# LECTURE 24 ( DECEMBER 2nd)

## Online Algorithms

Problem not computationally hard but the algorithm doesnot have all the information upfront.

Input arrives over time & we need to make a decision on the fly ← this decision can not be changed later
without knowing what will happen

**Goal:** Perform well relative to an omniscient algorithm ← Similar to appx. algs.

COMPETITIVE RATIO → Online alg. is c-competitive if

$$\forall \text{ inputs } x \quad \underbrace{ALG(x)}_{\text{cost}} \leq c \cdot \underbrace{OPT(x)}_{\text{cost}}$$

How to maximize your chances of finding the perfect partner ?

Suppose that the people you date arrive in a random order and you don't know who you are going to see next

If you date someone and break up, you can't date them again

At some point, you decide this is the person I want to date forever

What strategy should you choose ?

- Date the first person forever — maybe you will never see the best partner
- Keep breaking up forever — end up alone

This is also sometimes called the online secretary problem and has been studied for a long time.

**Best Strategy** In lecture

Let's see another problem

## Rent or Buy

Rent ski for $r/day

Buy skis for $b/day & keep them forever

Ski as many days as possible but don't know how many more days of the season will be viable for skiing

Each day wake up & see if weather is still good
At some point ski season is over

**Goal:** Decide whether to rent/buy skis each day
& minimize total amount of money you spend

**Ex 1:**

| | |
|---|---|
| Rental | $50 |
| Buy | $500 |

} If we know the future:
If ≥ 10 days of good weather, buy
ow rent

But how do we do this without knowing the future

Let's try some simple strategies:

① **Always immediately buy** — Buy on the first day if weather is good

Worst-case input — we only ski once

Optimal offline alg - $50
we pay - $500

10-competitive

② **Rent forever** Never buy, just rent

Worst-case input — ski season goes on arbitrarily long
we pay — ∞
Competitive ratio — ∞

Once we buy, no costs so we can characterize any online alg by the day on which it decides to buy.

For any such alg, worst-case input — weather is bad on the day after & remains so

With this in mind, consider one more bad strategy

③ Rent five times, then buy

└ Worst-case input — weather good for six days, then bad

We pay — $5 \times 50 + 500 = 750$

Optimal alg — always rents — $6 \times 50 = 300$

2.5 competitive

So, if we hedge our bets by renting longer, we get a better competitive ratio. At some point, this stops being true though.

In particular, it never makes sense to plan to rent for more than 10 days. because then we should have just bought the skis.

Most hedging we can do — rent for 10 days, then buy

This is the better-late-than-never algorithm.

└ Rent for $\frac{b}{r} - 1$ days, then buy, i.e. buy on day $\frac{b}{r}$

**Theorem** This is 2-competitive.

**Proof** Suppose weather is good for $n$ days.

**Case 1** If $nr < b$, then opt. soln. is to always rent but our alg. doesn't buy either, so it is optimal

**Case 2** If $nr \geq b$, the opt. soln buys immediately but we rent $\frac{b}{r} - 1$ days before buying

$$\text{Ratio} = \frac{\left(\frac{b}{r} - 1\right) r + b}{b} = \frac{b - r + b}{b} = 2 - \frac{r}{b} \leq 2$$

Can we do better? No! [Left as exercise]
↳ for det. algs.

## List Update Problem

List of $n$ items $\{1, \ldots n\}$

Operations ① Access $(x)$ — Traverse to $x$ in the list. Cost = position of $x$

② Swap$(x,y)$ — Swap two adjacent elements $(x,y)$ — Cost = 1

Goal: Process a sequence of access requests at min possible cost.

Example ① No swaps Competitive ratio = $n$

Worst-case input — Access $(n) \times t$

$$ALG = nt$$

$$OPT = n - 1 + t$$

② Single exchange Move accessed item one closer to the front

Worst case input Access $(n) \times n$ times
Access $(1)$

$$ALG \rightarrow n + n-1 + \cdots + 1 + n = \theta(n^2)$$

$$OPT \rightarrow n + 2$$

$$\text{Ratio} = \Omega(n)$$

③ Freq. count — Keep list in sorted order by how many times each element is accessed (most accessed at front)

Access(1) ×n
Acccess(2) ×n
⋮
Access(n) ×n

$ALG = n(1+2+\cdots+n) = \Theta(n^3)$

$OPT \le 2n \cdot n = \Theta(n^2)$

**Move-to-Front Algorithm** — After accessing (x), move it to the front of the list

**Thm** MTF is 4-competitive

**Pf** via potential functions.

MTF alg.
B — best offline alg.
**claim** $C_{MTF}(\sigma) \le 4 \cdot C_B(\sigma)$ $\forall$ input $\sigma$

Amortized $C_{MTF}(\sigma) \le 4 \cdot C_B(\sigma)$
So, amortized cost of each operation is at most 4 times the cost of operation in B

How to compare the relative costs of the two algorithms?
└ How diff their two lists are currently?

$\Phi$ — high when list are very diff
low — similar

$\Phi_t = 2($# inversions b/w MTF's list & B's at time $t)$
↳ pair (x,y) s.t. x before y in one list & after y in another list

e.g. B = 1, 2, 3, 4, 5
MTF = 2, 5, 3, 1, 4
(2,3) is not an inversion
(1,3) is an inversion

Note that $\Phi_0 = 0$

Suppose we are at some step $t$, & $C_{MTF}(t)$ is the cost at this step
& $C_B(t)$

Define $ac_{MTF}(t) = c^{(t)}_{MTF} + \phi_t - \phi_{t-1}$ ← amortized cost

Note : If we add up all amortized cost, then we have

$C_{MTF}$ + $\underbrace{\phi_T}_{\geq 0}$ so total cost is always smaller than sum of amortized costs

↓
Total cost

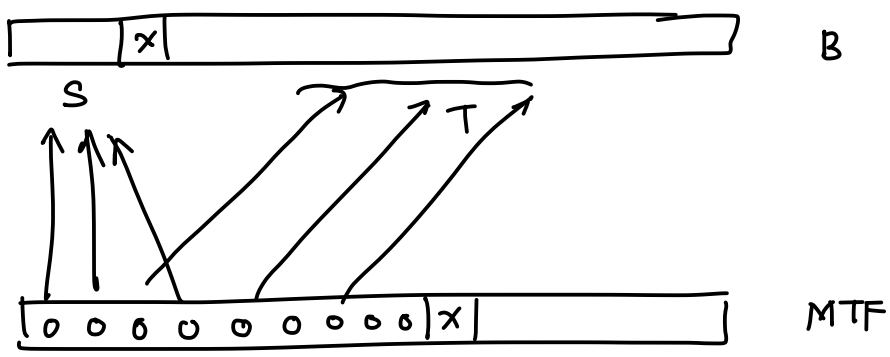
We will show that $ac_{MTF}(t) \leq 4 \cdot C_B(t) \Rightarrow$ This gives $C_{MTF} \leq 4 \cdot C_B$

## Analysis of Access(x)

Suppose we do access(x) at this step
and MTF brings x to the front
Suppose B does not swap x here

find cost

$C_{MTF} = 1 + \overbrace{|S| + |T|}$
$+ \underbrace{|S| + |T|}_{\text{swap cost}}$
$= 2|S| + 2|T| + 1$




$C_B \geq \underbrace{1 + |S|}$
since S lies before

$S = \{$ items before x in MTF & before x in B $\}$

$T = \{$ items before x in MTF & after x in B $\}$

#inversions increases by 1 for elements in S
dec. — by 1 — T

What happens to the potential ? Only changes involve pairs with x in it

$$\Delta\phi = \phi_t - \phi_{t-1} = 2 \times (|S| - |T|)$$

$$ac_{MTF}(t) = c^{(t)}_{MTF} + \Delta\phi = 1 + 2|S| + 2|T| + 2|S| - 2|T|$$
$$= 1 + 4|S| \leq 4(1 + |S|) \leq 4 C_B(t)$$

Above we assumed, B did not swap x

## Analysis of B swaps

We can view swaps of B as a separate operation & analyze it separately

$c_{MTF}(t) = 0$ & $c_B(t) = 1$ here

$\Delta\phi \leq 2$ since swap may introduce one new inversion

So, $ac_{MTF}(t) \leq 2c_B(t) \leq 4c_B(t)$

Overall, we get $\sum_t ac_{MTF}(t) = c_{MTF} + \underbrace{\phi_T}_{\geq 0} - \phi_0 \leq 4c_B$

$\Rightarrow c_{MTF} \leq 4c_B$