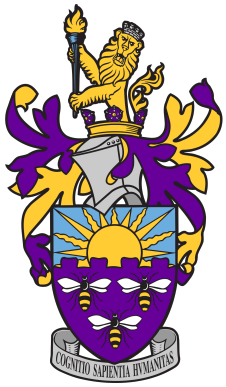
The Urinal Problem

Joss Moffatt



University of Manchester

Christmas 2019

Abstract

Consider a person walks into a male bathroom with n urinals and Φ is the set of free urinals and Ξ is the set of occupied ones, where should this person go to maximise their privacy? I demonstrate an algorithm to select the most appropriate urinal in linear time and propose a new extension to the original problem.

The Urinal Problem

Joss Moffatt

January 15, 2020

Contents

1 Introduction

Typically when walking into the men's bathroom urinals appear on the immediate wall, evenly spaced and without separators providing instant privacy. Under these constraints, a somewhat natural heuristic takes effect and we use this to perform the selection of which urinal we wish to go to. In the short mockumentary *"Male Restroom Etiquette"* by Phil R. Rice [Zarathustra Studios, 2006] the section called Urinal Selection explores the common selection heuristic:

> "Always select the urinal that is as far away as possible from men who are using other urinals."

In pursuing this approach, assuming P_1 is at urinal 1 and there are 5 urinals, then P_2 would try and maximise his distance from P_1 and this results in him picking urinal 5. If a third person P_3 enters he has to maximise the distance between him, P_1 and P_2 so naturally picks urinal 3. This results in the configuration seen in Figure 1.

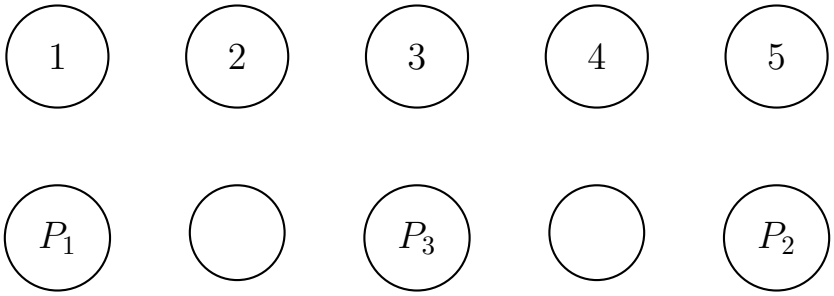


Figure 1: Uncompleted for $n = 5$

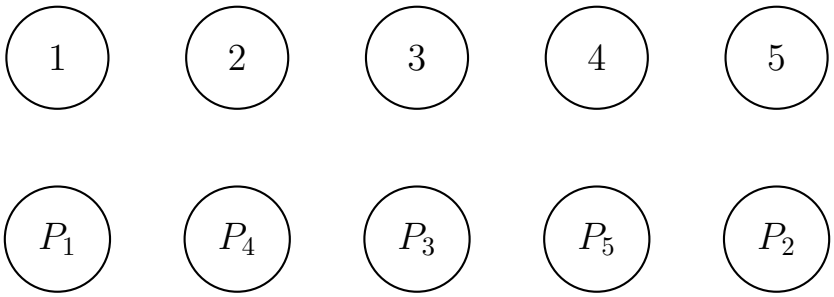


Figure 2: Completed for $n = 5$

However we now fall into a problem. Now if P_4 tries to select a urinal, both urinal 2 and 4 are the optimal choices, therefore we should define an extension to our original heuristic to always allow a full configuration of people at urinals.

Due to the laziness of most people, if there is a set of urinals that are equally optimal, then a person will choose the closest one to the door. Consequently our updated heuristic is:

> "Always select the urinal that is as far away as possible from men who are using other urinals. If there are multiple optimal urinals, choose the closest one to the door."

If we presume the distance to the door is proportional to the number of the toilet our final configuration for $n = 5$ is that seen in Figure 2.

2 Formalising the Selection

Clearly, under our current definition, we have a set of urinals this set is compromised of 2 disjoint sets of urinals. These sets are the occupied urinals and the free urinals. Consequently, we can now formally define these sets.

Set of Urinals

The set of all urinals U is equal to the union of the disjoint sets, the set of free urinals Φ and the set of occupied urinals Ξ:

- $U = \Phi \cup \Xi$.
- $\Phi \cap \Xi = \emptyset$.
- $|\Phi| \geq 0$.
- $|\Xi| \leq |U|$.
- $|\Xi| + |\Phi| \equiv |U|$

Further analysing the current definition it is clear that a function must be defined that takes a selected urinal position and the occupied urinals and returns the number of urinals away between the selected one and its nearest occupied neighbours. We will define this as the adjacent neighbour distance function.

2.1 Adjacent Neighbour Distance Function

Adjacent Neighbour Distance Function

The Adjacent Neighbour Distance function $\delta(u, \Xi)$:

- u is the selected urinal, $u \in \Phi$.
- Ξ is the set of occupied urinals.
- $\delta(u, \Xi) = \min(\{x \mid x = |n - u|, \forall n \in \Xi\})$.
- $\delta(u, \Xi) \in [1, n]$.

The Adjacent Neighbour Function returns the minimum distance from the selected urinal to its closest occupied neighbour. Therefore our definition of δ is valid as we are getting the absolute distance from the selected urinal to all the others and then finding the minimum value from this newly constructed set of distances. This works clearly as the minimum absolute distance from the all the occupied urinals will be its closest adjacent neighbour and hence the closest adjacent distance. □

Next, we want to find the set of urinals that maximise the adjacent neighbour distance function in order to maximise the person's privacy. Then we wish to find the minimum of the resulting set such that one may satisfy the closest to the door clause in the updated heuristic definition. Taking into account all of this, we can then produce our selection function.

2.2 Selection Function

Using Schmidt's definition for the *argmax* function [Schmidt, 2016]:

> We define the $argmax$ of a function f defined on a set D as:
> $\arg\max_{x \in D} f(x) = \{x \mid f(x) \geq f(y), \forall y \in D\}$

Selection Function

The Selection function $\sigma(\Xi, \Phi)$:

- Ξ is the set of occupied urinals.
- Φ is the set of free urinals.
- $\sigma(\Xi) = \min(\arg\max_{u \in \Phi} \delta(u, \Xi))$.
- $\sigma(\Xi) \in \Phi$.

Proving this algorithm is correct again is trivial, we are finding the argument that maximises our *Adjacent Neighbour Distance Function* and then getting the

smallest of these arguments. This is then the closest urinal to the door that maximises distance between adjacent neighbours which is the optimal selection given our updated heuristic definition. □

3 Algorithms

3.1 Quadratic Time Algorithm

One can simply convert our algorithms into python for it to resemble our *Selection function*. In our algorithm F represents Φ and O represents Ξ so by iterating over Φ and recording the greatest distances from the *Adjacent Neighbour Distance function* and then finding the minimal from the output set which is the same process as our σ function so our algorithm is valid.

```
def adjacent_neighbour_distance(u, O):
  for index, value in enumerate(O):
    if value > u:
      # O(1) due to the constant size
      return min(abs(O[index - 1] - u), abs(O[index] - u))
  return min(abs(O[- 1] - u), abs(O[-2] - u))

def selection(O, F):
  max_args = []
  max_dist = 0
  for index, u in enumerate(F):
    distance = adjacent_neighbour_distance(u, O)
    if distance > max_dist:
      max_dist = distance
      max_args = [u]
    elif distance == max_dist:
      max_args.append(u)
    else:
      continue
  return min(max_args)
```

However, one can clearly see that our function representing δ is $O(p)$ where p is the size of the input set Ξ. This is as you iterate over the input array until the current urinal number is greater than the selected urinal and then returns the minimum distance between the selected urinal and the urinals surrounding it. By considering the condition to check that the current urinal number is greater than the selected one allows the best case time complexity to be $O(1)$ however, worst and average case complexity is still $O(p)$.

In the selection algorithm we then iterate over the input set and if we say the size of Φ is m then we perform the iteration within the selection function m times. In this loop we only do constant time operations apart from the distance calculation, which we previously established is $O(p)$. By applying the product rule for asymptopic analysis the loop has an overall complexity of $O(pm)$. We then return the minimum of the resulting final set which has a worst case time complexity of $O(m)$ which only occurs when the $\arg\max_{u\in\Phi} \delta(u, \Xi) = \Phi$ so we will then have to do m checks to find the minimum value. Finally we can conclude that $\sigma \in O(pm)$ but as p and m are completely reliant on the total number of urinals n due to the equality $|\Xi| + |\Phi| \equiv |U|$ we can conclude that σ runs in quadratic time with regards to the total number of urinals then $\sigma \in O(n^2)$

In a paper on the Urinal Problem (Kranakis and Krizanc, 2019) for a non-empty initial configuration, it is made clear you can generate an algorithm in $O(n^2)$ as we have done. But, they further conjecture that there will be an algorithm that can select the optimal urinal in linear time. However, using our definition for σ we are having to find a globally optimal solution for a function that runs in linear time already by iterating over all the values and recording the maximum. This approach will always run in quadratic time and in order to reduce our time complexity to linear time we will have to find a better way to optimise our system.

3.2 Linear Time Algorithm

Crucially, we need to find a better way of finding the optimal solution without having to check each individual value. Consequently, consider the following configuration: There exists a urinal α where the gap between it and the urinal to

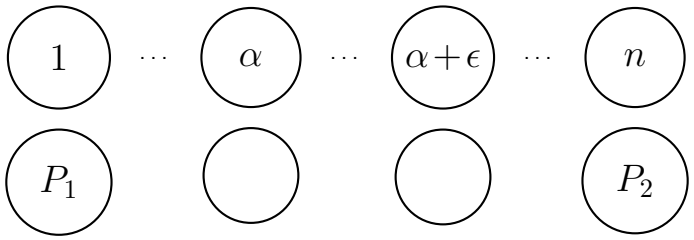


Figure 3: Configuration where largest gap between occupied urinals is ϵ

its right is ϵ is the greatest gap in the system. By considering how our heuristic was defined, we know that we are trying to maximise the distance between the person who is choosing a urinal and the adjacent neighbours. Consequently this the gap between the urinals α and $\alpha + \epsilon$ is where the person wants to go as it will guarantee to maximise their privacy. Therefore when $1,n \in \Xi$ the person will want to go at the position $\alpha + \lfloor \epsilon/2 \rfloor$ the floor function is used in order to consider the distance element of the heuristic. The value $\lfloor \epsilon/2 \rfloor$ is effectively the optimal *distance* value for the *Adjacent Neighbour Distance function* as it is the

maximum distance you can possibly achieve bewteen adjacent neighbours in the system. However, instead of taking $O(n^2)$ to solve we can reduce this to $O(n)$ by iterating over the set of occupied urinals and then finding the gap between consecutive urinals and recording α and *distance* when a *new* max *distance* is found. Note we must only record at a new max distance again in order to consider the distance element of the heuristic.

Next we must consider if there are runs of free urinals at the start starting from urinal 1, for example:

$\Phi = \{1, 2, 3, 4, 5, 7....\}$ 1,2,3,4,5 would be the run at the start as 6 is occupied

or the end of the set, ending at n:

$\Phi = \{...n-3, n-1, n\}$ n-1, n would be the run at the end as n-2 is occupied

It is important to do this as these pseudo-gaps will not show up when calculating the distance between the occupied urinals as they are clearly not occupied but are technically still gaps as they can be occupied. This is easily worked around as if there is a run of length s at the start,a run of length e at the end,if $s \geq \lfloor \epsilon/2 \rfloor$ and $s \geq e$ then it will want to go at urinal number one as it is the new distance away from any urinal and then urinal 1 is the optimal urinal. Conversely if $e > \lfloor \epsilon/2 \rfloor$ and $e > s$ then the end run is then the biggest distance away from any urinal and hence last urinal is the optimum choice.

```
def efficient_selection(O, F):
  distance = 0
  urinal_choice = -1
  alpha = 0
  for index in range(len(O) - 1):
    if distance < int((O[index+1] - O[index]) / 2):
      distance = int((O[index+1] - O[index]) / 2)
      alpha = O[index]
  start_run = O[0] - F[0]
  end_run = F[-1] - O[-1]
  if (start_run >= end_run) and (start_run >= distance):
    return  F[0]
  elif (end_run > start_run) and (end_run > distance):
    return F[-1]
  else:
    max_gap_center = alpha + distance
    closest_value = 2 * distance
```

```
for index, value in enumerate(F):
  if closest_value > abs(value - max_gap_center):
    closest_value = abs(value - max_gap_center)
    urinal_choice = F[index]
return urinal_choice
```

Analysing Figure 5, all operations are $O(1)$ apart from the loops. The first is iterated over $p-1$ times, again where p is the size of Ξ, and the second is iterated m times, where m is the size of Φ. In the best case where there is a long run at the start or end, the second loop is never even reached so we have the best case time complexity, therefore, it is $O(p)$. For the worst case it $\max(O(p), O(m))$ but as discussed before as both p and m dependant on the overall number of urinals n we have best, average and worst-case time complexity of $O(n)$ which then proves there is an algorithm to optimally choose a urinal in polynomial time.

3.3 Comparing the Algorithms

Figure 6 shows the time taken to select an optimal solution for an increasing number of urinals where the number of occupied urinals is randomly chosen and the free urinals are then calculated from that. The linear time algorithm is barely visible as it is so close to the x axis whilst the quadratic algorithm is tending towards a n^2 relationship which is what we expect, but the more efficient algorithm is clearly performing very well as the time taken is barely increasing even with 10000 urinals.

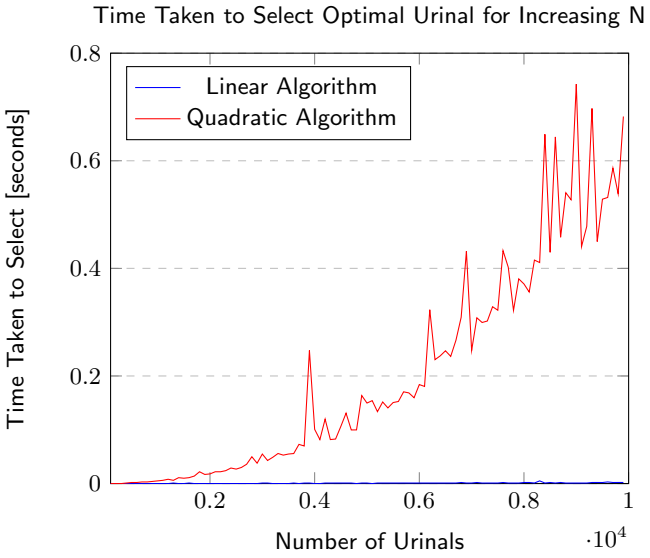


Figure 4: Graph comparing the algorithms

4 Extensions to the Theme

4.1 The Infinite Bathroom

Consider a bathroom with a total number of urinals which tends towards ∞. We then encounter the problem that in an empty configuration the first person would choose the first urinal, but then based on the algorithm, the next person would choose the last toilet. However, it would take a **very** long time to get to the last toilet in this case - you would never get there. When this happens our algorithm starts to fail as we are assuming getting to the toilet is inexpensive and we can somehow just magically teleport there. Consequently, I conjecture there is a value κ that will serve as an optimum threshold. This will mediate how the algorithm predicts the optimal urinal, as now getting to the urinal will be expensive and as the number of urinals tends towards some large number it would be more optimal to travel only a certain distance.

5 Bibliography

- Kranakis, E. and Krizanc, D. (2019). The Urinal Problem. [ebook] Available at: https://people.scs.carleton.ca/ kranakis/Papers/urinal.pdf (Accessed 24 Dec. 2019).
- Schmidt, M. (2016). Argmax and Max Calculus. [ebook] UBC, p.1. Available at: https://www.cs.ubc.ca/ schmidtm/Documents/2016_540_Argmax.pdf (Accessed 21 Dec. 2019).
- Zarathustra Studios. (2006). Male Restroom Etiquette. [Online]. available at: https://www.youtube.com/watch?v=IzO1mCAVyMw (Accessed: 21 December 2019).