Online Algorithms
- Algorithms that make decisions without full knowledge of input.
- Full knowledge of past
- No knowledge of future.

- How to analyze? Two common approaches:
  - Competitive analysis (multiplicative)
  - Regret (additive)

Ski-rental
- You are about to go skiing for the first time. You need to decide between renting skis, say for $50 or buying skis for $500.
- What do you do?
- How about on your second time?
- Third time?

- What if you knew in advance exactly how many times you’d go skiing?

Rent until you’ve paid the cost of buying
- Has competitive ratio 2.

- An online algorithm $A$ is $c$-competitive if there is a constant $b$ such that for all possible input sequences $\sigma$
  $$A(\sigma) \leq c \OPT(\sigma) + b$$

Example 2: scheduling
- $m$ identical machines
- Sequence of jobs arrives one at a time.
- Upon arrival, learn processing time of job $j$.
- But don’t know the future jobs.
- Goal: minimize the makespan (maximum load = sum of processing times on any machine)

Example
- $m=3$, sequence of processing times: 7, 3, 4, 5, 6, 10

  makespan = 17

Most obvious algorithm
- Greedy: schedule the job on the least loaded machine.
- Theorem: This algorithm is $2-1/m$ competitive.
List update

• Using a linked list to store a set of items.
• Implementation of Dictionary data structure.

Storing a set $S$ of items, each with key.
Operations supported
Insert($k$) – add item with key $k$ to the set $S$
Find ($k$) – is the key $k$ in $S$?
Delete ($k$) – remove the key $k$ from $S$.

Find($k$)

• Search through the list until find the key (or reach the end)

• Cost model:
  – if item is at position $j$ in list, cost is $j$
  – Can move requested element anywhere in first $j$ positions for free.
  – Can exchange adjacent items for free at cost $1/\text{swap}$.

Online algorithm for list update?

• Move To Front:
  – On each request, move the requested element to the front of the list.

• Theorem: MTF is 2-competitive

In class exercise

• Doubly linked list: Find the element, given a pointer somewhere in the list.