This lecture we consider the problem of selecting the best of n options as they show up one at-a-time. Here we will aim to exploit randomness in the input, in particular in the order of the items showing up.

The problem. Given n numbers, say each written on a separate card. Suppose we are presented these cards one at-a-time, and we need to selected one just as it appears. What is the algorithm that maximizes the chance of selecting the highest number? Without any assumption of randomness this appears to be a really hard (or hopeless) task. For some historical reason this problem is often refereed to as the secretary selection problem. Surprisingly, one can make this probability quite high, a constant independent of the number of cards n.

1 First idea

Maybe a natural first idea is that with decent probability that the first few items are not the largest. For example, with probability 1/2 the highest item is not among the first half of the set. So we can afford to see half of the cards. In fact, with probability $\approx 1/4$ the top card is in the second half of the sequence while the second highest is in he first half. In this case, after looking at the first half, the maximum is the unique item higher than any of the numbers in the first half, so it is easy to identify.

One way to phrase this algorithm is as follows, where $r = 1/2$ and we use v_i to be the value of the ith number

```
Compute $m = \max_{1\le i\le rn} v_i$ while seeing the first $rn$ items
$i = \lfloor rn \rfloor + 1$
While $i \le n$ and no item selected
    Select $i$ if $v_i > m$
    $i \leftarrow i + 1$
Endwhile
Output $v_i$ if an item is selected
```

As we argued above:

Claim The probability that the above algorithm outputs the maximum value $\max v_i$ is at least 0.25.

Is this algorithm best possible?

- Maybe the simplest alternative to think about is the amount of time we wait before trying to look for a larger value. i.e., to pick a different value $r \neq 1/2$.
- Another important point to think about, even if the maximum value is not the unique item bigger than m in the second part, we can get lucky, if the maximum is the first larger item that appears.

2 Better Analysis of the Algorithm

We start with an improved analysis of the algorithm above, and then optimize the value r for best performance. A nice idea to make the analysis simpler is to separate out cases depending on what position the maximum value item is. For each position i the probability that the maximum item is in position i is $1/n$, as the sorting of items is uniform random.

Consider now the special case that the maximum item is in position i. If $i \le rn$, then we will definitely not select the maximum item. The following simple claim is the key to the analysis:

Claim. The algorithm will select the maximum item when it is in position $i > rn$ if and only if the maximum value among the $i-1$ items that are in the first $i-1$ position is in one of the rn first position.

Proof. This condition is clearly both necessary and sufficient. Assume that the maximum item among the first $i-1$ position has value v_j. If this occurs in one of the rn first positions than $m = v_j$, and the first item of value at least v_j is the maximum item in position i. If this is not the case, then $m < v_j$, and either v_j or an item even earlier will be picked before the algorithm reached position i.

Now we are ready to analyze the probability of success. Assume for simplicity of notation that rn is integer.

$$\begin{aligned} Pr(\text{Algorithm pick max item}) &= \sum_{i=rn+1}^{n} \frac{1}{n} Pr(\text{Algorithm pick max item}|\text{max item is in position } i) \\ &= \sum_{i=rn+1}^{n} \frac{1}{n}\frac{rn}{i-1} = r \sum_{i=rn+1}^{n} \frac{1}{i-1} \end{aligned}$$

Now using the close bound that

$$\sum_{i=rn}^{n-1} \frac{1}{i} \geq \int_{rn}^{n} \frac{1}{x} dx = \ln n - \ln(rn) = -\ln r$$

Using this approximation we get that

$$Pr(\text{Algorithm pick max item}) \geq -r \ln r$$

Based on this formula, the best r is the one maximizing $-r\ln r$, which we get by setting the derivative to 0, that is solving $-\ln r - 1 = 0$, which gives us $r = 1/e = 0.368$.

With this $r = n/e$, the probability of picking the maximum value is

$$\frac{rn}{n} \sum_{i=rn}^{n-1} \frac{1}{i} \geq \int_{rn}^{n} \geq \frac{1}{e} \int_{n/e-2}^{n} 1/x dx \approx 1/2.$$

Giving is the following claim.

Claim The algorithm stated above, with $r = 1/e \approx 0.37$ will select the maximum element with probability ≈ 0.37.

We