Lecture 20:
Online Algorithms

Goals for today

- Understand the **motivation** and **definition** of **online** algorithms
- See some examples of online algorithms and their analyses:
    - The **rent-or-buy** problem
    - The **list update** problem
    - Using **potential functions** to analyze online algorithms

Motivation: Don’t have all the information

- Recall: **Approximation algorithms** settle for a “pretty good” solution because the problem is too computationally hard

- Today, we will also settle for “pretty good” solutions, but for a different reason.
  - Your algorithm gets an input fed to it over time (very similar to streaming!)
  - It **can not see the future,** but must **make a decision anyway**
  - “Pretty good” performance is measured by comparing **against an optimal omnipotent algorithm** (it can see the future!)

Formal Definition

Definition (c-competitive algorithm):

$$ALG \leq c \cdot OPT$$

(for all inputs)

c is called the competitive ratio

Rent-or-buy

Problem: You want to go skiing every day for snow season.

- **Costs** $\$r$ to **rent** a pair of skis, or $\$b$ to **buy** a pair of skis ($r per day)
- **Issue**: Don't know how long snow season will last
- **Goal**: Decide each day whether to rent or buy.
- **Example**: Renting costs $50 and buying costs $500

Strat: Buy immediately: $500 / 50 = 10$-competitive

Rent forever: $50 \times n / 500 \to \infty$

Good strategies

Observation: All strategies can be characterized by "buy on day k"

Question: What is the worst-case input?

Season ends on $k+1$

Example: Buy on day 6. $50 \times 5 + 500 = 750$

$OPT = 300$ $\quad c = \frac{750}{300} = 2.5$

Example: Buy on day 10. $50 \times 9 + 500 = 950$

$OPT = 500$ $\quad c = 1.9$

The best strategy

Strategy (Better-late-than-never): Buy on day $\frac{b}{r}$

Claim: *Better-late-than-never* is 2-competitive

Proof: n is length of season

Case 1: $nr < b$: ALG = OPT

Case 2: $nr \geq b$

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

The best strategy

Claim: *Better-late-than-never* is optimal for deterministic algorithms

Proof: Case: k more times

$$\frac{(\frac{b}{r}-1)r+kr+b}{b} > \frac{(\frac{b}{r}-1)r+b}{b}$$

Case: k fewer times

$$\frac{(\frac{b}{r}-1)r-kr+b}{b-kr} = \frac{2b-r-kr}{b-kr} = 2+\frac{kr-r}{b-kr} > 2.$$

Summary of rent-or-buy

- Argued that all strategies are "buy on day k" for some k
- The ***Better-late-than-never*** algorithm buys on day $\frac{b}{r}$
  - This is point where buying would have been optimal in hindsight
- Better-late-than-never is **2-competitive**
- Argued that better-late-than-never is optimal

List update

Problem: We have a list of n items $\{1,2,\ldots,n\}$ and two operations

- **Access**(x): Traverse to x in the list. The cost is the position of x
- **Swap**(x,y): Swap any two adjacent elements x and y. Costs 1

Goal: Process a sequence of Access requests at minimum possible cost

Example: Do no swaps. What is the competitive ratio?

Worst case Always Access (n)

$ALG = n \times t$

$OPT = n - 1 + t$

$c = n$.

More examples

Example: Single-exchange. Move accessed item one closer to front

What's the competitive ratio?

Access $n-1, n, \ldots$

$$C \approx \frac{n}{1.5} = \Omega(n)$$

More examples

Example: Frequency count. Count frequency of access for each item. Keep list sorted by frequency

What's the competitive ratio?

$n \times 1, \quad n \times 2, \quad \ldots \quad n \times n$

$ALG = \Theta(n^3)$

$OPT = \Theta(n^2)$

Okay, time for a good algorithm

Algorithm (Move-to-front): After Access(x), swap x towards the front

Claim: **Move-to-front** is a 4-competitive algorithm

Proof: call the competitor B,

$$\Phi = 2\,(\text{The \# of inversions between ALG's list and B's list})$$

Proof continued…

Two key steps:

1. Analyze the AC of Access(x) of MTF
   Goal: $AC \leq 4 \cdot C_B$
2. Account for the cost of swaps of B
   (because this affects potential!!)

Proof continued…

Notation/setup:

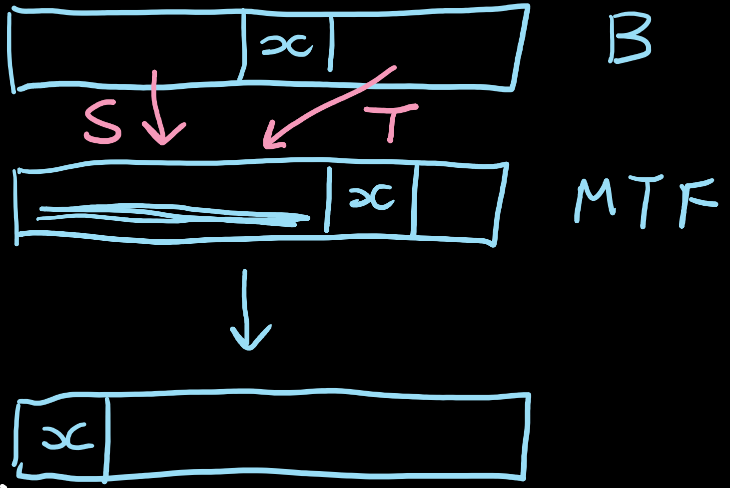
Let C_{MTF} = actual cost of MTF, C_B = actual cost of B

Let $AC_{MTF} = C_{MTF} + \Delta\Phi = C_{MTF} + \Phi_{new} - \Phi_{old}$

Goal: Show $AC_{MTF} \leq 4 \cdot C_B$

Proof continued…

Analysis of Access(x):



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

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

$$C_{MTF} = \underbrace{1 + |S| + |T|}_{\text{find}} + \underbrace{|S| + |T|}_{\text{swaps}} = 1 + 2(|S| + |T|)$$

$$\boxed{C_B \geq 1 + |S|}$$

$$\Delta\Phi = 2(|S| - |T|)$$

Proof continued…

Analysis of Access(x) continued…

$$
\begin{aligned}
AC_{MTF} &= C_{MTF} + \Delta\Phi \\
&= 1 + 2(|S| + |T|) + 2(|S| - |T|) \\
&= 1 + 4|S| \\
&\leq 4(1 + |S|) \\
&\leq 4C_B
\end{aligned}
$$

Proof continued…

Analysis of B swapping:

$C_{MTF} = 0 \qquad C_B = 1$

$\Delta\Phi \leq 2$

$AC_{MTF} \leq 0 + 2 = 2 = 2 \cdot C_B \leq 4 \cdot C_B$

Proof continued…

Putting it together:

$$\text{Total MTF cost} = \sum AC_{MTF} + \cancelto{0}{\Phi_{initial}} - \cancelto{\geq 0}{\Phi_{final}}$$

$$\leq \sum AC_{MTF}$$

$$\leq \sum 4 \cdot C_B$$

$$= 4\,(\text{Total cost of } B)$$

$\Rightarrow$ MTF is 4-competitive

Summary

- We defined **online algorithms**, algorithms that must make decisions without knowing the future (the full input)
- The **Rent-or-buy** problem as an example
- The **list-update** problem as an example
- Important: ***Potential functions*** were super useful for analyzing the list-update algorithm!!