Warm-up

Problem 1. Explain why every undirected graph on n vertices has exactly $2^{n-1} - 1$ distinct cuts (not all minimum cuts).

Problem 2. Solve the two recurrence relations for the Karger–Stein algorithm (time and probability).

Problem 3. Analyse/describe what would happen to the time complexity and success probability if we only did 1 run (instead of 2) for the Karger–Stein algorithm. What if we did 3 runs instead of 2?

Problem solving

Problem 4. Show how to, given as input a (multi)graph $G = (V, E)$ in either the adjacency matrix or adjacency list representation, to sample an edge uniformly at random in time $O(n)$, where $n = |V|$.

Problem 5. Consider the following generalisation of MIN-CUT:

> k-MIN-CUT: Given an (undirected) connected graph $G = (V, E)$ on n vertices and m edges ad an integer $k \geq 2$, output a k-cut $(A_1, \ldots, A_k)$ (partition of V) *minimising* the number $c_k(A_1, \ldots, A_k)$ of edges between the different connected components $A_1, \ldots, A_k$.

a) Adapt (the basic version of) Karger's algorithm to solve this problem.

b) Analyse the success probability and running time.

c) Provide a bound on the maximum number of k-Min-Cuts a graph can have.

Problem 6. Consider the following algorithm:

1. Draw, independently for every edge $e \in E$, a weight w_e in $[0, 1]$ uniformly at random.
2. Build the MST of $G = (V, E, w)$ (according to these weights)
3. Remove the heaviest edge of the MST, and let A, B be the resulting 2 components.
4. Return (A, B) as cut.

Show that this is equivalent to Karger's algorithm (the "basic" version). Deduce how to implement this algorithm in time $O(m \log m)$.

Advanced

Problem 7. Prove that (a suitable modification of) Karger's algorithm still works for weighted graphs (with non-negative weights). Do the same for the Karger–Stein algorithm.