(ii) $b_i = \prod_{j=1}^{t} p_j^{e_{ij}}$, $e_{ij} \geq 0$; that is, b_i is p_t-smooth.

Next find a subset of the b_i's whose product is a perfect square. Knowing the factorizations of the b_i's, this is possible by selecting a subset of the b_i's such that the power of each prime p_j appearing in their product is even. For this purpose, only the parity of the non-negative integer exponents e_{ij} needs to be considered. Thus, to simplify matters, for each i, associate the binary vector $v_i = (v_{i1}, v_{i2}, \ldots, v_{it})$ with the integer exponent vector $(e_{i1}, e_{i2}, \ldots, e_{it})$ such that $v_{ij} = e_{ij} \bmod 2$. If $t+1$ pairs (a_i, b_i) are obtained, then the t-dimensional vectors $v_1, v_2, \ldots, v_{t+1}$ must be linearly dependent over $\mathbb{Z}_2$. That is, there must exist a non-empty subset $T \subseteq \{1, 2, \ldots, t+1\}$ such that $\sum_{i \in T} v_i = 0$ over $\mathbb{Z}_2$, and hence $\prod_{i \in T} b_i$ is a perfect square. The set T can be found using ordinary linear algebra over $\mathbb{Z}_2$. Clearly, $\prod_{i \in T} a_i^2$ is also a perfect square. Thus setting $x = \prod_{i \in T} a_i$ and y to be the integer square root of $\prod_{i \in T} b_i$ yields a pair of integers (x, y) satisfying $x^2 \equiv y^2 \pmod{n}$. If this pair also satisfies $x \not\equiv \pm y \pmod{n}$, then $\gcd(x - y, n)$ yields a non-trivial factor of n. Otherwise, some of the (a_i, b_i) pairs may be replaced by some new such pairs, and the process is repeated. In practice, there will be several dependencies among the vectors $v_1, v_2, \ldots, v_{t+1}$, and with high probability at least one will yield an (x, y) pair satisfying $x \not\equiv \pm y \pmod{n}$; hence, this last step of generating new (a_i, b_i) pairs does not usually occur.

This description of the random square methods is incomplete for two reasons. Firstly, the optimal choice of t, the size of the factor base, is not specified; this is addressed in Note 3.24. Secondly, a method for efficiently generating the pairs (a_i, b_i) is not specified. Several techniques have been proposed. In the simplest of these, called *Dixon's algorithm*, a_i is chosen at random, and $b_i = a_i^2 \bmod n$ is computed. Next, trial division by elements in the factor base is used to test whether b_i is p_t-smooth. If not, then another integer a_i is chosen at random, and the procedure is repeated.

The more efficient techniques strategically select an a_i such that b_i is relatively small. Since the proportion of p_t-smooth integers in the interval $[2, x]$ becomes larger as x decreases, the probability of such b_i being p_t-smooth is higher. The most efficient of such techniques is the quadratic sieve algorithm, which is described next.

3.2.6 Quadratic sieve factoring

Suppose an integer n is to be factored. Let $m = \lfloor \sqrt{n} \rfloor$, and consider the polynomial $q(x) = (x + m)^2 - n$. Note that

$$q(x) \;=\; x^2 + 2mx + m^2 - n \;\approx\; x^2 + 2mx, \tag{3.1}$$

which is small (relative to n) if x is small in absolute value. The quadratic sieve algorithm selects $a_i = (x + m)$ and tests whether $b_i = (x + m)^2 - n$ is p_t-smooth. Note that $a_i^2 = (x + m)^2 \equiv b_i \pmod{n}$. Note also that if a prime p divides b_i then $(x + m)^2 \equiv n \pmod{p}$, and hence n is a quadratic residue modulo p. Thus the factor base need only contain those primes p for which the Legendre symbol $\left(\frac{n}{p}\right)$ is 1 (Definition 2.145). Furthermore, since b_i may be negative, -1 is included in the factor base. The steps of the quadratic sieve algorithm are summarized in Algorithm 3.21.

3.21 Algorithm Quadratic sieve algorithm for factoring integers

INPUT: a composite integer n that is not a prime power.
OUTPUT: a non-trivial factor d of n.

1. Select the factor base $S = \{p_1, p_2, \ldots, p_t\}$, where $p_1 = -1$ and p_j ($j \geq 2$) is the $(j-1)^{\text{th}}$ prime p for which n is a quadratic residue modulo p.
2. Compute $m = \lfloor \sqrt{n} \rfloor$.
3. (*Collect $t+1$ pairs (a_i, b_i). The x values are chosen in the order $0, \pm 1, \pm 2, \ldots$.*) Set $i \leftarrow 1$. While $i \leq t+1$ do the following:
   3.1 Compute $b = q(x) = (x+m)^2 - n$, and test using trial division (cf. Note 3.23) by elements in S whether b is p_t-smooth. If not, pick a new x and repeat step 3.1.
   3.2 If b is p_t-smooth, say $b = \prod_{j=1}^{t} p_j^{e_{ij}}$, then set $a_i \leftarrow (x+m)$, $b_i \leftarrow b$, and $v_i = (v_{i1}, v_{i2}, \ldots, v_{it})$, where $v_{ij} = e_{ij} \bmod 2$ for $1 \leq j \leq t$.
   3.3 $i \leftarrow i+1$.
4. Use linear algebra over $\mathbb{Z}_2$ to find a non-empty subset $T \subseteq \{1, 2, \ldots, t+1\}$ such that $\sum_{i \in T} v_i = 0$.
5. Compute $x = \prod_{i \in T} a_i \bmod n$.
6. For each j, $1 \leq j \leq t$, compute $l_j = (\sum_{i \in T} e_{ij})/2$.
7. Compute $y = \prod_{j=1}^{t} p_j^{l_j} \bmod n$.
8. If $x \equiv \pm y \pmod{n}$, then find another non-empty subset $T \subseteq \{1, 2, \ldots, t+1\}$ such that $\sum_{i \in T} v_i = 0$, and go to step 5. (In the unlikely case such a subset T does not exist, replace a few of the (a_i, b_i) pairs with new pairs (step 3), and go to step 4.)
9. Compute $d = \gcd(x - y, n)$ and return(d).

3.22 Example (*quadratic sieve algorithm for finding a non-trivial factor of $n = 24961$*)

1. Select the factor base $S = \{-1, 2, 3, 5, 13, 23\}$ of size $t = 6$. (7, 11, 17 and 19 are omitted from S since $\left(\frac{n}{p}\right) = -1$ for these primes.)
2. Compute $m = \lfloor \sqrt{24961} \rfloor = 157$.
3. Following is the data collected for the first $t+1$ values of x for which $q(x)$ is 23-smooth.

i	x	$q(x)$	factorization of $q(x)$	a_i	v_i
1	0	-312	$-2^3 \cdot 3 \cdot 13$	157	$(1,1,1,0,1,0)$
2	1	3	3	158	$(0,0,1,0,0,0)$
3	-1	-625	-5^4	156	$(1,0,0,0,0,0)$
4	2	320	$2^6 \cdot 5$	159	$(0,0,0,1,0,0)$
5	-2	-936	$-2^3 \cdot 3^2 \cdot 13$	155	$(1,1,0,0,1,0)$
6	4	960	$2^6 \cdot 3 \cdot 5$	161	$(0,0,1,1,0,0)$
7	-6	-2160	$-2^4 \cdot 3^3 \cdot 5$	151	$(1,0,1,1,0,0)$

4. By inspection, $v_1 + v_2 + v_5 = 0$. (In the notation of Algorithm 3.21, $T = \{1, 2, 5\}$.)
5. Compute $x = (a_1 a_2 a_5 \bmod n) = 936$.
6. Compute $l_1 = 1$, $l_2 = 3$, $l_3 = 2$, $l_4 = 0$, $l_5 = 1$, $l_6 = 0$.
7. Compute $y = -2^3 \cdot 3^2 \cdot 13 \bmod n = 24025$.
8. Since $936 \equiv -24025 \pmod{n}$, another linear dependency must be found.
9. By inspection, $v_3 + v_6 + v_7 = 0$; thus $T = \{3, 6, 7\}$.
10. Compute $x = (a_3 a_6 a_7 \bmod n) = 23405$.
11. Compute $l_1 = 1$, $l_2 = 5$, $l_3 = 2$, $l_4 = 3$, $l_5 = 0$, $l_6 = 0$.

12. Compute $y = (-2^5 \cdot 3^2 \cdot 5^3 \bmod n) = 13922$.
13. Now, $23405 \not\equiv \pm 13922 \pmod{n}$, so compute $\gcd(x-y, n) = \gcd(9483, 24961) = 109$. Hence, two non-trivial factors of 24961 are 109 and 229. □

3.23 Note (*sieving*) Instead of testing smoothness by trial division in step 3.1 of Algorithm 3.21, a more efficient technique known as *sieving* is employed in practice. Observe first that if p is an odd prime in the factor base and p divides $q(x)$, then p also divides $q(x+lp)$ for every integer l. Thus by solving the equation $q(x) \equiv 0 \pmod{p}$ for x (for example, using the algorithms in §3.5.1), one knows either one or two (depending on the number of solutions to the quadratic equation) entire sequences of other values y for which p divides $q(y)$.

The *sieving process* is the following. An array $Q[\,]$ indexed by x, $-M \leq x \leq M$, is created and the x^{th} entry is initialized to $\lfloor \lg |q(x)| \rfloor$. Let x_1, x_2 be the solutions to $q(x) \equiv 0 \pmod{p}$, where p is an odd prime in the factor base. Then the value $\lfloor \lg p \rfloor$ is subtracted from those entries $Q[x]$ in the array for which $x \equiv x_1$ or $x_2 \pmod{p}$ and $-M \leq x \leq M$. This is repeated for each odd prime p in the factor base. (The case of $p = 2$ and prime powers can be handled in a similar manner.) After the sieving, the array entries $Q[x]$ with values near 0 are most likely to be p_t-smooth (roundoff errors must be taken into account), and this can be verified by factoring $q(x)$ by trial division.

3.24 Note (*running time of the quadratic sieve*) To optimize the running time of the quadratic sieve, the size of the factor base should be judiciously chosen. The optimal selection of $t \approx L_n[\frac{1}{2}, \frac{1}{2}]$ (see Example 2.61) is derived from knowledge concerning the distribution of smooth integers close to $\sqrt{n}$. With this choice, Algorithm 3.21 with sieving (Note 3.23) has an expected running time of $L_n[\frac{1}{2}, 1]$, independent of the size of the factors of n.

3.25 Note (*multiple polynomial variant*) In order to collect a sufficient number of (a_i, b_i) pairs, the sieving interval must be quite large. From equation (3.1) it can be seen that $|q(x)|$ increases linearly with $|x|$, and consequently the probability of smoothness decreases. To overcome this problem, a variant (the *multiple polynomial quadratic sieve*) was proposed whereby many appropriately-chosen quadratic polynomials can be used instead of just $q(x)$, each polynomial being sieved over an interval of much smaller length. This variant also has an expected running time of $L_n[\frac{1}{2}, 1]$, and is the method of choice in practice.

3.26 Note (*parallelizing the quadratic sieve*) The multiple polynomial variant of the quadratic sieve is well suited for parallelization. Each node of a parallel computer, or each computer in a network of computers, simply sieves through different collections of polynomials. Any (a_i, b_i) pair found is reported to a central processor. Once sufficient pairs have been collected, the corresponding system of linear equations is solved on a single (possibly parallel) computer.

3.27 Note (*quadratic sieve vs. elliptic curve factoring*) The elliptic curve factoring algorithm (§3.2.4) has the same[4] expected (asymptotic) running time as the quadratic sieve factoring algorithm in the special case when n is the product of two primes of equal size. However, for such numbers, the quadratic sieve is superior in practice because the main steps in the algorithm are single precision operations, compared to the much more computationally intensive multi-precision elliptic curve operations required in the elliptic curve algorithm.

[4] This does not take into account the different $o(1)$ terms in the two expressions $L_n[\frac{1}{2}, 1]$.