Often: assume algo design is a one-shot problem
- take input
- produce output

But in this case already we've seen that algorithms require decision-making over time
- input is revealed incrementally
- take decisions over time
- decisions made with partial information (don't know the future)
- so "information-theoretic" barriers to the quality of solution we can get.

Examples: data structure problems
  don't know what queries will come next

  Saw paging in an "offline" setting where the entire request sequence was known up-front and we computed the optimal page eviction policy

  But what if we don't know the requests in future?

Today: give model for these problems
  show some examples.
Here are some examples of interest

1. Ski Rental
2. List Update
3. Paging
4. Optimal Search

Model : competitive analysis

1. Input revealed in timesteps

\[ I = (r_1, r_2, r_3, \ldots, r_k, r_f) \]

2. On seeing \( r_t \), have to make some decision

\[ \text{to "satisfy the request"} \]

3. cost (\( Ayo(I) \)) = total cost if \( Ayo \)

4. cost (\( OPT(I) \)) = best way to solve problem if known entire \( I \).
competitive ratio on input $I = \frac{\text{cost}(\text{Alg}(I))}{\text{cost}(\text{OPT}(I))}$

comp. ratio of Alg = $\max_{I} \frac{\text{cost}(\text{Alg}(I))}{\text{cost}(\text{OPT}(I))}$

What is the "price of uncertainty"? Is not being able to see the future?

will show that some natural ideas can give good bounds in this very demanding worst-case setting

Also see the power of randomization (again)
Two comparisons

(A) **streaming**

\[
\text{streaming problem are typically simpler}
\]
\[
\text{but work in more restrictive model}
\]
\[
(\text{low space, fast runtimes})
\]

**online**: no restriction on space/time, just try to understand lack of information

(B) **regret analysis**

will see in next lecture.

* Typically regret = additive gap

* vs.

  \[
  \text{comp. ratio = multiplicative gap}
  \]

* But in regret we compare to best fixed decision in hindsight.

  \[
  \text{In C.R. compare to best adaptive solution in hindsight.}
  \]
For Ski-rental and ListUpdate see lecture notes from (451).

**Paging**: Have \( n \) pages in main memory

fast cache of size \( k \leq n \)

@ each time \( t \) get a request for page \( x_t \)

- if not in cache, must move into cache before returning its contents
- if cache full, must evict a page

Which page to evict?

What is "eviction policy"?

Randomized Agy: (Random Marking)

Each page in cache is marked or unmarked.

1) At start of phase, all pages in cache unmarked.
2) Get next page request \( x_t \)

   - if \( x_t \) in cache, mark the page
   - else
     - if all pages marked, unmark all (start new phase)
     - evict random unmarked page
     - bring in \( x_t \), mark it
Fact: In each phase, have \( k+1 \) distinct requests.

So \( \text{OPT} \) should have paid \( \geq 1 \) eviction somewhere.

Theorem: \( E[\text{\# evictions made by Random Marking on seq } I] \)

\[ \leq O(\log k) \cdot \text{\# evictions (OPT (I))}. \]

Here's an idea of the proof:

\begin{itemize}
  \item Sps. start of phase, have pages 1, 2, \ldots, \( k \) in cache.
  \item then get requests \( k+1, 1, 2, \ldots, k-1 \)
\end{itemize}

What is optimum? 1 eviction (just evict \( k \))

What will random marking do?

\begin{itemize}
  \item \( 1, 2, \ldots, k \)
  \item now if we request \( k+1 \), it will evict a random page.
  \item This is expected to be half way through the request sequence.
\end{itemize}

\begin{itemize}
  \item Say we evict \( k/2 \). Then when we request \( k/2 \), we evict a random remaining unmarked page.
  \item Say we evict \( 3k/4 \) now. Etc.
\end{itemize}

Overall we'll evict \( O(\log k) \) pages.
Formally: the expected # of evictions = \( \ln k \).

Same as this problem from HW 9 (exercise).

(c) An airplane in Politesville has \( n \) seats, and \( n \) passengers assigned to these seats. The first passenger to board is a bit confused, and sits down at a uniformly random seat. The rest of the passengers do the following: when they board, if their assigned seat is free they sit in it, else (being too polite) they choose a uniformly random empty seat and sit in it. (a) What is the probability that the last person to board sits in their assigned seat? (b) What is the expected number of people who board to find their assigned seat occupied?