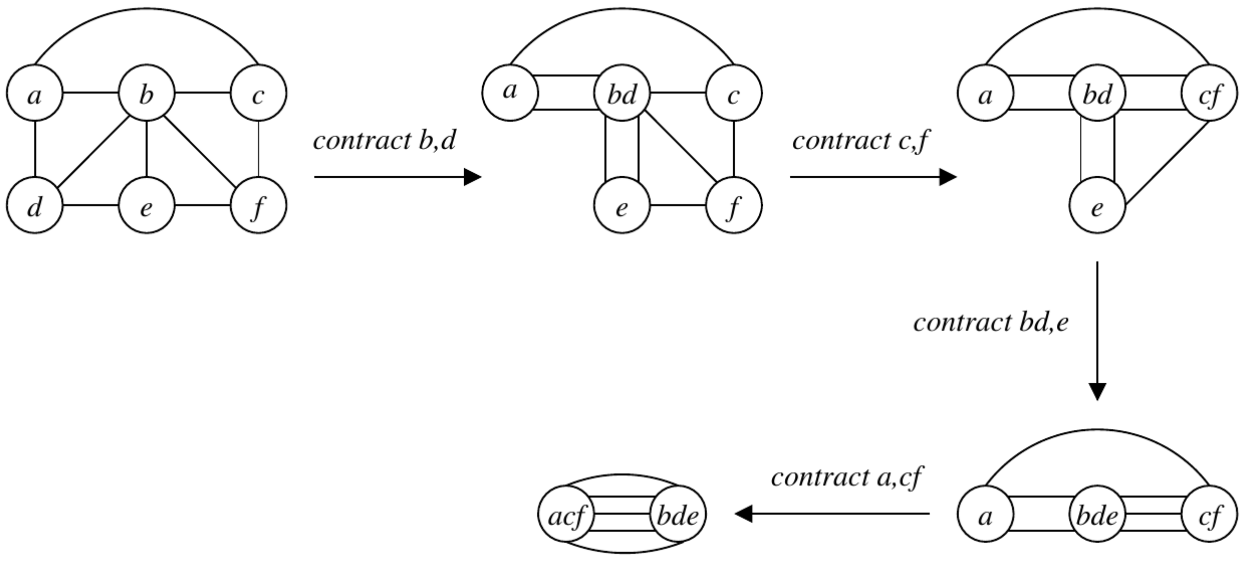
Karger's Min-Cut Algorithm



There exists a vertex of degree ≤ $2m/n$, so $c^* \leq 2m/n$.

First choice: $$1 - \frac{c^*}{m} \geqslant 1 - \frac{2m/n}{m} = 1 - \frac{2}{n} = \frac{n-2}{n}.$$

$(i + 1)$-st choice: $$\geqslant 1 - \frac{2}{n-i} = \frac{n-i-2}{n-i}.$$

Success probability for one run:

$$\geqslant \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\left(\frac{n-5}{n-3}\right)\cdots\left(\frac{2}{4}\right)\left(\frac{1}{3}\right) = \frac{2}{n(n-1)} \approx \frac{2}{n^2}.$$

Failure probability for $n^2/2$ independent runs:

$$\leqslant \left(1 - \frac{2}{n^2}\right)^{n^2/2} \leqslant \frac{1}{e}$$

Karger–Stein Algorithm

$$\geqslant \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\cdots\left(\frac{l-1}{l+1}\right) = \frac{l(l-1)}{n(n-1)} \approx \frac{l^2}{n^2}.$$

```
KargerStein(G)
   If the number of vertices of $G$ is small,
      return a minimum cut by exhaustive search,
   else do the following:
      Contract $G$ to $n/\sqrt{2}$ vertices to get a multigraph $G_1$.
      Compute $S_1, T_1 =$ KargerStein($G_1$).
      Contract $G$ to $n/\sqrt{2}$ vertices to get a multigraph $G_2$.
      Compute $S_2, T_2 =$ KargerStein($G_2$).
      Return the better of the two cuts $S_1, T_1$ and $S_2, T_2$.
```

Success probability: $$P(n) \geqslant 1 - \left(1 - \frac{1}{2}P\left(\frac{n}{\sqrt{2}}\right)\right)^2.$$

Running time: $$T(n) = 2T\left(\frac{n}{\sqrt{2}}\right) + \mathrm{O}(n^2).$$

Bloom Filter

Suppose we want to store 3-digit integers in a bloom filter. Take a bit array A of size $s = 16$. For an input $x = (x_2x_1x_0)_{10}$ (in base 10), we compute the hash values $h_0(x) = (3^{x_0} \pmod{17}) - 1$, $h_1(x) = (5^{x_1} \pmod{17}) - 1$, and $h_2(x) = (7^{x_2} \pmod{17}) - 1$. Initially, the filter does not contain any element, so the bit array A is:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Now, we insert 825 in the filter. We have $h_0(825) = 4$, $h_1(825) = 7$ and $h_2(825) = 15$. So the insertion of 825 updates A as:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1

Then, we insert 357 for which the hash values are 10, 13 and 2, so A changes to:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	0	1	0	0	1	0	0	1	0	0	1	0	1

Finally, let us insert 471 with hash values $2, 9, 3$. The bit at position 2 in A is already set. Setting the bits at positions 3 and 9 leaves A as:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	1	1	0	0	1	0	1	1	0	0	1	0	1

Let us now see how we can search in the filter. Let us first search for 789. Applications of the hash functions give the indices 13, 15 and 11. The bits at indices 13 and 15 are set, but the bit at index 11 is not set, implying that 789 has not been inserted in the filter.

Then, we search for 513. The hash values in this case are 9, 4 and 10. The bits of A at all these three positions are set, so we output *Yes*. This is a false positive, since 513 has not been inserted in the filter. The bits at indices 4, 9 and 10 were set during the insertions of 825, 471 and 357, respectively.

s is the size of the hash table, k hash functions, n insertions
p is the allowed probability of false positives

Probability a bit is set: $$1 - \left(1 - \frac{1}{s}\right)^{kn}$$

Probability of a false positive: $$\left(1 - \left(1 - \frac{1}{s}\right)^{kn}\right)^k \approx \left(1 - e^{-kn/s}\right)^k$$

Minimized at $k \approx s \ln 2 / n$ with $$2^{-k} \approx (2^{-\ln 2})^{s/n} = p$$

This gives $$s \approx -\frac{n \ln p}{\ln^2 2}.$$