The events of 8 September prompted Foch to draft the later legendary signal: “My centre is giving way, my right is in retreat, situation excellent. I attack.” It was probably never sent.

– The first world war, John Keegan.
1. **Slack form**

**[10 Points]**

Let $L$ be a linear program given in slack form, with $n$ nonbasic variables $N$, and $m$ basic variables $B$. Let $N'$ and $B'$ be a different partition of $N \cup B$, such that $|N' \cup B'| = |N \cup B|$. Show a polynomial time algorithm that computes an equivalent slack form that has $N'$ as the nonbasic variables and $B'$ as the basic variables. How fast is your algorithm?

2. **Tedious Computations**

**[20 Points]**

Provide detailed solutions for the following problems, showing each pivoting stage separately.

(a) **[5 Points]**

maximize $6x_1 + 8x_2 + 5x_3 + 9x_4$

subject to

$2x_1 + x_2 + x_3 + 3x_4 \leq 5$

$x_1 + 3x_2 + x_3 + 2x_4 \leq 3$

$x_1, x_2, x_3, x_4 \geq 0.$

(b) **[5 Points]**

maximize $2x_1 + x_2$

subject to

$2x_1 + x_2 \leq 4$

$2x_1 + 3x_2 \leq 3$

$4x_1 + x_2 \leq 5$

$x_1 + 5x_2 \leq 1$

$x_1, x_2 \geq 0.$

(c) **[5 Points]**

maximize $6x_1 + 8x_2 + 5x_3 + 9x_4$

subject to

$x_1 + x_2 + x_3 + x_4 = 1$

$x_1, x_2, x_3, x_4 \geq 0.$

(d) **[5 Points]**

minimize $x_{12} + 8x_{13} + 9x_{14} + 2x_{23} + 7x_{24} + 3x_{34}$

subject to

$x_{12} + x_{13} + x_{14} \geq 1$

$-x_{12} + x_{23} + x_{24} = 0$

$-x_{13} - x_{23} + x_{34} = 0$

$x_{14} + x_{24} + x_{34} \leq 1$

$x_{12}, x_{13}, \ldots, x_{34} \geq 0.$

3. **[10 Points] Linear Programming for Graph**

(a) **[3 Points]** Given a weighted, directed graph $G = (V, E)$, with weight function $w : E \rightarrow \mathbb{R}$ mapping edges to real-valued weights, a source vertex $s$, and a destination vertex $t$. Show how to compute the value $d[t]$, which is the weight of a shortest path from $s$ to $t$, by linear programming.

(b) **[4 Points]**

Given a graph $G$ as in (a), write a linear program to compute $d[v]$, which is the shortest-path weight from $s$ to $v$, for each vertex $v \in V$. 


(c) [4 Points]
In the minimum-cost multicommodity-flow problem, we are given a directed graph \( G = (V, E) \), in which each edge \((u, v) \in E\) has a nonnegative capacity \(c(u, v) \geq 0\) and a cost \(\alpha(u, v)\). As in the multicommodity-flow problem (Chapter 29.2, CLRS), we are given \(k\) different commodities, \(K_1, K_2, \ldots, K_k\), where commodity \(i\) is specified by the triple \((s_i, t_i, d_i)\). Here \(s_i\) is the source of commodity \(i\), \(t_i\) is the sink of commodity \(i\), and \(d_i\) is the demand, which is the desired flow value for commodity \(i\) from \(s_i\) to \(t_i\). We define a flow for commodity \(i\), denoted by \(f_i\), (so that \(f_i(u, v)\) is the flow of commodity \(i\) from vertex \(u\) to vertex \(v\)) to be a real-valued function that satisfies the flow-conservation, skew-symmetry, and capacity constraints. We now define \(f(u, v)\), the aggregate flow, to be sum of the various commodity flows, so that \(f(u, v) = \sum_{i=1}^{k} f_i(u, v)\). The aggregate flow on edge \((u, v)\) must be no more than the capacity of edge \((u, v)\).

The cost of a flow is \(\sum_{u,v \in V} c(u, v)\), and the goal is to find the feasible flow of minimum cost. Express this problem as a linear program.

4. Find \(k\)th smallest number.
   [20 Points]
This question asks you to design and analyze a randomized incremental algorithm to select the \(k\)th smallest element from a given set of \(n\) elements (from a universe with a linear order).

In an incremental algorithm, the input consists of a sequence of elements \(x_1, x_2, \ldots, x_n\). After any prefix \(x_1, \ldots, x_{i-1}\) has been considered, the algorithm has computed the \(k\)th smallest element in \(x_1, \ldots, x_{i-1}\) (which is undefined if \(i \leq k\), or if appropriate, some other invariant from which the \(k\)th smallest element could be determined). This invariant is updated as the next element \(x_i\) is considered.

Any incremental algorithm can be randomized by first randomly permuting the input sequence, with each permutation equally likely.

(a) [5 Points] Describe an incremental algorithm for computing the \(k\)th smallest element.

(b) [5 Points] How many comparisons does your algorithm perform in the worst case?

(c) [10 Points] What is the expected number (over all permutations) of comparisons performed by the randomized version of your algorithm? (Hint: When considering \(x_i\), what is the probability that \(x_i\) is smaller than the \(k\)th smallest so far?) You should aim for a bound of at most \(n + O(k \log(n/k))\). Revise (a) if necessary in order to achieve this.

5. Adapt min-cut
   [20 Points]
Consider adapting the min-cut algorithm to the problem of finding an \(s-t\) min-cut in an undirected graph. In this problem, we are given an undirected graph \(G\) together with two distinguished vertices \(s\) and \(t\). An \(s-t\) min-cut is a set of edges whose removal disconnects \(s\) from \(t\); we seek an edge set of minimum cardinality. As the algorithm proceeds, the vertex \(s\) may get amalgamated into a new vertex as the result of an edge being contracted; we call this vertex the \(s\)-vertex (initially \(s\) itself). Similarly, we have a \(t\)-vertex. As we run the contraction algorithm, we ensure that we never contract an edge between the \(s\)-vertex and the \(t\)-vertex.

(a) [10 Points] Show that there are graphs in which the probability that this algorithm finds an \(s-t\) min-cut is exponentially small.

(b) [10 Points] How large can the number of \(s-t\) min-cuts in an instance be?
6. MAJORITY TREE

[20 Points]

Consider a uniform rooted tree of height $h$ (every leaf is at distance $h$ from the root). The root, as well as any internal node, has 3 children. Each leaf has a boolean value associated with it. Each internal node returns the value returned by the majority of its children. The evaluation problem consists of determining the value of the root; at each step, an algorithm can choose one leaf whose value it wishes to read.

(a) Show that for any deterministic algorithm, there is an instance (a set of boolean values for the leaves) that forces it to read all $n = 3^h$ leaves. (hint: Consider an adversary argument, where you provide the algorithm with the minimal amount of information as it request bits from you. In particular, one can devise such an adversary algorithm.).

(b) Consider the recursive randomized algorithm that evaluates two subtrees of the root chosen at random. If the values returned disagree, it proceeds to evaluate the third sub-tree. If they agree, it returns the value they agree on. Show the expected number of leaves read by the algorithm on any instance is at most $n^{0.9}$.