Online Algorithms

There are many cases in the real world in which the algorithm does not know the entire input yet, but still has to make partial decisions about the output. Algorithms which have to make their decisions gradually as data arrives are called online algorithms.

The input can even be generated by an adversary that creates new input portions based on the system's reactions to previous ones. We seek algorithms that have a provably good performance.

Competitive Ratio

We define the competitive ratio for online algorithms to capture how much worse the algorithm does compared to one that knows about the future.

Let $C_{\text{OPT}}$ be the optimum cost of offline algorithm, and $C_{\text{ALG}}$ - the cost of online algorithm, then

$$\text{CR} = \frac{C_{\text{ALG}}}{C_{\text{OPT}}}$$

An ALG is $c$-competitive, if $\exists \delta$, that $\forall I$

$$C_{\text{ALG}}(I) \leq c \cdot C_{\text{OPT}}(I) + \delta$$

Plan:

Buy or Rent
List Update
Cat and Mouse

Online Algorithms

Formally, an online algorithm receives a sequence of requests $\sigma(1), \ldots, \sigma(m)$.

When serving request $\sigma(t)$, an online algorithm does not know requests $\sigma(t')$ with $t' > t$.

Serving requests incurs cost and the goal is to minimize the total cost paid on the entire request sequence.

Going Skiing!

Buy - 500$
Rent - 50$/ day

What to do, buy or rent?

Strategy 1: Buy instantly
Cost: 500$

Worst case: Go only once.
Optimal Cost: 50$
CR: 10
Going Skiing!

Buy - 500$
Rent - 50$/ day

What to do, buy or rent?

Strategy 2: Always rent (say, for n days)
Cost: 50∙n$

Worst case: Go forever
Optimal Cost: 500$
CR: n/10 (infinity)

Going Skiing!

In general, r is the cost to rent, p is the cost to buy
Claim, ∃ deterministic algorithm s.t.

\[ CR \geq 2 - \frac{r}{p} \]

Strategy, if (d+1) r ≤ p rent, then buy.

Proof.
Case 1. \( C_{OPT} = p \), i.e. \( (d+1) r \geq p \) or \( d r \geq p - r \)

\[ CR = \frac{d r + p}{p} \geq \frac{p - r + p}{p} = 2 - \frac{r}{p} \]

List Update

Here, we focus on accessing the elements of a linked list of the size n. Specifically, if the k-th element of the list is accessed, then the cost incurred for this access is k.

The algorithm can also exchange any two consecutive items at a cost of 1.

The goal is to devise and analyze an on-line algorithm for doing accesses with a small competitive factor.
List Update

**TRANSPOSITION heuristic:** always move the most recently accessed element one position forward (to front), by swapping it with its neighbor.

**Claim:** CR is $\Omega(n)$

Consider a list: head, ..., x, y
Then access last two elements in the alternating sequence of size $2t$: x, y, x, y, ..., x, y

$C_{ALG} = t \cdot (n + 1 + n + 1)$

TRANSPOSITION

**Claim:** CR is $\Omega(n)$

We could move x and y to front on the first call

$C_{OPT} = \frac{2t(n+1)}{4n+12}\Omega(n), t \rightarrow \infty$

List Update

**FREQUENCY COUNT heuristic:**
Maintain a frequency of access for each item. Keep the list sorted by decreasing frequency.

**Claim:** CR is $\Omega(n)$

We could construct a sequence in the following way: access the first element $k > n$ times, the second - $(k - 1)$ times, and finally the last element - $(k - n + 1)$ times.

Observe, the FC heuristic will never reorganize the list.

FREQUENCY COUNT

We could construct a sequence in the following way: access the first element $k > n$ times, the second - $(k - 1)$ times, and finally the last element - $(k - n + 1)$ times.

$C_{ALG} = k + 2(k-1) + \ldots + n(k-n+1)$

$\geq (k-n)(1 + 2 + \ldots + n) = \Omega(kn^2)$

The OPT sol. will move each element to the front, when accessed for the first time

$C_{OPT} = \Omega(kn)$

$CR = \Omega(n), k \rightarrow \infty$

List Update

**MOVE TO FRONT heuristic:** Always move the most recently accessed element to the front of the list.

**Claim:** MTF is 4-competitive.

Consider a potential function

$\Phi(k)=2(t)$ (the number of inversions between B & MTF)

An inversion is a pair of distinct elements that appear in one order in one list and in a different order in the other list. Example, a, c, d, e, the number of inversions is 3.

MOVE TO FRONT

$S = \text{all items before } j \text{ in MTF and before } j \text{ in B}$

$T = \text{all items before } j \text{ in MTF and after } j \text{ in B}$

Then, $C_{MTF}(j) = 1 + |S| + |T| + (|S| + |T|)$

$C_{OPT}(j) \geq 1 + |S|$ because all of S is before j in OPT.
**Cat and Mouse Game**

There is one cat and one mouse which has \( n \) hiding places.

A cat has a sequence of probes for finding a mouse.

At each time step, the cat comes to one of the places.

If it’s the one a mouse is hiding in, the mouse has to move. The cost is the number of times the mouse has moved.

Find a good strategy for the mouse.

**Deterministic Approach**

What is the optimal offline algorithm?

Find a hole which is the far away - longest forward distance.

The cost of the online algorithm is the length of the probing sequence, mouse will just match the sequence.

So, the competitive ratio is linear.

**List Update**

It is possible to prove that no deterministic online algorithm can achieve a competitive ratio less that 2.

**Splay Trees**

Sleator and Tarjan conjectured that the splay tree algorithm has a bounded competitive ratio.

**MOVE TO FRONT**

\[ \Delta \Phi = 2(|S| - |T|) \]

\[ AC_{MTF} = 1 + 2(|S| + |T|) + 2(|S| - |T|) = 1 + 4|S| \]

Now, we find a change in potential. Moving \( j \) to front, will eliminate \( T \) inversions and create \( S \) new inversions.

\[ AC_{MTF} = 1 + 2(|S| + |T|) + 2(|S| - |T|) = 1 + 4|S| \]

Finally, we add up over all requests

\[ \sum_{j=1}^{n} C_{MTF}(j) + \Delta \Phi(j) \leq 4 \sum_{j=1}^{n} C_{OPT}(j) \]

\[ C_{MTF} - \Phi(0) + \Phi(n-1) \leq 4C_{OPT} \]

\[ C_{MTF} \leq 4C_{OPT} - \Phi(n-1) \leq 4C_{OPT} \]

**QED**
Need to think randomly

Strategy: Randomly pick a hole each time.

Let us assume the following sequence of cat's visits \((1, 2, \ldots, n-1)\) and so on.

Optimal would've been to just sit at \(n\), cost = 1

Any other place would eventually force a move until the mouse chooses \(n^{th}\) hole.

Probability of choosing \(n^{th}\) hole: \(1/n\). Expected number of moves until then: \(n\).

Need to think randomly

It follows from,

\[
E = \frac{1}{n} (1) + \frac{n-1}{n} (1+E)
\]

\[
E = 1 + \frac{n-1}{n} E
\]

\[
E = n
\]

Danny can do better

In the next lecture....