Question 1. Network Flow [25 MARKS]

Let $N = (V, E)$ be a network with a set of nodes V and a set of directed edges E with integer capacities. Let F be the maximum flow value in N from node s to node t. Let P be a simple path from s to t in N consisting of k edges, and let N^+ be the network obtained from N by adding 1 to the capacity of every edge in P.

(a) (5 marks) TRUE/FALSE: The maximum flow value in N^+ is at least $F+1$. (If this is true, argue why. If this is false, provide a counter-example.)

(b) (5 marks) Prove that the maximum flow value in N^+ may not be exactly $F+1$.

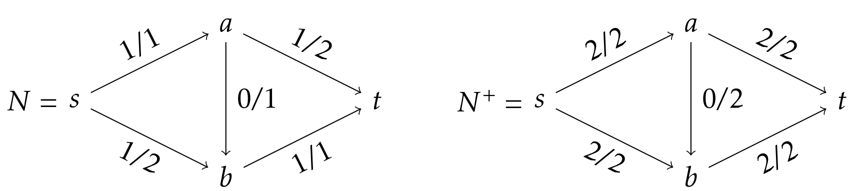
(c) (10 marks) What is a tight upper bound on the maximum flow value in N^+ in terms of F and k? You must prove that your upper bound always holds. To show that it is tight, you must produce a network in which the maximum flow value in N^+ exactly matches your upper bound.

(d) (5 marks) Let N^- be the network obtained from N by subtracting 1 from the capacity of every edge in P (assume each edge in P had capacity at least 1 in N). TRUE/FALSE: The maximum flow value in N^- is always strictly less than F. (If this is true, argue why. If this is false, provide a counter-example.)

Sample Solution:

(a) TRUE. This is because one can simply take a maximum flow in N with value F, and augment it along P by one unit in N^+ resulting in a flow of value $F+1$.

(b) Consider the following network.

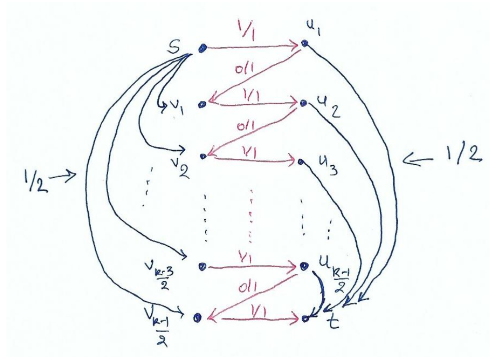


In network N, the maximum flow is $F = 2$. Network N^+ is obtained by increasing the capacity of every edge on path $s \to a \to b \to t$ by 1. The maximum flow in N^+ is $4 > F+1$.

(c) We show that the tight bound is $F + \lfloor \frac{k+1}{2} \rfloor$.

Upper bound: Take any min-cut (A, B) of network N. By max-flow-min-cut theorem, it has capacity F. We are interested in edges of P that go from A to B (since the increase in capacity of those edges can increase the cut capacity). Note that for every $A \to B$ edge in P, we must have a subsequent $B \to A$ edge before we can have another $A \to B$ edge. Hence, P has at most $\lfloor \frac{k+1}{2} \rfloor$ edges going $A \to B$. Hence, when we increase the capacity of each edge in P by 1, the capacity of cut (A, B) increases by at most $\lfloor \frac{k+1}{2} \rfloor$. Hence, N^+ has a cut of capacity at most $F + \lfloor \frac{k+1}{2} \rfloor$, so by max-flow-min-cut, its max flow is at most $F + \lfloor \frac{k+1}{2} \rfloor$.

Lower bound: Consider the network N in the image below. The edges in red are the edges in path P. The idea is that every edge going from left to right is a bottleneck (it has capacity 1 right now, but if its capacity increases to 2, it can add a unit flow). Indeed, there are $\lfloor \frac{k+1}{2} \rfloor$ edges going from left to right, and increasing all their capacities by 1 increases the max flow precisely by $\lfloor \frac{k+1}{2} \rfloor$.



(d) TRUE. Take any min-cut (A,B) in network N. By max-flow-min-cut, it has capacity F. Since P goes from $s \in A$ to $t \in B$, it has at least one edge going from A to B. Since the capacity of this edge is decreasing by 1, the capacity of (A,B) in N^- is at most $F-1$. Hence, min cut (and thus max flow) in N^- is at most $F-1$.

Question 2. Integer Linear Programming [20 MARKS]

Recall that given an undirected graph $G=(V,E)$, we say that G is k-colourable if we can assign a colour $c(v) \in \{1,\dots,k\}$ to every vertex $v \in V$ such that no two adjacent vertices have the same colour (i.e. $c(u) \neq c(v)$ for all $(u,v) \in E$). The minimum vertex coloring problem, MINCOLOUR, is defined as follows.

- *Input:* An undirected graph $G=(V,E)$.
- *Output:* The smallest positive integer k such that G is k-colourable.

We wish to write a *binary integer program* (with each variable taking value in $\{0,1\}$) to solve MINCOLOUR.

(a) (5 marks) Argue that the solution of MINCOLOUR is never greater than n (the number of nodes in G).

(b) (15 marks) Provide a binary integer linear program to solve MINCOLOUR. Clearly explain why your objective minimizes the number of colours used and why your constraints ensure a valid colouring. For full marks, your program should use at most $O(n^2)$ binary variables, where n is the number of nodes in G.

Sample Solution:

(a) Giving each node of G a different colour is certainly a valid colouring and uses n colours. Hence, the minimum number of colours needed is at most n.

(b) We will keep n colours at our disposal (by part (a), these are sufficient). For each possible colour $k \in \{1,\dots,n\}$, we will use a binary variable y_k to indicate whether colour k is used anywhere. For each node $i \in V$ and each possible colour $k \in \{1,\dots,n\}$, we will use a binary variable $x_{i,k}$ to denote whether node i is given colour k. Given this, the binary IP is as follows. The role of the objective function and each constraint is explained in comments next to it.

Minimize $\sum_{k=1}^{n} y_k$ #Minimize number of colours used

Such that

$\sum_{k=1}^{n} x_{i,k} = 1, \forall i \in V$ #Each node gets one colour

$x_{i,k} \leqslant y_k, \forall i \in V, k \in \{1,\dots,n\}$ #Only colours that are used can be assigned

$x_{i,k} + x_{j,k} \leqslant 1, \forall (i,j) \in E, k \in \{1,\dots,n\}$ #Adjacent nodes cannot get the same colour

$x_{i,k}, y_k \in \{0,1\}, \forall i \in V, k \in \{1,\dots,n\}$ #Binary variables

Question 3. P/NP/coNP [10 MARKS]

(a) (2.5 marks) Does $P = NP$ imply $NP = coNP$? Justify your answer.

(b) (2.5 marks) Does $NP = coNP$ imply $P = NP$? Justify your answer.

(c) (2.5 marks) Given an undirected graph G and a positive integer k, the k-COLOUR problem asks whether G is k-colourable (i.e. whether it has a valid colouring using *at most* k colours). Show that given a polynomial-time algorithm for k-COLOUR, one can solve MINCOLOUR in polynomial time.

(d) (2.5 marks) Show that given a polynomial-time algorithm for MINCOLOUR, one can solve k-COLOUR in polynomial time.

Sample Solution:

(a) Yes. The complement of a problem in P is also in P. Hence, if P=NP, then the complement of every problem in NP (i.e. every coNP problem) is also in P, i.e., P=NP=coNP.

(b) Not necessarily. NP=coNP simply means that for each problem in NP=coNP, both YES and NO answers can be verified in polynomial time. This does not trivially imply that these problems can also be *solved* in polynomial time.

(c) Given a graph k, we can solve k-COLOUR for each $k \in \{1, \ldots, n\}$. (By Q2 part (a), we know that n is sufficient.) The smallest k for which G is k-colourable is then the answer of MINCOLOUR.

(d) Given a graph G and an integer k, we can solve MINCOLOUR to obtain k^*, and then check if $k^* \leqslant k$.