The Taxman Game

Robert K. Moniot

September 5, 2003

1 Introduction

Want to know how to beat the taxman? Legally, that is? Read on, and we will explore this cute little mathematical game.

The taxman game is a "golden oldie" computer game. It has been used as an exercise in introductory programming classes and in books on recreational computing since time immemorial. (Well, at least since the 1970s. I haven't been able to determine who invented it or when.)

Here's how to play: you start with a pot consisting of all the positive integers up to some chosen limit, N. You take one, and the taxman gets all the others that divide it evenly. The number you picked is added to your score, and the numbers the taxman got are added to his score. This process is repeated until there's nothing left in the pot. What makes the game interesting is that the taxman has to get something on every turn, so you can't pick a number that has no divisors left. And when none of the numbers remaining in the pot have any divisors left, the taxman gets them all!

The game is pretty easy to program on a computer, and if programming is one of your skills, you might want to pause now to do so and play a few games, just to get the hang of it. Right now we'll demo a couple of example games with $N = 10$ to show you how it goes. First, a game where we try to win by always taking the largest number in the pot.

```
Hi!  I'm the taxman!  Let's start.
How many numbers in the pot? 10
The available numbers are:
1 2 3 4 5 6 7 8 9 10
Pick one of the numbers: 10
I take: 1 2 5
The available numbers are:
3 4 6 7 8 9
Pick one of the numbers: 9
I take: 3
The available numbers are:
4 6 7 8
Pick one of the numbers: 8
I take: 4
I take: 6 7
All gone.  My score = 28, your score = 27
I win.
```

As you can see, it's not so easy to beat the taxman. You do have an advantage because each number you take is always bigger than any of the numbers the taxman takes, and so it's not hard to get out in front of the taxman and stay there while play continues. But slowly and silently the unpickable numbers with no divisors left are accumulating, and suddenly the game ends with the taxman's score getting a big boost. If you haven't accumulated enough of a cushion, he'll jump ahead and beat you. In this game, the numbers 6 and 7 ended up with no factors left, so the taxman got them, boosting his score from 15 to 28.

Still, if you ponder your moves well, it is possible to beat the taxman. This time we'll choose more carefully.

```
Hi!  I'm the taxman!  Let's start.
How many numbers in the pot? 10
The available numbers are:
1 2 3 4 5 6 7 8 9 10
Pick one of the numbers: 7
I take: 1
The available numbers are:
2 3 4 5 6 8 9 10
Pick one of the numbers: 9
I take: 3
The available numbers are:
2 4 5 6 8 10
Pick one of the numbers: 6
I take: 2
The available numbers are:
4 5 8 10
Pick one of the numbers: 8
I take: 4
The available numbers are:
5 10
Pick one of the numbers: 10
I take: 5
All gone.  My score = 15, your score = 40
You win.
```

In this game, we limited the taxman's take to just one number on each move, and no numbers were left unpickable at the end. It's not always possible to trounce the taxman this badly, but it is generally possible to win. Below, we will explore some strategies for making good picks. We will also see the results of a lengthy computer search for the best possible sequences of picks for pots of up to 49 numbers.

2 Some basics of strategy

First of all, it is worthwhile noticing that since each of the N numbers originally in the pot ends up in either the player's score or the taxman's score, the sum of the two scores must equal the sum of the numbers from 1 to N. Therefore the bigger the taxman's score, the smaller your score, and vice-versa. This kind of game is called a "zero-sum game," because the sum of the two players' final scores, minus a constant that is known at the start of play, is equal to zero.

The taxman always gets the number 1 and all but at most one of the primes that are initially in the pot. This is because the number 1 can never be picked, and the only turn on which the player can pick a prime number is the first turn, when the number 1 is still in the pot. This gives the taxman a built-in advantage. There is no simple formula for this advantage, but a crude estimate can be calculated by using the approximation for the density of primes near x of $1/\ln x$. We will skip the details of the calculation, but the result is that for a pot of size N, the taxman is guaranteed to get roughly $1/\ln N$ of the pot just from the leftover primes. This fraction slowly decreases as N increases: it is about 30% for $N = 30$, and about 15% for $N = 1000$.

On the other hand, we can ask what is the absolute best that the player can do in any game? Since the taxman must get something on every turn, the best you can hope for is to take just as many numbers as the taxman does. This can only be done for even N. In that case, the best score is gotten if you take all largest numbers and leave the smallest ones for the taxman. (We were in fact able to do this in the second example game played above, with $N = 10$.) There is a well-known formula giving the sum of the numbers from 1 to N as $N(N+1)/2$. So for even N, if the player takes all the numbers from $\frac{N}{2}+1$ through N, leaving the taxman the numbers 1 through $\frac{N}{2}$, a little algebra shows that the taxman's score will be $N(N+2)/8$. Since the pot is $N(N+1)/2$, the taxman's take as a fraction of the pot is $\frac{1}{4}\frac{(N+2)}{(N+1)}$. This fraction is always a bit bigger than 1/4, and gets closer to 1/4 or 25% as N gets larger. This means that the very best the player can ever do is to get a bit less than 75% of the pot.

2.1 Optimal strategy

An *optimal strategy* for this game would be a strategy that always gives the player the largest possible score for a given N (and gives the taxman the least). What can we say about the optimal strategy?

The best choice for the first pick is not hard to see: simply take the largest *prime* number in the pot. We can prove that this is always the best pick as follows. First, this pick limits the taxman's take to 1, which the taxman will take on the first move for any pick. Further, removing this prime, which we will call $p_{\max}$, and 1 from the pot does not prevent any sequence of nonprime picks that was originally possible. That is, if the player could have played by picking the sequence of numbers $n_1, n_2, \ldots, n_k$ where k is the total number of picks and n_1 is not prime, then the alternative sequence $p_{\max}, n_1, n_2, \ldots n_k$ is also possible and gives the player a higher score. Picking $p_{\max}$ does not remove any divisors of $n_1, n_2, \ldots, n_k$ other than 1, because $p_{\max}$ cannot divide any other number in the pot. Its smallest multiple, $2p_{\max}$, cannot be in the pot, because there is a theorem from number theory that says that there is always a prime between $p_{\max}$ and $2p_{\max}$, which would contradict the choice of $p_{\max}$ as the largest prime in the pot. Since none of the picks in the original sequence is prime, they must therefore all have other divisors in the pot that will still allow them to be picked.

On the other hand, if n_1 is prime and $n_1 < p_{\max}$, then by a similar argument replacing n_1 by $p_{\max}$ does not require any change in the rest of the sequence, and increases the player's score by $p_{\max} - n_1$.

From this result, it follows that if N itself is prime, then the optimal sequence of picks for a pot of size N is the same as for a pot of size $N-1$, except that on the first turn we should pick N instead of the next-smaller prime.

3 A heuristic strategy

Beyond the first pick, there don't seem to be any simple rules for the optimal strategy. In situations like this, we can often be happy to settle for a strategy that may not be optimal but will generally give a high score. Such a strategy, which is not proved to be the best but is sufficiently successful in practice, is called a *heuristic.* Heuristics can be based on various principles. For the taxman game, a "greedy" principle yields a very good heuristic that is easy to apply. A greedy principle is one that says to make the move that maximizes the short-term gain. For some problems, a greedy strategy can turn out to be optimal, but in many cases it falls short because of the long-term consequences. (The same is true of life in general, isn't it?)

In the first example game we saw a naive greedy heuristic that failed even to win the game: always picking the largest available number in the pot. This is a bad heuristic because it ignores the taxman's take. The game is zero-sum, so the true measure of gain is not your take alone, but the difference between your take (the value of the number you pick) and the taxman's take (the sum of the remaining divisors of the number).

Therefore, a better greedy heuristic for the taxman game is on each turn to pick the number that maximizes the difference between your pick and the taxman's take for that pick. This heuristic is easy to calculate (at least, easy for a computer) and generally gives a good score.

Trials of the greedy method show that it sometimes overlooks "freebies," which are picks that can be taken in between greedy picks without altering the remaining sequence. The freebies are numbers for which the taxman's take is a proper subset of his take for the greedy pick on the same turn. That is, if you picked the freebie, the taxman would take some, but not all, of the divisors of the greedy pick, so the greedy pick could still be taken afterward. Thus the greedy pick would either give the freebie to the taxman or make it unpickable subsequently, whereas picking the freebie before the greedy pick does not prevent the greedy pick. (The freebie also must not be a divisor of any other number in the greedy sequence, lest it make that number unpickable.)

For example, when $N = 15$, after first picking 13, the largest prime, the greedy pick is 15. The remaining divisors of 15 are 3 and 5. This means that we can pick 9, giving the taxman 3, and still pick 15, giving the taxman 5. This makes 9 a freebie.

Therefore the greedy method can be improved by scanning for freebies before each greedy pick, and inserting them into the sequence. We call this the "improved greedy" heuristic, and it is the best strategy that we have found so far. But is it optimal? Alas, only sometimes. The experiments described later show that the improved greedy heuristic often falls short of the optimal score.

4 Search for optimal plays

Since there does not seem to be any simple strategy that guarantees an optimal score, we programmed a computer to search for the optimal play for any given N. In essence, the program tries all possible choices for the first pick. Then, based on what is left in the pot, it tries all permissible choices for the second pick, and so on, continuing each sequence of picks till the game is ended. Having played all possible games for that N, it prints out a sequence of picks that gives player the best score. (Here, I say "*a* sequence" rather than "*the* sequence" because there can be different sequences that achieve the same score.)

This type of approach is called a "brute-force search" because it simply tries all possibilities. Usually, when trying to find an optimal solution to a problem, you should apply as much mathematical analysis as possible first, in order to find a strategy that homes in directly on the solution. A brute-force search usually requires a lot of computing power, so that it may be impractical for large problems.

Our search was organized as a "tree search," in which it is thought of as exploring the branches of a tree. The trunk of the tree is the starting point, when all the numbers from 1 to N are in the pot. Then each of the main branches of the tree represents one of the possible choices for the first pick. Having chosen one of these main branches, further branches represent the available choices at each step. After traversing a number of branches, the search eventually reaches a leaf, where a game ends. The search then backs up to the last branch point and picks another branch to follow. When all the branches from that point have been explored, it backs up further and tries another branch closer to the trunk. This process is repeated until all the branches have been explored and every leaf has been visited.

4.1 Time complexity

As N gets bigger, the brute-force search takes an ever-longer time to explore the whole tree of possibilities and find the optimal score. Let's call the number of possible pick sequences $S(N)$. This number grows rapidly as N increases. The rate of growth of $S(N)$ is called the "time complexity" of the brute-force search method, since it determines how many possibilities the program has to examine in order to arrive at the answer. The term "complexity" in computer science is used to describe the level of difficulty of solving a given class of problems. A linear rate of growth, where the complexity is proportional to N, means the problem is easy and can be readily solved even for large values of N. Unfortunately, our $S(N)$ grows very rapidly, limiting the maximum value of N for which a solution can be found in a reasonable time.

We can easily see that $S(N)$ grows less rapidly than $N!$ because, although we have almost N possible choices for the first pick, after the first pick there are fewer than $S(N-1)$ sequences for the rest of the game. Still, this line of thinking suggests that $S(N)$ probably grows more quickly than an exponential function such as 2^N, since the number of sequences usually more than doubles as N is increased by 1.

We used our search program to count $S(N)$ for pot sizes up to $N = 35$. The result is shown in Figure 1 (filled circles). Notice that the vertical scale of the graph is logarithmic. On this kind of graph, an exponential curve would be a straight line. The upward curvature of the trend in the figure shows that $S(N)$ grows somewhat faster than exponentially.

4.2 Improving the brute-force search

The straightforward tree search is too lengthy to solve for the optimal play for pots bigger than about $N = 40$. So the program was improved to "prune the tree." This pruning is done by keeping track of the unpickable numbers, those whose divisors are already taken. These numbers already belong to the taxman, although they are still in the pot. By adding these numbers to the taxman's score, the program can tell when it

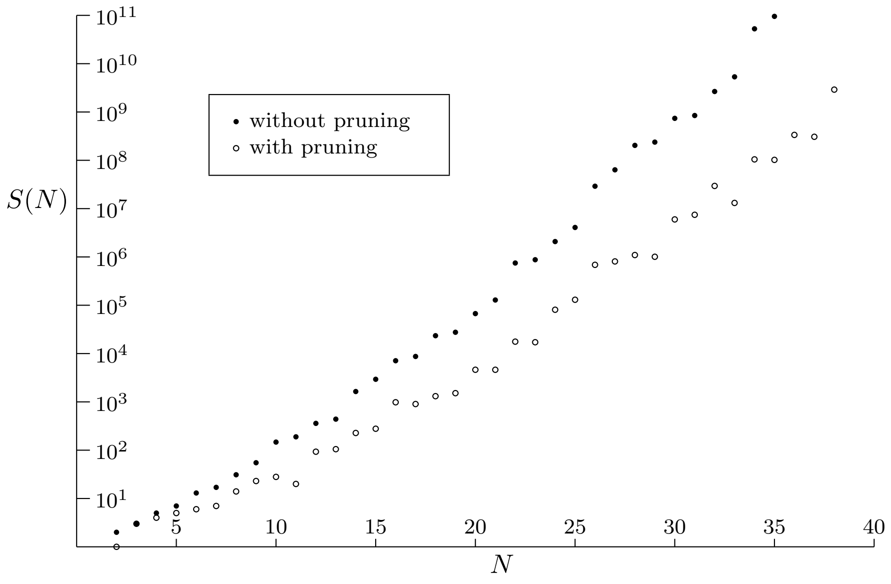

Figure 1: Number $S(N)$ of pick sequences as a function of initial pot size N, with and without pruning.

is impossible for the player to do better than the best score that has already been found, and abandon that branch of the tree. The program uses the score from the improved greedy heuristic as an initial target, and replaces this by any higher score that is found later on. The improvement with pruning is shown by the open circles in Figure 1. The reduction in the number of sequences examined is by a factor of nearly 1000 for $N = 35$. Still, even with pruning, the number of possibilities that need to be checked follows a trend that increases faster than exponentially with N.

To speed the search still further, the program uses the proven optimal strategy for the first pick: instead of trying all numbers from 2 to N, the program starts by picking the largest prime in the pot. This prunes away all the first branches of the tree except one.

The program also skips the search entirely whenever N is prime. As shown earlier, the optimal play for prime N is easily found from the optimal play for $N - 1$.

5 Results of the search

The search for the optimal sequence of picks was carried out for all pots up to $N = 49$ before I reached the limit of my computing resources. (Most of the search was done over a period of about 5 years on a Pentium 90 running the Linux operating system, with the final cases being done on a 1.8 GHz Pentium Linux system in a few additional months. For the case $N = 49$, the search needed to examine more than 3×10^{12} possible sequences.) The results are given in the Appendix.

One interesting question that we can look at using these results is: how often can we achieve the absolute best score, in which we take all the numbers in the upper half of the pot? The search shows that this is possible only for the cases $N = 2$, 4, 6, and 10. For $N = 8$, and for all N above 10, there are at least two primes in the upper half of the pot, so the taxman always gets at least one of them. For $N = 10$, the player's score is $40/55 \approx 73\%$ of the pot. This is close to the upper limit of 75% derived in section 2. It is probably the best score, as a fraction of the pot, that can be achieved for any pot size.

Another question is: how well does the improved greedy heuristic do as compared to the optimal play? The results show that the heuristic fails to yield the optimal play for about half the cases solved. Still, it always does a good job. Since the improved greedy heuristic can be calculated quickly, I explored how well it does out to $N = 1000$. In this range, it always makes the optimal first pick and it always wins (except for the case $N = 3$ in which the optimal score is a tie). On average, it gives the player 60% of the pot.

If you look closely at the experimental results you can see an interesting feature: the optimal *second* pick is almost always the largest *square* of a prime. The only cases where this pick is not optimal are $N = 8$ and $N = 20$. (Some of the sequences could be rearranged to pick it later than second and still get the same score.)

6 Open questions

There are many questions about the taxman game that I haven't been able to answer. Here are a few.

- Does the greedy heuristic always make the optimal first pick, namely the largest prime? It seems likely that it does, since the largest prime is usually one of the biggest numbers in the pot, and so the difference between it and 1 (the taxman's take for that pick), is going to be hard to beat. I have tested it out to $N = 500,000$ without finding any cases where it fails to make the correct first pick. Still, I don't have a downright proof.

- Is there a simple *winning strategy* for the game? This would be a strategy that is guaranteed to win every time. The improved greedy heuristic is a good candidate, and tests show that it wins all games up to $N = 1000$, but I have not proved that it always wins.

- Is the choice of the largest squared prime as the second pick optimal for all N above 20? If not, how can we identify the values of N for which it is not?

- We saw how the optimal play for prime N can be based on the optimal play for $N - 1$. Are there any other cases for which the optimal play for one pot size can be used to find the optimal play for another? The experiments show that very often the optimal sequences for successive N are quite similar, but there isn't an obvious systematic way to relate them.

If you can answer any of these questions, be sure to let me know. Perhaps you can also think of some interesting questions of your own! In any case, I hope you have had fun exploring with me how to beat the taxman!

7 Appendix

Listing of pick sequences found by brute-force search program. The table lists the improved greedy sequence first, with the "freebies" in boldface, followed by the optimal sequence, if different, on the next line.

N	**Score**	**Sequence**
2	2	2
3	3	3
4	7	3 4
5	9	5 4
6	15	5 4 6
7	17	7 4 6
8	21	7 8 6
9	30	7 9 6 8
10	40	7 9 6 10 8
11	44	11 9 6 10 8
12	48	11 9 6 12 10
	50	11 9 10 8 12
13	50	13 9 6 12 10
	52	13 9 10 8 12
14	66	13 9 14 10 8 12
15	81	13 **9** 15 10 14 8 12
16	89	13 **9** 15 10 14 16 12
17	93	17 **9** 15 10 14 16 12
18	102	17 15 10 14 16 12 18
	111	17 9 15 10 18 14 12 16
19	104	19 15 10 14 16 12 18
	113	19 9 15 10 18 14 12 16
20	124	19 15 10 20 16 14 12 18
21	135	19 21 14 15 20 16 12 18
	144	19 9 21 15 14 18 12 20 16
22	157	19 21 14 22 15 20 16 12 18
	166	19 9 21 15 14 22 18 12 20 16

N	**Score**	**Sequence**
23	161	23 21 14 22 15 20 16 12 18
	170	23 9 21 15 14 22 18 12 20 16
24	173	23 21 14 22 15 20 16 24 18
	182	23 9 21 15 14 22 20 18 16 24
25	198	23 25 15 21 14 22 20 16 24 18
26	224	23 25 15 21 14 26 22 20 16 24 18
27	247	23 25 **15** 27 21 14 26 18 22 12 24 20
	251	23 25 15 27 21 14 26 22 20 18 16 24
28	279	23 25 **15** 27 21 14 28 26 18 22 20 16 24
29	285	29 25 **15** 27 21 14 28 26 18 22 20 16 24
30	300	29 25 27 21 14 28 26 18 22 20 30 16 24
	301	29 25 26 22 15 27 21 18 30 20 28 16 24
31	302	31 25 27 21 14 28 26 18 22 20 30 16 24
	303	31 25 26 22 15 27 21 18 30 20 28 16 24
32	318	31 25 27 21 14 28 26 18 22 20 30 32 24
	319	31 25 26 22 15 27 21 18 30 20 32 28 24
33	337	31 25 33 22 27 21 26 18 28 20 30 32 24
	352	31 25 15 33 27 22 26 21 18 30 20 32 28 24
34	371	31 25 33 22 27 34 21 26 18 28 20 30 32 24
	386	31 25 15 33 27 22 34 26 21 18 30 20 32 28 24
35	406	31 **25** 35 **21** 33 22 27 34 26 18 28 20 30 32 24
	418	31 25 35 21 33 27 22 34 26 18 12 28 24 32 20 30
36	442	31 **25** 35 **21** 33 22 27 34 26 18 36 24 32 28 20 30
37	448	37 **25** 35 **21** 33 22 27 34 26 18 36 24 32 28 20 30
38	486	37 **25** 35 **21** 33 22 38 27 34 26 18 36 24 32 28 20 30
39	478	37 39 26 **21** 35 33 38 27 34 18 36 24 32 28 20 30
	503	37 25 35 21 39 33 27 26 38 34 28 20 18 36 30 24 32
40	498	37 39 26 **21** 35 33 38 27 34 18 36 24 32 28 40 30
	525	37 25 35 21 39 33 27 26 38 34 30 20 40 32 28 24 36
41	502	41 39 26 **21** 35 33 38 27 34 18 36 24 32 28 40 30
	529	41 25 35 21 39 33 27 26 38 34 30 20 40 32 28 24 36
42	548	41 39 26 **25** 35 33 38 27 34 18 36 24 32 28 42 40 30
	571	41 25 35 21 39 33 27 26 42 38 34 28 20 40 32 30 24 36

N	**Score**	**Sequence**
43	550	43 39 26 **25** 35 33 38 27 34 18 36 24 32 28 42 40 30
	573	43 25 35 21 39 33 27 26 42 38 34 28 20 40 32 30 24 36
44	596	43 39 26 **25** 35 33 38 44 27 34 28 42 20 40 32 30 24 36
	617	43 25 35 21 39 33 27 26 44 38 34 28 42 20 40 32 30 24 36
45	621	43 39 26 **25** 35 33 **27** 45 38 44 34 30 28 42 40 32 24 36
	660	43 25 35 21 39 33 27 45 26 38 34 18 42 30 28 44 36 24 40 32
46	667	43 39 26 46 **25** 35 33 **27** 45 38 44 34 30 28 42 40 32 24 36
	706	43 25 35 21 39 33 27 45 26 46 38 34 18 42 30 28 44 36 24 40 32
47	671	47 39 26 46 **25** 35 33 **27** 45 38 44 34 30 28 42 40 32 24 36
	710	47 25 35 21 39 33 27 45 26 46 38 34 18 42 30 28 44 36 24 40 32
48	695	47 39 26 46 **25** 35 33 **27** 45 38 44 34 30 28 42 40 32 48 36
	734	47 25 35 21 39 33 27 45 26 46 38 34 18 42 30 28 44 40 36 32 48
49	719	47 49 35 39 26 46 33 27 45 38 44 34 30 28 42 40 32 48 36
	758	47 49 35 21 39 33 27 45 26 46 38 34 18 42 30 28 44 40 36 32 48