Generilizations of the Josephus problem

Nicolas Thériault

Abstract. This article presents two algorithmic solutions to an extended version of the Josephus problem. In addition, the number of steps required for one of these algorithms is discussed. Finally, a variation on the Josephus problem and an algorithmic solution to this new problem are also given.

1 Introduction

Let be given n players labeled from 1 to n placed consecutively around a circular table and k a positive integer (sometimes with the condition $n \geq k$). Suppose that, counting from player 1, the k^{th} player is removed from the table, then the following k^{th} player and so forth until all the players are removed. The problem is to predict beforehand the number $F(n,k)$ of the last remaining player. This is known as the Josephus Problem. More generaly, one might want to predict the number $F_l(n,k)$ of the next player to be removed when all but l players have been removed from the table; for instance, $F_1(n,k)$ $(= F(n,k))$ is the last player removed, while $F_2(n,k)$ is the second-to-last. This will be called the extended problem

The first known reference to such a process dates to the life of the jewish historian Josephus (also known as Flavius Josephe, 37 to 100 AD), who, according to Hegesippus (*De Bello Judaico*, book 3, chapters 16-18) used it to save his life. After the capture of Jotapat by the Romans, Josephus and forty other Jews found refuge in a cave. Knowing that all the others, with the exception of one, were resolved to kill themselves rather than be taken, and not wanting to oppose himself openly to the group, Josephus proposed to procede in an orderly fashion. All 41 men would place themselves in a cercle and every third person would be killed until the last man remaining would commit suicide. Josephus then placed himself, and the other man, in the 16^{th} and 31^{th} place, so both of them could survivre.

There was very little developpement in the following centuries althought *decimation puzzles* (puzzles based on the progressive elimination of players) gained some popularity during the Middle Ages [1]. It was only in 1775 that Euler [2] generalized the Josephus Problem to all values of n and k with $n \geq k$. The first algorithmic method was presented by P.G. Tait in

1899 [3]. During the 20^{th} century, methods of computing $F(n,k)$ more rapidly (when $k>1$) by finding the values of n greater than k for which $1 \leq F(n,k) \leq k-1$ were developped (see [4], [5], [6]). Of these methods, only a variation of the algorithm presented by Klaus Burde will be discussed here.

A number of new results are presented here. First, the algorithms developped by Tait and Burde are adapted to solve the extended problem (the computation of $F_l(n,k)$). Then, a theorem on the number of iterations needed when using the extended version of Burde's algorithm is given. Finally, a variation on the Josephus problem and an algorithmic solution to this new problem are also given.

Throughout the text, the fonction $modp(a,b)$ will be used. It refers to the fonction modp(a,b) of Maple (equivalent to Mod[a,b] for Mathematica) which returns the residue of a modulo b. When this function is written as $modp(a-1,b)+1$, it will return the integer in $\{1,\ldots,b\}$ which is in the same class as a modulo b. Maple programs will be given for the generalized versions of the algorithms of Tait and Burde and for the solution of the extended problem.

2 Algorithms for $F(n,k)$

2.1 Tait's algorithm

Let n and k be positive integers and $F(n,k)$ as defined above, i.e. the initial position of the last player removed. Then $F(n,k)$ is obtained by progressively increassing j from 1 to n in $F(j,k)$, as follows:

$$F(1,k) = 1, \tag{1}$$
$$F(j+1,k) = modp\left(F(j,k)+k-1, j+1\right)+1. \tag{2}$$

Proof: Since (1) is trivial by definition of $F(1,k)$, only (2) needs to be proven. Suppose there are j players around the table of which every k^{th} will be removed and that $F(j,k)$ is already known. If player $j+1$ is added between players j and 1 and the counting begins with player number $modp\Big((1-k)-1, j+1\Big)+1$ instead of player 1, then player $j+1$ is the first to be removed and $F(j,k)$ is still the last player removed from the table. If players are given new numbers from 1 to $j+1$ begining from player $modp\Big((1-k)-1, j+1\Big)+1$, which is equivalent to replacing the m^{th} position by the $\Big(modp(m+k-1, j+1)+1\Big)^{\text{th}}$ position, then $F(j,k)$ becomes $modp(F(j,k)+k-1, j+1)+1$. But the resulting situation is $j+1$ players of which every k^{th} is removed, the counting begining with

player 1 and player $modp\Big(F(j,k)+k-1, j+1\Big)+1$ being the last removed. Therefore $F(j+1,k) = modp\Big(F(j,k)+k-1, n+1\Big)+1$. Q.E.D.

2.2 Burde's algorithm

The algorithm developped by Burde allows to compute $F(n,k)$ (if $k \geq 2$) by finding values of $n \geq n_0$ for which $1 \leq F(n,k) \leq k-1$ with n_0 the first value of $n \geq k$ for which $F(n,k) = 1$ (this value can be relatively large, for exemple, when $k = 9$ one finds $n_0 = 91$ and for $k = 14$, $n_0 = 146$); for values of n less than n_0, $F(n,k)$ is found using Tait's algorithm. The algorithm presented here is a variation of Burde's that allows to set n_0 as the smallest $n \geq k$ for which $1 \leq F(n,k) < k$.

Given k, a positive integer greater than 1, we define $\rho = \frac{k}{k-1}$. The main body of the algorithm consists of finding n_i (and $F(n_i,k)$) such that $n_i \leq n < n_{i+1}$. One begins with

$$f_0 = F(k,k) \quad \text{and} \quad n_0 = k \tag{3}$$

($F(k,k)$ being computed with Tait's algorithm) and progressively increases the value of i up to the first value of i for which $n_{i+1} > n$, each step requiring the computation of f_i, n_i and $F(n_i,k)$ as follows:

$$\begin{aligned} f_{i+1} &= modp\Big(F(n_i,k) - n_i - 1, k-1\Big) + 1 - F(n_i,k), & (4)\\ n_{i+1} &= (kn_i + f_{i+1})/(k-1), & (5)\\ F(n_{i+1},k) &= F(n_i,k) + f_{i+1}. & (6) \end{aligned}$$

Once the values of the desired n_i and $F(n_i,k)$ have been determined, $F(n,k)$ is computed as follows:

$$F(n,k) = F(n_i,k) + k(n - n_i). \tag{7}$$

In addition, if one were to record the value of f_j for $0 \leq j \leq j'$, it would be easy to recover n_i and $F(n_i,k)$ with $i \leq j'$ using the sums:

$$n_i = n_0\rho^i + \frac{1}{k-1}\sum_{j=1}^{i} f_j \rho^{i-j} \tag{8}$$

and

$$F(n_i,k) = \sum_{j=0}^{i} f_i. \tag{9}$$

Proof: To prove (3), simply note that when $n = k$, the first player removed from the table will be player k and $F(n,k)$ will be one of the players $\{1,\ldots,k-1\}$, so k is the smallest value of $n \geq k$ for which $1 \leq F(n,k) < k$.

It should be noted that for $n > k$, we have

$$k < F(n-1,k) + k \leq (n-1) + k < 2n,$$

and (2) becomes

$$F(n,k) = \begin{cases} F(n-1,k) + k & \text{if } F(n-1,k) + k \leq n, \\ F(n-1,k) + k - n & \text{if } F(n-1,k) + k > n. \end{cases} . \qquad (10)$$

Given n_i and $F(n_i,k)$, we define n_{i+1} as the smallest value of $n > n_i$ for which $1 \leq F(n,k) \leq k-1$. It is easy to see from the (10) that n_{i+1} is the smallest value of $n > n_i$ for which $F(n,k) = F(n-1,k)+k-n$; (under this condition, we have $1 \leq F(n,k) = F(n-1,k)+k-n \leq (n-1)+k-n < k$). In addition, it is easily shown that for every $n_i < n < n_{i+1}$ we have $F(n,k) = F(n-1,k) + k = F(n_i,k) + k(n-n_i)$, which gives us (7) and $F(n_{i+1},k) = F(n_i,k) + k(n_{i+1} - n_i) - n_{i+1}$. If we define f_{i+1} for $i \geq 0$ as the difference between $F(n_{i+1},k)$ and $F(n_i,k)$ (from what (6) is trivial), the last equation can be rewritten as $f_{i+1} = (k-1)n_{i+1} - kn_i$, from which we have (5) and $(k-1)n_{i+1} = \rho(k-1)n_i + f_{i+1}$.

Since n_i is an integer for all $i \geq 0$, we have $(k-1)n_{i+1} \equiv 0 \pmod{k-1}$. We will therefore choose f_{i+1} such that $kn_i + f_{i+1} \equiv 0 \pmod{k-1}$, that is $f_{i+1} \equiv -n_i \pmod{k-1}$. What's more, from $1 \leq F(n_{i+1},k) \leq k-1$ and (6), we have $1 \leq F(n_i,k) + f_{i+1} \leq k-1$. Combining these two conditions for f_{i+1} and noting that

$$1 - (k-1) < 1 - (k-2) \leq F(n_i,k) + modp(-n_i, k-1) \leq (k-1),$$

we have

$$\begin{aligned} F(n_i,k) + f_{i+1} &= \begin{cases} F(n_i,k) + modp(-n_i,k-1) \\ \qquad \text{if } F(n_i,k) + modp(-n_i,k-1) \geq 1 \\ F(n_i,k) + modp(-n_i,k-1) + (k-1) \\ \qquad \text{if } F(n_i,k) + modp(-n_i,k-1) < 1 \end{cases} \\ &= modp\Big(F(n_i,k) + modp(-n_i,k-1) - 1, k-1\Big) + 1 \\ &= modp\Big(F(n_i,k) - n_i - 1, k-1\Big) + 1. \end{aligned}$$

This can be written as (4).

To obtain (8) we prove $(k-1)n_i = (k-1)n_0\rho^i + \sum_{j=1}^{i} f_j \rho^{i-j}$ from (3) and (5) as follows:

- If $i = 1$, we have $(k-1)n_1 = (k-1)n_0\rho + f_0 = (k-1)n_0\rho^1 + \sum_{j=1}^{1} f_j \rho^{1-j}$;

- If $(k-1)n_i = (k-1)n_0\rho^i + \sum_{j=1}^{i} f_j\rho^{i-j}$, then

$$\begin{aligned}(k-1)n_{i+1} &= \rho(k-1)n_i + f_{i+1} \\ &= \rho\left((k-1)n_0\rho^i + \sum_{j=1}^{i} f_j\rho^{i-j}\right) + f_{i+1} \\ &= (k-1)n_0\rho^{i+1} + \sum_{j=1}^{i} f_j\rho^{i+1-j} + f_{i+1} \\ &= (k-1)n_0\rho^{(i+1)} + \sum_{j=1}^{(i+1)} f_j\rho^{(i+1)-j}\ .\end{aligned}$$

Dividing by $k-1$ gives us (8). Similarly, (9) is proven by a simple iteration of (6). Q.E.D.

3 Algorithms for $F_l(n,k)$

3.1 Generalized Tait

Since $F_l(n,k)$ is well defined only if there are l or more players around the table, it will be obtained by progressively increasing j from l to n in $F_l(j,k)$, as follows:

$$F_l(l,k) = modp(k-1,l)+1, \tag{11}$$

$$F_l(j+1,k) = modp\left(F_l(j,k)+k-1, j+1\right)+1. \tag{12}$$

Proof: By definition, $F_l(n,k)$ is the next player removed from the table when there are l players remaining in the situation of l players of which every k^{th} is removed. Then $F_l(l,k)$ is the first player to be removed from the table, but this is the k^{th} player starting from player 1, that is player $modp(k-1,l)+1$, which gives (11). Finally, (12) is proven by the same argument used to prove (2). Q.E.D.

3.2 Generalized Burde

The first step in this algorithm is to determine f_0 and n_0, which is done as follows:

$$f_0 = \begin{cases} modp(-l,k-1)+1 & \text{if } k \le l, \\ F_l(k,k) & \text{if } k > l; \end{cases} \tag{13}$$

$$n_0 = \begin{cases} \frac{1}{k-1}\Big(f_0 + k(l-1)\Big) & \text{if } k \le l, \\ k & \text{if } k > l. \end{cases} \tag{14}$$

For $n > n_0$, we proceed to determine n_i such that $n_i \leq n < n_{i+1}$ in the same way as used for $F(n,k)$:

$$\begin{aligned} f_{i+1} &= modp\Big(F(n_i,k) - n_i - 1, k-1\Big) + 1 - F(n_i,k), \quad (15) \\ n_{i+1} &= (kn_i + f_{i+1})/(k-1), \quad (16) \\ F_l(n_{i+1},k) &= F_l(n_i,k) + f_{i+1}. \quad (17) \end{aligned}$$

Once n_i and $F_l(n_i,k)$ have been determined, $F_l(n,k)$ is given by

$$F_l(n,k) = F_l(n_i,k) + k(n - n_i). \quad (18)$$

When $l \leq n < k$, $F_l(n,k)$ is obtained with the generalized version of Tait and when $k \leq l \leq n < n_0$, $F_l(n,k)$ is given by

$$F_l(n,k) = F_l(l,k) + k(n-l) = k(n-l+1). \quad (19)$$

Notice that the two sums still hold when given as follows ($\rho = \frac{k}{k-1}$):

$$n_i = n_0\rho^i + \frac{1}{k-1}\sum_{j=1}^{i} f_j\rho^{i-j} \quad (20)$$

and

$$F_l(n_i,k) = \sum_{j=0}^{i} f_i \quad (21)$$

Proof: Since Burde's algorithm is in fact an improvement of the iterative part of Tait's algorithm, (15)-(18), (20) and (21) are proven in the same way as the equivalent equations for $F(n,k)$. Problems are encountered only when choosing the value of n_0. If $k > l$, there is no change: we define $n_0 = k$ and compute $f_0 = F_l(k,k)$ using Tait's algorithm (starting from $F_l(l,k)$ obviously). If $k \leq l$, Tait's algorithm can be used only from $n = l$ and we have

$$F_l(l,k) = modp(k-1,l) + 1 = k - 1 + 1 = k$$

(since $1 \leq k-1 < l$), which does not verify $1 \leq F_l(n,k) \leq k-1$. By the definition of n_0 (the smallest value of $n \geq k$ for which $1 \leq F_l(n,k) \leq k-1$) and (10) written for $F_l(n,k)$, if $l \leq n < n_0$, then we have $F_l(n,k) = F_l(l,k) + k(n-l) = k(n-l+1)$ (so (19) is proven). In addition, we find for n_0: $f_0 = F_l(n_0,k) = F_l(l,k) + k(n_0 - l) - n_0$, which is solved by taking

$$\begin{aligned} f_0 &= modp\Big(F_l(l,k) + k(n_0-l) - n_0 - 1, k-1\Big) + 1 \\ &= modp\Big(k + (k-1)n_0 - kl - 1, k-1\Big) + 1 \\ &= modp(-l, k-1) + 1 \end{aligned}$$

and then

$$\begin{aligned} n_0 &= \frac{1}{k-1}\Big(F_l(n_0,k) - F_l(l,k) + kl\Big) \\ &= \frac{1}{k-1}(f_0 - k + kl) \\ &= \frac{1}{k-1}\Big(f_0 + k(l-1)\Big). \end{aligned}$$

Combining the definitions of f_0 (and n_0) for $k \leq l$ and $k > l$ gives us (13) (and (14)). Q.E.D.

3.3 Number of steps

The following results are used to determine the number of iterations needed when using the generalized version of Burde's algorithm. In all cases, ρ is defined as $\frac{k}{k-1}$.

Lemma: *For $i \geq 1$, n_i verifies*

$$\left(n_0 - \frac{k-2}{k}\right)\rho^i \leq n_i \leq \left(n_0 + \frac{k-2}{k}\right)\rho^i \tag{22}$$

Proof: From (21) and since f_0 is defined as the value of $F_l(n_0,k)$, we have $\sum_{j=1}^{i} f_j = F_l(n_i,k) - F_l(n_0,k)$, hence $\left|\sum_{j=1}^{i} f_j\right| \leq k-2$. To prove the lemma, we will use this fact and "forget" that n_i must be an integer, which will allow us to "move" the f_j. As an exemple, if $\{f_1,\ldots,f_j,0,1,f_{j+3},\ldots\}$ is valid (that is, it verifies the condition on $\sum_{j=1}^{i} f_j$ for all values of i), then $\{f_1,\ldots,f_j,1,0,f_{j+3},\ldots\}$ is also considered valid. f_0 is not considered because it does not affect the value of n_i (from (20)). We will say that $\{f_1,f_2,\ldots,f_{i-1},f_i\}$ is the *expansion* of r (in fractionnal basis ρ) if it is valid and $r = \sum_{j=1}^{i} f_j \rho^{i-j}$.

Let $j \geq 1$; then from (21),

$$\begin{aligned} f_j + f_{j+1} &= \Big(F_l(n_{j+1},k) - F_l(n_j,k)\Big) + \Big(F_l(n_j,k) - F_l(n_{j-1},k)\Big) \\ &= F_l(n_{j+1},k) - F_l(n_{j-1},k), \end{aligned}$$

and we have $|f_j + f_{j+1}| \leq k-2$. If $f_j \geq 1$, then we have $-(k-2) \leq f_{j+1} \leq k-2-f_j$ and f_j, f_{j+1} can be replaced by $0, (f_{j+1}+f_j)$. Similarly, if $f_j \leq -1$, then f_j, f_{j+1} can be replaced by $0, (f_{j+1}+f_j)$, and all non-zero values of f_j can be moved to the right. By the same reasonning from $f_{j+1} \geq 1$ and $f_{j+1} \leq -1$, all non-zero values of f_{j+1} can be moved to the left.

What's more, replacing f_j, f_{j+1} by $(f_j - 1), (f_{j+1} + 1)$ (when possible) in the expansion of r to create the expansion of r' gives

$$r' = r - \rho^{i-j+1}(\rho + 1) = r - \rho^{i-j+1}\left(\frac{1}{k-1}\right) < r.$$

Similarly, replacing f_j, f_{j+1} by $(f_j + 1), (f_{j+1} - 1)$ gives $r' > r$. From these results, we obtain that moving negative values to the right and positive values to the left in the expansion of r is allowed and increasses its value, while moving positive values to the right and negative values to the left is also allowed and decreasses its value.

It is now possible to determine an upper bound for n_i. We define r as $(k-1)(n_i - n_0\rho^i)$ and, from (20), we have $r = \sum_{j=1}^{i} f_j \rho^{i-j}$, that is the expansion of r is $\{f_1, f_2, \ldots, f_{i-1}, f_i\}$. By moving all negative values to the right until they are canceled or placed on f_i, and then moving all the remaining positive values onto f_1, we obtain r' with $r' \geq r$ (under the current conditions, it is possible that no movement my be needed). The greatest possible value of f_1' is $k-2$ while f_i' is between 0 and $-(k-2)$ and all other f_j' are 0. Therefore we have

$$r \leq r' = f_1'\rho^{i-1} + f_i' \leq f_1'\rho^{i-1} \leq (k-2)\rho^{i-1},$$

and, replacing r,

$$(k-1)(n_i - n_0\rho^i) \leq (k-2)\rho^{i-1},$$

which gives the upper bound

$$n_i \leq n_0\rho^i + \frac{k-2}{k-1}\rho^{i-1} = \left(n_0 + \frac{k-2}{k}\right)\rho^i.$$

The lower bound is found using a similar process. Q.E.D.

Proposition: *Let $n > n_0 \geq k$. Then the integer $i \in \mathbb{N}$ for which $n_i \leq n < n_{i+1}$ verifies*

$$\log_\rho\left(\frac{kn}{kn_0 + k - 2}\right) - 1 < i \leq \log_\rho\left(\frac{kn}{kn_0 - k + 2}\right). \tag{23}$$

Proof: To determine the upper bound we use the first inequality of (22) and $n \leq n_i$. We have

$$n \geq \left(n_0 - \frac{k-2}{k}\right)\rho^i.$$

But $n_0 - \frac{k-2}{k} > n_0 - 1 \geq 1$ since $n_0 \geq k \geq 2$; hence

$$\rho^i \leq n\frac{1}{\left(n_0 - \frac{k-2}{k}\right)} = \frac{kn}{kn_0 - k + 2}.$$

And since $\rho = \frac{k}{k-1} > 1$, we have

$$i = \log_\rho\left(\rho^i\right) \leq \log_\rho\left(\frac{kn}{kn_0 - k + 2}\right).$$

The lower bound is found in the same way from the second inequality of (22) and $n < n_{i+1}$. Q.E.D.

Theorem: *Let $n > n_0 \geq k$. Then it will take between $\left[\log_\rho\left(\frac{n}{n_0}\right)\right] - 1$ and $\left[\log_\rho\left(\frac{n}{n_0}\right)\right] + 1$ iterations from n_0 to find n_i which verifies $n_i \leq n < n_{i+1}$.*

Proof: To determine the upper bound we note that if

$$0 \leq \log_\rho\left(\frac{kn}{kn_0 - k + 2}\right) - \log_\rho\left(\frac{n}{n_0}\right) < 1$$

is proven, we obtain $i < \log_\rho\left(\frac{n}{n_0}\right) + 1$ from the second inequality of (23). Since

$$1 < \frac{n}{n_0} \leq \frac{kn}{kn_0 - k + 2}$$

for all $k \geq 2$, and $\rho > 1$, the difference is greater or equal to 0. We also have:
$\log_\rho\left(\frac{kn}{kn_0-k+2}\right) - \log_\rho\left(\frac{n}{n_0}\right) < 1$

$$\begin{aligned}
&\iff \quad \frac{\log\left(\frac{kn}{kn_0-k+2}\right) - \log\left(\frac{n}{n_0}\right)}{\log\left(\frac{k}{k-1}\right)} < 1 \\
&\iff \quad \frac{\log\left(\frac{kn_0}{kn_0-k+2}\right)}{\log\left(\frac{k}{k-1}\right)} < 1 \\
&\iff \quad \log\left(\frac{k^2}{k^2-k+2}\right) < \log\left(\frac{k}{k-1}\right) \\
&\iff \quad \frac{kn_0}{kn_0-k+2} < \frac{k}{k-1} \\
&\iff \quad n_0(k-1) < kn_0 - k + 2.
\end{aligned}$$

And since $n_0(k-1) = kn_0 - n_0 < kn_0 - k + 2$ for $n_0 \geq k$, we have $i < \log_\rho\left(\frac{n}{n_0}\right) + 1$. But i is an integer and for any integer n satifying $n > n_0 \geq k$, $\frac{n}{n_0}$ cannot be an integer power of ρ, hence $i \leq \left[\log_\rho\left(\frac{n}{n_0}\right)\right] + 1$ and the upper bound is proven. The lower bound is found in the same way,

using the first inequality of (23) and $k^2 n_0 - k(n_0 - k) - 3k + 2 < k^2 n_0$ for all $n_0 \geq k \geq 2$. Q.E.D.

For large values of n compared to k it is therefore more efficient to assume that the first $\left\lfloor \log_\rho \left(\frac{n}{n_0}\right) \right\rfloor - 1$ iterations will not give $n_i > n$ and make sure that $n_{i+1} < n$ only before doing the next two iterations (since no more then $\left\lfloor \log_\rho \left(\frac{n}{n_0}\right) \right\rfloor + 1$ iterations are needed).

4 New Problem

One problem encountered when trying to apply the process on which the Josephus problem is based is to get one player to sit in each position. To illustrate this suppose you take 13 persons (who have never heard of the Josephus problem) and tell them you will remove every 4^{th} person and give ten thousand dollars to the last person removed; it is very likely there will be no volunteer to sit in the 4^{th}, 8^{th} or 12^{th} place. To solve this problem, you could proceed as follows:

Given n players labeled from 1 to n placed consecutively around a circular table and k a positive integer (usually between 1 and $n-1$). Suppose that after removing the players as in the Josephus problem, the last player removed is placed in the first position on a second table. The remaining $n-1$ players take back their original positions on the first table and are removed once again, the last one removed taking the first empty position on the second table. This process is repeated until all the players have been placed on the second table. Then they are removed (this time from the second table) and the last player removed is declared the winner. The problem is then to predict beforehand the initial position $G(n,k)$ (on the first table) of the last remaining player on the second table.

Exemple: Let $n = 6$ and $k = 4$, using this new process, the players are placed on the second table in the following order (starting from the first position): $5, 1, 3, 4, 2, 6$; and player 2 will be the last one removed.

Algorithm: The first step is to compute $F(n,k)$ using Tait's algorithm and saving every value of $F(j,k)$. We begin with $G_{n-F(n,k)+1}$ defined by

$$G_{n-F(n,k)+1} = F\Big(n - F(n,k) + 1, k\Big) \tag{24}$$

and compute G_j for f from $n - F(n,k) + 2$ to n using

$$G_j = \begin{cases} G_{j-1} + 1 & \text{if } F(j,k) \leq G_{j-1} \\ G_{j-1} & \text{if } F(j,k) > G_{j-1} \end{cases} \tag{25}$$

and finish with $G(n,k) = G_n$.

Proof: It is easily seen that the last player removed from the second table is in the $F(n,k)^{\text{th}}$ position on that table, that is, the last player removed from the first table when $F(n,k)-1$ players had already been moved to the second table. But this player must have been the $F\big(n-\Big(F(n,k)-1\Big),k\big)^{\text{th}}$ of the $n-\Big(F(n,k)-1\Big)$ players remaining at the first table. We define G_j as the position of this player among the remaining j players when there were $n-j$ players on the second table (equation (24) is then obvious). To prove (25), we go from the moment when j players remained on the first table (that is when $n-j$ players had been placed on the second table) to the moment when $j-1$ players remained, with $j > n-F(n,k)+1$. This is equivalent to moving player $F(j,k)$ to the second table. Since $j > n-F(n,k)+1$, the player in the ${G_j}^{\text{th}}$ position is not the last one removed and $G_j \neq F(j,k)$. Moving the $F(j,k)^{\text{th}}$ player to the second table will not change the position of player G_j unless $F(j,k)$ is smaller than G_j, under which condition G_j will decrease by one. That is:

$$G_{j-1} = \begin{cases} G_j - 1 & \text{if } F(j,k) < G_j, \\ G_j & \text{if } F(j,k) > G_j, \end{cases}$$

which can be writen as (25). Q.E.D.

References

[1] C.G. Bachet de Mézirac., *Problèmes plaisants et délectables qui se font par les nombres*, 1612.

[2] L. Euler, *Observationes circa novum et singulare progressionum genus*, Opera Omnia Series Prima, Opera Mathematica **7**, pp. 246-261, 1923.

[3] P.G. Tait, *Collected Scientific Papers*, volume 2, Cambridge, 1900, pp. 432-435 = Proc. Roy. Soc. Edinburgh **22**, pp. 165-168, 1899.

[4] W.W.R. Ball and H.S.M. Coxeter, *Mathematical Recreations and Essays*, 12^{th} edition, University of Toronto Press, 1974, ISBN 0–8020–6138–9.

[5] K. Burde, *Das Problem der Abzählreime und Zahlentwicklungen mit gebrochenen Basen*, J. Number Theory **26**, pp. 192-209, 1987.

[6] A.M. Odlyzko and H.S. Wilf, *Functional Iteration and the Josephus Problem*, Glasgow Mathematical Journal **33**, pp. 235-240, 1991.

5 Addendum

Generalized Tait (returns $F_l(n,k)$)

```
tait:=proc(n,k,l)
local i,F:
if n<l then
    F:=0:                          returns 0 if F_l(n,k) is not defined
else
    F:=modp(k-1,l)+1:
    for i from l+1 to n do
        F:=modp(F+k-1,i)+1:
    od:
fi:
F:
end:
```

Generalized Burde (returns $F_l(n,k)$)

```
burde:=proc(n,k,l)
local n_i,F,f,i,imin:
if n<l then
    F:=0:                          returns 0 if F_l(n,k) is not defined
else
    if k=1 then
        F:=n-(l-1):
    else
        if n<=k then
            F:=modp(k-1,l)+1:
            for i from l+1 to n do
                F:=modp(F+k-1,i)+1:
            od:
        else
            if k>l then
                n_i:=k:
                f:=modp(k-1,l)+1:
                for i from l+1 to k do
                    f:=modp(f+k-1,i)+1:
                od:
            else
                f:=modp(-l,k-1)+1:
                n_i:=(f+k*(l-1))/(k-1):
            fi:
            if n<n_i then
                F:=(n-(l-1))*k:
```

```
            else
                F:=f:
                f:=modp(F-n_i-1,k-1)+1-F:
                imin:=floor(log(n/n_i)/log(k/(k-1))):
                for i from 1 to imin do
                    n_i:=(n_i*k+f)/(k-1):
                    F:=F+f:
                    f:=modp(F-n_i-1,k-1)+1-F:
                od:
                for i from imin+1 while (n_i*k+f)<=(n*(k-1)) do
                    n_i:=(n_i*k+f)/(k-1):
                    F:=F+f:
                    f:=modp(F-n_i-1,k-1)+1-F:
                od:
                F:=F+k*(n-n_i):
            fi:
        fi:
    fi:
    F:
    end:
```

New Problem (returns $G(n,k)$)

```
    extended:=proc(n,k)
    local F,G,i:
    F[1]:=1:
    for i from 2 to n do
        F[i]:=modp(F[i-1]+k-1,i)+1:
    od:
    G:=F[n-F[n]+1]:
    for i from n-F[n]+2 to n do
        if G>=F[i] then
            G:=G+1:
        fi:
    od:
    G:
    end:
```

Nicolas Thériault, Dept. of Mathematics, University of Toronto, Toronto, Canada M5S 3G3