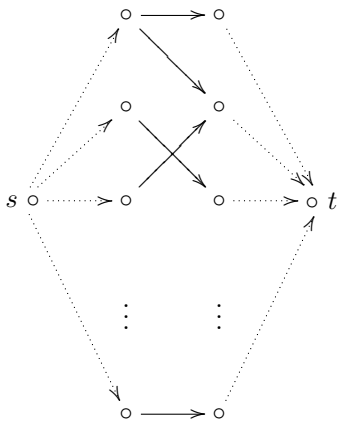

FIGURE 1. Reduction of matching to flow.

One final observation is that perfect matching is not known to be complete for any natural complexity class.

8.1.2. Primality testing. We present the Rabin-Miller randomized algorithm for primality testing. Although a polytime (deterministic) algorithm for primality is now known (see [**MA04**]), randomized algorithms[1] are simpler and more efficient, and therefore still used in practice.

ALGORITHM 8.2 (Rabin-Miller).
On input $(n)_b$*:*

1. *If* $n = 2$*, accept; if* n *is even and* $n > 2$*, reject.*
2. *Choose at random a positive* a *in* $\mathbb{Z}_n$*.*
3. *If* $a^{(n-1)} \not\equiv 1 \pmod{n}$*, reject.*
4. *Find* s, h *such that* s *is odd and* $n - 1 = s2^h$*.*
5. *Compute the sequence* $a^{s\cdot 2^0}, a^{s\cdot 2^1}, a^{s\cdot 2^2}, \ldots, a^{s\cdot 2^h} \pmod{n}$*.*
6. *If all elements in the sequence are 1, accept.*
7. *If the last element different from 1 is* -1*, accept. Otherwise, reject.*

Note that this is a polytime (randomized) algorithm: computing powers $\pmod{n}$ can be done efficiently with repeated squaring—for example, if $(n-1)_b = c_r \ldots c_1 c_0$, then compute

$$a_0 \equiv_n a, a_1 \equiv_n a_0^2, a_2 \equiv_n a_1^2, \ldots, a_r \equiv_n a_{r-1}^2,$$

and so $a^{n-1} \equiv_n a_0^{c_0} a_1^{c_1} \cdots a_r^{c_r}$. Thus obtaining the powers in lines 3 and 5 is not a problem. The highest power of 2 that divides $n-1$ is evident from $(n)_b$, so line 4 is not a problem either. Finally, choosing a non-zero $a \in \mathbb{Z}_n$ in a random way can be done by "flipping a coin" to obtain a string of bits of length $\log n$ (to ensure that a is not zero we choose at random the position of a 1 in the string, and then generate the other bits).

[1] In fact it was the randomized test for primality that stirred interest in randomized computation in the late 1970's. Historically, the first randomized algorithm for primality was given by [**SS77**]; a nice self-contained exposition of this algorithm can be found in [**Pap94**, §11.1], and another in [**vzGG99**, §18.5].

THEOREM 8.3. *If n is a prime then the Rabin-Miller algorithm accepts it; if n is composite, then the algorithm rejects it with probability $\geq \frac{1}{2}$.*

PROOF. If n is prime, then by Fermat's little theorem $a^{(n-1)} \equiv 1 \pmod{n}$, so line 3 cannot reject n. Suppose that line 7 rejects n; then there exists a b in $\mathbb{Z}_n$ such that $b \not\equiv \pm 1 \pmod{n}$ and $b^2 \equiv 1 \pmod{n}$. Therefore, $b^2 - 1 \equiv 0 \pmod{n}$, and hence

$$(b-1)(b+1) \equiv 0 \pmod{n}.$$

Since $b \not\equiv \pm 1 \pmod{n}$, both $(b-1)$ and $(b+1)$ are strictly between 0 and n, and so a prime n cannot divide their product. This gives a contradiction, and therefore no such b exists, and so line 7 cannot reject n.

If n is an odd composite number, then we say that a is a *witness* (of compositness) for n if the algorithm rejects on a. We show that if n is an odd composite number, then at least half of the a's in $\mathbb{Z}_n$ are witnesses. The distribution of those witnesses in $\mathbb{Z}_n$ appears to be very irregular, but if we choose our a at random, we hit a witness with probability $\geq \frac{1}{2}$.

Because n is composite, either n is the power of an odd prime, or n is the product of two odd co-prime numbers. This yields two cases.

Case 1. Suppose that $n = q^e$ where q is an odd prime and $e > 1$. Set $t := 1 + q^{e-1}$. From the binomial expansion of t^n we obtain:

$$t^n = (1 + q^{e-1})^n = 1 + nq^{e-1} + \sum_{l=2}^{n} \binom{n}{l} (q^{e-1})^l, \tag{24}$$

and therefore $t^n \equiv 1 \pmod{n}$. If $t^{n-1} \equiv 1 \pmod{n}$, then $t^n \equiv t \pmod{n}$, which from the observation about t and t^n is not possible, hence t is a line 3 witness. But the set of line 3 nonwitnesses, $S_1 := \{a \in \mathbb{Z}_n | a^{(n-1)} \equiv 1 \pmod{n}\}$, is a subgroup of $\mathbb{Z}_n^*$, and since it is not equal to $\mathbb{Z}_n^*$ (t is not in it), by Lagrange's theorem S_1 is at most half of $\mathbb{Z}_n^*$, and so it is at most half of $\mathbb{Z}_n$.

Case 2. Suppose that $n = qr$, where q, r are co-prime. Among all line 7 nonwitnesses, find a nonwitness for which the -1 appears in the largest position in the sequence in line 5 of the algorithm (note that -1 is a line 7 nonwitness, so the set of these nonwitnesses is not empty). Let x be such a nonwitness and let j be the position of -1 in its sequence, where the positions are numbered starting at 0; $x^{s \cdot 2^j} \equiv -1 \pmod{n}$ and $x^{s \cdot 2^{j+1}} \equiv 1 \pmod{n}$. The line 7 nonwitnesses are a subset of $S_2 := \{a \in \mathbb{Z}_n^* | a^{s \cdot 2^j} \equiv \pm 1 \pmod{n}\}$, and S_2 is a subgroup of $\mathbb{Z}_n^*$.

By the CRT there exists $t \in \mathbb{Z}_n$ such that

$$\begin{array}{ll} t \equiv x & \pmod{q} \\ t \equiv 1 & \pmod{r} \end{array} \quad \Rightarrow \quad \begin{array}{ll} t^{s \cdot 2^j} \equiv -1 & \pmod{q} \\ t^{s \cdot 2^j} \equiv 1 & \pmod{r} \end{array}$$

Hence t is a witness because $t^{s \cdot 2^j} \not\equiv \pm 1 \pmod{n}$ (see footnote[2]) but on the other hand $t^{s \cdot 2^{j+1}} \equiv 1 \pmod{n}$.

---

[2]To see why $t^{s \cdot 2^j} \not\equiv \pm 1 \pmod{n}$ observe the following: suppose that $a \equiv -1 \pmod{q}$ and $a \equiv 1 \pmod{r}$, where $\gcd(q, r) = 1$. Suppose that $n = qr | (a+1)$, then $q | (a+1)$ and $r | (a+1)$, and since $r | (a-1)$ as well, it follows that $r | [(a+1) - (a-1)]$, so $r | 2$, so $r = 2$, so n must be even, which is not possible since we deal with even n's in line 1 of the algorithm.

Therefore, just as in case 1, we have constructed a $t \in \mathbb{Z}_n^*$ which is not in S_2, and so S_2 can be at most half of $\mathbb{Z}_n^*$, and so at least half of the elements in $\mathbb{Z}_n$ are witnesses. □

EXERCISE 8.4. *First show that the sets S_1 and S_2 (in the proof of theorem 8.3) are indeed a subgroups of $\mathbb{Z}_n^*$, and that in case 2 all nonwitnesses are contained in S_2. Then show that at least half of the elements of $\mathbb{Z}_n$ are witnesses when n is composite, without using group theory.*

Note that by running the algorithm k times on independently chosen a, we can make sure that it rejects a composite with probability $\geq 1 - \frac{1}{2^k}$ (it will always accept a prime with probability 1). Thus, for $k = 100$ the probability of error, i.e., of a false positive, is negligible.

Thus we have a Monte Carlo algorithm for composites, and therefore PRIMES $= \{(n)_b | n \text{ is primes}\} \in$ co-RP. Here $(n)_b$ denotes the binary encoding of the number n; see §8.2 for a definition of co-RP.

8.1.3. Pattern matching. In this section we design a randomized algorithm for pattern matching. Consider the set of strings over $\{0,1\}$, and let $M : \{0,1\} \longrightarrow M_{2\times 2}(\mathbb{Z})$, that is, M is a map from strings to 2×2 matrices over the integers ($\mathbb{Z}$) defined as follows:

$$M(\varepsilon) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad M(0) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}; \quad M(1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

and for strings $x, y \in \{0,1\}^*$, $M(xy) = M(x)M(y)$, where the operation on the LHS is concatenation of strings, and the operation on the RHS is multiplication of matrices.

First of all, $M(x)$ is well defined because matrix multiplication is associative, and second of all, $M(x) = M(y)$ implies that $x = y$ (i.e., the map M is 1-1). Given $M = M(x)$ we can "decode" x uniquely as follows: if the first column of M is greater than the second (where the comparison is made component-wise), then the last bit of x is zero, and otherwise it is 1. Let M' be M where we subtract the smaller column from the larger, and repeat.

For $x \in \{0,1\}^n$, the entries of $M(x)$ are bounded by Fibonacci number F_n. Let $F_0 = F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n > 1$. For a given string x, $M(x_1 x_2 \dots x_n)$ is such that the "smaller" column is bounded by F_{n-1} and the "larger" column is bounded by F_n. We can show this inductively: the basis case, $x = x_1$, is obvious. For the inductive step, assume it holds for $x \in \{0,1\}^n$, and show it still holds for $x \in \{0,1\}^{n+1}$: this is obvious as whether x_{n+1} is 0 or 1, one column is added to the other, and the other column remains unchanged.

By considering the matrices $M(x)$ modulo a suitable prime p, we perform efficient randomized pattern matching. We wish to determine whether x is a substring of y, where $|x| = n$, $|y| = m$, $n \leq m$. Let $y(i) = y_i y_{i+1} \dots y_{n+i-1}$, for appropriate i's. Select a prime $p \in \{1, \dots, nm^2\}$, and let $A = M(x) \pmod p$ and $A(i) \equiv M(y(i)) \pmod p$. Note that

$$A(i+1) \equiv M^{-1}(y_i) A(i) M(y_{n+i}) \pmod p,$$

which makes the computation of subsequent $A(i)$'s efficient.