In general: If our prior model (before running the test) is that $\Pr(E) \geq 2^i/(2^i + 1)$ and if the test returns that the identity is correct (event B), then

$$\Pr(E \mid B) \geq \frac{\dfrac{2^i}{2^i+1}}{\dfrac{2^i}{2^i+1} + \dfrac{1}{2}\dfrac{1}{2^i+1}} = \frac{2^{i+1}}{2^{i+1}+1} = 1 - \frac{1}{2^i+1}.$$

Thus, if all 100 calls to the matrix identity test return that the identity is correct, our confidence in the correctness of this identity is at least $1 - 1/(2^{100} + 1)$.

1.4. Application: A Randomized Min-Cut Algorithm

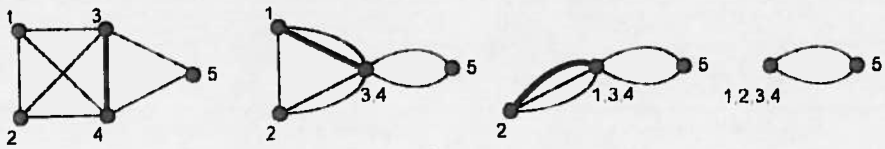
A *cut-set* in a graph is a set of edges whose removal breaks the graph into two or more connected components. Given a graph $G = (V, E)$ with n vertices, the minimum cut – or *min-cut* – problem is to find a minimum cardinality cut-set in G. Minimum cut problems arise in many contexts, including the study of network reliability. In the case where nodes correspond to machines in the network and edges correspond to connections between machines, the min-cut is the smallest number of edges that can fail before some pair of machines cannot communicate. Minimum cuts also arise in clustering problems. For example, if nodes represent Web pages (or any documents in a hypertext-based system) and two nodes have an edge between them if the corresponding nodes have a hyperlink between them, then small cuts divide the graph into clusters of documents with few links between clusters. Documents in different clusters are likely to be unrelated.

We shall proceed by making use of the definitions and techniques presented so far in order to analyze a simple randomized algorithm for the min-cut problem. The main operation in the algorithm is *edge contraction*. In contracting an edge $\{u, v\}$ we merge the two vertices u and v into one vertex, eliminate all edges connecting u and v, and retain all other edges in the graph. The new graph may have parallel edges but no self-loops. Examples appear in Figure 1.1, where in each step the dark edge is being contracted.

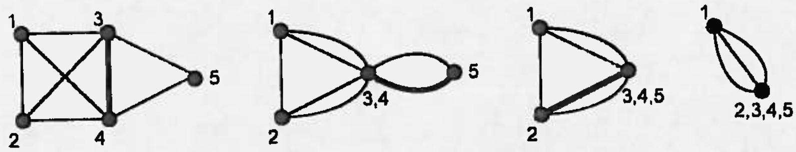
The algorithm consists of $n - 2$ iterations. In each iteration, the algorithm picks an edge from the existing edges in the graph and contracts that edge. There are many possible ways one could choose the edge at each step. Our randomized algorithm chooses the edge uniformly at random from the remaining edges.

Each iteration reduces the number of vertices in the graph by one. After $n - 2$ iterations, the graph consists of two vertices. The algorithm outputs the set of edges connecting the two remaining vertices.

It is easy to verify that any cut-set of a graph in an intermediate iteration of the algorithm is also a cut-set of the original graph. On the other hand, not every cut-set of the original graph is a cut-set of a graph in an intermediate iteration, since some edges of the cut-set may have been contracted in previous iterations. As a result, the output of the algorithm is always a cut-set of the original graph but not necessarily the minimum cardinality cut-set (see Figure 1.1).



(a) A successful run of min-cut.



(b) An unsuccessful run of min-cut.

Figure 1.1: An example of two executions of min-cut in a graph with minimum cut-set of size 2.

We now establish a lower bound on the probability that the algorithm returns a correct output.

Theorem 1.8: *The algorithm outputs a min-cut set with probability at least* $2/n(n-1)$.

Proof: Let k be the size of the min-cut set of G. The graph may have several cut-sets of minimum size. We compute the probability of finding one specific such set C.

Since C is a cut-set in the graph, removal of the set C partitions the set of vertices into two sets, S and $V - S$, such that there are no edges connecting vertices in S to vertices in $V - S$. Assume that, throughout an execution of the algorithm, we contract only edges that connect two vertices in S or two vertices in $V - S$, but not edges in C. In that case, all the edges eliminated throughout the execution will be edges connecting vertices in S or vertices in $V - S$, and after $n - 2$ iterations the algorithm returns a graph with two vertices connected by the edges in C. We may therefore conclude that, if the algorithm never chooses an edge of C in its $n - 2$ iterations, then the algorithm returns C as the minimum cut-set.

This argument gives some intuition for why we choose the edge at each iteration uniformly at random from the remaining existing edges. If the size of the cut C is small and if the algorithm chooses the edge uniformly at each step, then the probability that the algorithm chooses an edge of C is small – at least when the number of edges remaining is large compared to C.

Let E_i be the event that the edge contracted in iteration i is not in C, and let $F_i = \bigcap_{j=1}^{i} E_j$ be the event that no edge of C was contracted in the first i iterations. We need to compute $\Pr(F_{n-2})$.

We start by computing $\Pr(E_1) = \Pr(F_1)$. Since the minimum cut-set has k edges, all vertices in the graph must have degree k or larger. If each vertex is adjacent to at least k edges, then the graph must have at least $nk/2$ edges. The first contracted edge is chosen uniformly at random from the set of all edges. Since there are at least $nk/2$ edges in the graph and since C has k edges, the probability that we do not choose an edge of C in the first iteration is given by

$$\Pr(E_1) = \Pr(F_1) \geq 1 - \frac{2k}{nk} = 1 - \frac{2}{n}.$$

Let us suppose that the first contraction did not eliminate an edge of C. In other words, we condition on the event F_1. Then, after the first iteration, we are left with an $(n-1)$-node graph with minimum cut-set of size k. Again, the degree of each vertex in the graph must be at least k, and the graph must have at least $k(n-1)/2$ edges. Thus,

$$\Pr(E_2 \mid F_1) \geq 1 - \frac{k}{k(n-1)/2} = 1 - \frac{2}{n-1}.$$

Similarly,

$$\Pr(E_i \mid F_{i-1}) \geq 1 - \frac{k}{k(n-i+1)/2} = 1 - \frac{2}{n-i+1}.$$

To compute $\Pr(F_{n-2})$, we use

$$\begin{aligned}
\Pr(F_{n-2}) &= \Pr(E_{n-2} \cap F_{n-3}) = \Pr(E_{n-2} \mid F_{n-3}) \cdot \Pr(F_{n-3}) \\
&= \Pr(E_{n-2} \mid F_{n-3}) \cdot \Pr(E_{n-3} \mid F_{n-4}) \cdots \Pr(E_2 \mid F_1) \cdot \Pr(F_1) \\
&\geq \prod_{i=1}^{n-2}\left(1 - \frac{2}{n-i+1}\right) = \prod_{i=1}^{n-2}\left(\frac{n-i-1}{n-i+1}\right) \\
&= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\cdots\left(\frac{4}{6}\right)\left(\frac{3}{5}\right)\left(\frac{2}{4}\right)\left(\frac{1}{3}\right) \\
&= \frac{2}{n(n-1)}.
\end{aligned}$$

■

Since the algorithm has a one-sided error, we can reduce the error probability by repeating the algorithm. Assume that we run the randomized min-cut algorithm $n(n-1)\ln n$ times and output the minimum size cut-set found in all the iterations. The probability that the output is not a min-cut set is bounded by

$$\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)\ln n} \leq e^{-2\ln n} = \frac{1}{n^2}.$$

In the first inequality we have used the fact that $1 - x \leq e^{-x}$.

1.5. Exercises

Exercise 1.1: We flip a fair coin ten times. Find the probability of the following events.

(a) The number of heads and the number of tails are equal.
(b) There are more heads than tails.
(c) The ith flip and the $(11-i)$th flip are the same for $i = 1, \ldots, 5$.
(d) We flip at least four consecutive heads.