Die Schlacht von Hastings

aus *Amusements in Mathematics*

von Henry Ernest Dudeney, 1917

Die folgende Aufgabe bezieht sich auf die Schlacht bei Hastings, jenen berühmten Kampf, in dem 1066 die Normannen unter Wilhelm dem Eroberer die Sachsen unter König Harald besiegten, und fortan die Geschicke Englands bestimmten. Nach *Dudeney* schildert die alte Chronik:
Haralds Mannen standen tapfer zusammen und bildeten 61 Quadrate mit gleich vielen Recken in jedem Quadrat. Als Harald sich in die Schlacht warf, bildeten die Sachsen mit ihm zusammen ein einziges, mächtiges Quadrat.
Abbildung 1 illustriert die beiden Stellungen der Sachsen mit seinem König Harald. Gesucht ist nun die minimal mögliche Anzahl der Krieger in den kleinen,

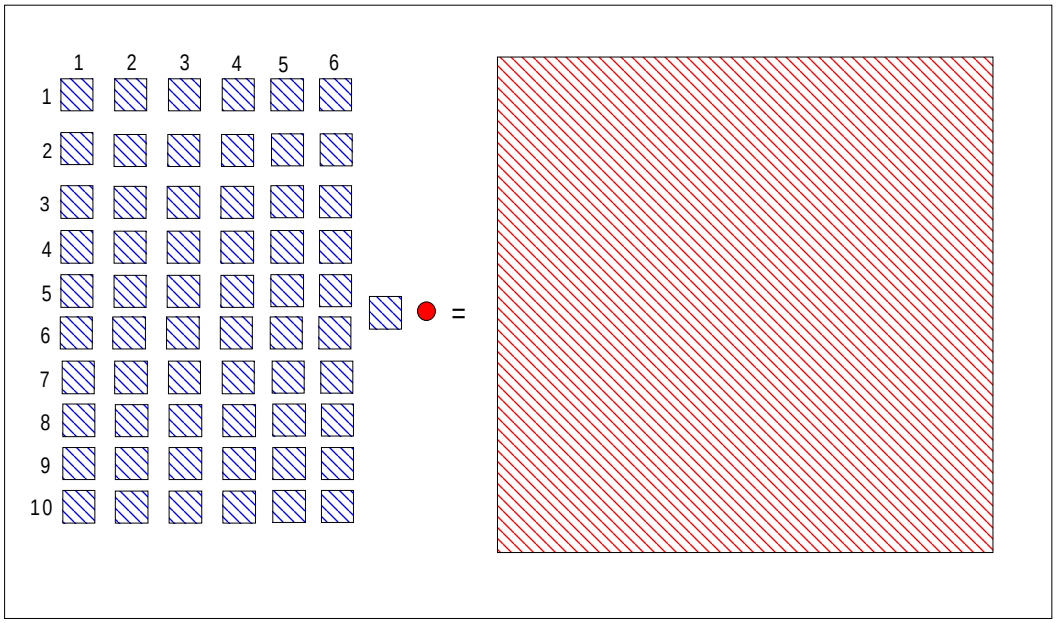

Abbildung 1: Die Schlacht bei Hastings mit König Harald

Literaturhinweise

/1/ Scheidt, H. : Zahlentheorie , 2. Auflage 1998, Wissenschaftsverlag Mannheinm
/2/ Forster, O. : Algorithmische Zahlentheorie, 1. Auflage 1999, Vieweg Verlag

Lösungsvorschlag

Wir bezeichen die Kantenlänge der kleinen (blauen) Quadrate mit x und die vom großen Quadrat mit y. Abbildung 1 entspricht der folgenden Gleichung:

$$61\,x^2 + 1 = y^2 \tag{1}$$

Bei (1) handelt es sich um eine diophantische Gleichung 2.Ordnung. In der Zahlentheorie bezeichnet man Gleichungen der Form :

$$y^2 - d\,x^2 = 1 \tag{2}$$

als *Pellsche Gleichung*, wobei d quadratfrei sein muß. Mit Hilfe der Kettenbruchentwicklung können *Pellsche Gleichungen* eindeutig gelöst werden. In /1/ wird der Algorithmus ausführlich erklärt. Im Programm ARIBAS

`http://www.mathematik.uni-muenchen.de/~forster/sw/aribas.html`

sieht der Quelltext wie folgt aus :

```
(************************************************************)
(*
** Otto Forster: Algorithmische Zahlentheorie
** Vieweg-Verlag 1996, ISBN 3-528-06580-X
**
** ARIBAS-Code zu Paragraph 25
** Die Pellsche Gleichung
*)
(*------------------------------------------------------------*)
(*
** Berechnet die kleinste nicht-triviale Loesung der
** Pellschen Gleichung x*x - d*y*y = 1
** d > 0 darf kein Quadrat sein.
**
** Beispiel-Aufruf: pell(109).
*)
function pell(d: integer): array[2];
var
    u,v,c,u2: integer;
    z: array[3];
begin
    z := pell4(d);
    u := z[0]; v := z[1]; c := z[2];
    if abs(c) = 4 then
        u2 := u*u; c := c div 4;
        u := u*((u2 - 3*c) div 2);
        v := v*((u2 - c) div 2);
```

```
    end;
    if c < 0 then
        v := 2*u*v;
        u := 2*u*u + 1;
    end;
    return (u,v);
end;
(*-------------------------------------------------------------*)
(*
** Berechnet die kleinste nicht-triviale Loesung von
**      u*u - d*v*v = c, mit c aus {-4,+4,-1,+1}
** mittels Kettenbruch-Entwicklung von sqrt(d)
** und gibt (u,v,c) zurueck.
** Das Argument d muss eine positive ganze Zahl sein,
** die kein Quadrat ist.
*)
function pell4(d: integer): array[3];
var
    a, w, q, q0, q1, m, m1, sign: integer;
    u, u0, u1, v, v0, v1: integer;
begin
    if d=5 then return (1,1,-4) end;
    w := isqrt(d); q := d - w*w;
    if q=0 then
        writeln("error: argument d is a perfect square");
        halt();
    end;
    q0 := 1; m := w;
    u0 := 1; u := m; v0 := 0; v := 1;
    sign := -1;
    while q /= 4 and q /= 1 do
        a := (m + w) div q;
        m1 := a*q - m; q1 := q0 + a*(m - m1);
        u1 := a*u + u0; v1 := a*v + v0;
        m := m1; q0 := q; q := q1;
        u0 := u; u := u1; v0 := v; v := v1;
        sign := -sign;
    end;
    return (u,v,sign*q);
end;
(*-------------------------------------------------------------*)
(*
** Kettenbruch-Entwicklung von sqrt(N)
** Resultat ist ein Vektor (a0,a1,a2,...,an,2*a0),
** wobei a0 = [sqrt(N)] und (a1,a2,...,an,2*a0) die Periode ist.
*)
function sqrtn2cfrac(N: integer): array;
```

```
var
    a, w, w2, q, q0, q1, m, m1: integer;
    st: stack;
begin
    w := isqrt(N); q := N - w*w;
    if q = 0 then
        writeln(N," is a perfect square");
        return {w};
    end;
    w2 := 2*w; a := w; m := w; q0 := 1;
    stack_push(st,a);
    while a /= w2 do
        a := (m + w) div q;
        stack_push(st,a);
        m1 := a*q - m; q1 := q0 + a*(m - m1);
        m := m1; q0 := q; q := q1;
    end;
    return stack2array(st);
end;
(*-------------------------------------------------------------------*)
```

Für $d = 61$ folgt mit *ARIBAS* die Lösung $x = 226153980, \quad y = 1766319049$.