will call these variables y_{ij}. However, the flow balance constraints do still produce decision variables - one for each vertex - that are important for the constraints, since they ensure that the matrices work out appropriately. We can call these variables v_i for $i = 1, ..., n$; these will be unconstrained, while the edge-based variables get the usual non-negativity constraints. Then for the edges coming from the source vertex, we will get constraints that look like $y_{sj} + v_j \geq 1$; for all other edges, the constraints will have a ≥ 0 form in alignment with the fact that these edges are not part of the objective function in the max flow problem. The final dual LP looks like this:

$$\begin{aligned}
\text{Minimize: } & z = \sum_{(i,j)\in E} q_{ij} y_{ij} \\
\text{Subject to: } & y_{sj} + v_j \geq 1, (s,j) \in E \\
& y_{ij} - v_i + v_j \geq 0, (i,j) \in E \\
& y_{it} - v_i \geq 0, (i,t) \in E \\
& y_{ij} \geq 0
\end{aligned}$$

Note that in this problem, there is exactly one constraint for each edge in the graph, and no additional constraints. When solved, this problem will set $y_{ij} = 1$ for any edge (i, j) such that $i \in S, j \in T$, and $v_i = 1$ for any vertex $i \in S$. The first set of constraints ensures that if $j \in T$, the edge (s, j) must be in the cut; the second set of constraints ensures that if $i \in S, j \in T$, the edge (i, j) must be in the cut; and the third set of constraints ensures that if $i \in S$, the edge (i, t) must be in the cut (where all of these are contingent on the existence of their associated edges).

2.2 The Gold Mine Problem

(a) We'll define the dynamic programming result on a cell-wise basis, where $f(i, j)$ will be the maximum amount of gold that the miner can have collected upon reaching cell $M_{i,j}$. We can also talk about a partial ordering on the cells, where the relation is $M_{i,j} > M_{k,l}$ if there exists a sequence of steps by which the miner can get from $M_{k,l}$ to $M_{i,j}$. Moreover, for a given cell $M_{i,j}$, it can be reached from $M_{i-1,j-1}, M_{i,j-1}$, and $M_{i+1,j-1}$ unless $i = 1$ (in which case there is no $M_{i-1,j-1}$ or $i = m$ (in which case there is no $M_{i+1,j-1}$. If you would prefer to think of this as a network, consider a source vertex s from which the miner starts where he can step to any cell in the first column, and a sink vertex t to which the miner can step from any cell in the last column. All other steps within the cell network must obey the provided rules, and each edge adds the weight g_{ij} associated with the origin cell (so edges from s will add no weight). In recursive form, we are solving $t = \max\{f(i, n)\}$ over $i = 1, ..., m$, where we calculate $f(i, j) = g_{ij} + \max\{f(i-1, j-1), f(i, j-1), f(i+1, j-1)\}$ for $i = 2, ..., m-1$, $f(1, j) = g_{1j} + \max\{f(1, j-1), f(2, j-1)\}$, and $f(m, j) = g_{mj} + \max\{f(m-1, j-1), f(m, j-1)\}$. The initial conditions for this setup are $f(i, 1) = g_{i1}$, the amount of gold found at position $M_{i,1}$.

(b) There are three columns, so four stages to the dynamic programming process; the first stage is the step from s, and the last stage is the step to t. Whenever a cell can't reach some other cell, we'll assign a max gold of 0 to that route. Ties will be broken at random.

Stage 4:

Leading cell	t	Max gold	Best route
$M_{1,3}$	12	12	$M_{1,3} \to t$
$M_{2,3}$	4	4	$M_{2,3} \to t$
$M_{3,3}$	4	4	$M_{3,3} \to t$
$M_{4,3}$	0	0	$M_{4,3} \to t$

Stage 3:

Leading cell	$M_{1,3} \to t$	$M_{2,3} \to t$	$M_{3,3} \to t$	$M_{4,3} \to t$	Max gold	Best route
$M_{1,2}$	17	9	0	0	17	$M_{1,2} \to M_{1,3}$
$M_{2,2}$	16	8	8	0	16	$M_{2,2} \to M_{1,3}$
$M_{3,2}$	0	10	10	6	10	$M_{3,2} \to M_{3,3}$
$M_{4,2}$	0	0	4	0	4	$M_{4,2} \to M_{3,3}$

Stage 2:

Leading cell	$M_{1,2} \to t$	$M_{2,2} \to t$	$M_{3,2} \to t$	$M_{4,2} \to t$	Max gold	Best route
$M_{1,1}$	18	17	0	0	18	$M_{1,1} \to M_{1,2}$
$M_{2,1}$	19	18	12	0	19	$M_{2,1} \to M_{1,2}$
$M_{3,1}$	0	16	10	4	16	$M_{3,1} \to M_{2,2}$
$M_{4,1}$	0	0	13	7	13	$M_{4,1} \to M_{3,2}$

Stage 1:

Leading cell	$M_{1,3} \to t$	$M_{2,3} \to t$	$M_{3,3} \to t$	$M_{4,3} \to t$	Max gold	Best route
s	18	19	16	13	19	$s \to M_{2,1}$

So our final solution is the path $s \to M_{2,1} \to M_{1,2} \to M_{1,3} \to t$, i.e. the grid path $M_{2,1} \to M_{1,2} \to M_{1,3}$, for a total of 19 tons of gold mined.