ONLINE ALGORITHMS & APPLICATIONS

Prof. Navneet Goyal
Department of Computer Science & Information Systems
BITS, Pilani
Topics

- Online Algorithms
- Offline Algorithms
- Competitive Analysis
- Adversaries
- Applications
In online computation a computer algorithm must decide how to act on incoming items of information without any knowledge of future inputs.

- How should the next call be routed?
- Which cache block to be removed when the cache is full?
An **online algorithm** is one that can process its input piece-by-piece, without having the entire input available from the start.

In contrast, an **offline algorithm** is given the whole problem data from the beginning and is required to output an answer which solves the problem at hand.

For example, **selection sort** requires that the entire list be given before it can sort it.
An algorithm is called “online” if it produces (partial) output while still reading its input.

Some algorithms must be online, because they produce a stream of output for a stream of input; output is produced while the input (which might even be infinite in length) is being read.

All scheduling algorithms are online algorithms.

When an OS is paging memory, or when a dispatcher is dispatching ambulances around the city, it is often important to be able to guarantee certain levels of performance.

OS or dispatcher have no idea what happens next.

Must decide strictly according to data available at the time of the action taken.
Online Algorithms

- Input is revealed to the algorithm incrementally
- Output is produced incrementally
- Some output must be produced before the entire input is known to the algorithm
- How to make decisions with partial information?
- Unknown information: the future.
APPLICATIONS

- **Resource Allocation**
  - Scheduling
  - Memory Management
  - Routing

- **Robot Motion Planning**
  - Exploring an unknown terrain
  - Finding a destination
METHODS OF ANALYSIS

- **Probabilistic Analysis**
  - Assume a distribution generating the input.
  - Find an algorithm which minimizes the expected cost of the algorithm.

- **Pros**: can incorporate information predicting the future.

- **Cons**: can be difficult to determine probability distributions accurately.
Methods of Analysis

- **Competitive Analysis (Worst Case)**
  - For any input, the cost of our online algorithm is never worse than ‘c’ times the cost of the optimal offline algorithm.

- **Pros:** can make very robust statements about the performance of a strategy.

- **Cons:** results tend to be pessimistic.
Finding a shortest path in a finite connected graph when the graph is unknown and the algorithm receives the node neighbors only when it "enters" the node.

Problem can not be solved optimally without a simple exhaustive search.

New performance measures have to be introduced, such as competitive analysis, which compares the performance of an online algorithm with that of a hypothetical offline algorithm that knows the entire input in advance.
EXAMPLE: SKI RENTAL*

- Suppose you decide to learn to ski
- After each trip, you will make an irrevocable decision whether to stop skiing or continue learning
- You have no idea in advance what your decision will be
- Skiing is an equipment-intensive sport and before each trip you have two options: rent the equipment at $x per day or buy the equipment for a grand sum of $y such that:
  
  \[ y = cx \text{ for some integer } c > 1. \]
- Before each trip to the mountains you have to decide whether to rent or buy

* Example taken from ‘An Introduction to Competitive Analysis for Online Optimization’ Maurice Queyranne, University of British Columbia.
EXAMPLE: SKI RENTAL

- OBJECTIVE: to minimize cost

- Buying equipment even before taking one lesson would be a terrible waste if you decide to stop after the first trip

- On the other hand, if you take many trips then at some point it would be cheaper to buy than rent.

- At what point you should stop renting and buy?
EXAMPLE: SKI RENTAL

- There is some number $t$ of ski trips that you will take before stopping.
- Suppose you are told $t$ in advance.
- Then it is easy to decide: rent or buy.
- If $tx \leq y$, then rent otherwise buy right at the start.
- OFFLINE ski-rental problem.
- Its solution is called the OPTIMAL SOLUTION and the cost of optimal solution is called OPTIMAL COST.
- Optimal cost is $tx$ for $t \leq c$ and $y$ for $t > c$. 
EXAMPLE: SKI RENTAL

- In the online version of the problem, the rent or buy decision must be made prior to each trip, without knowledge of t
- Strategy: rent until \( c = \frac{y}{x} \) trips have occurred, and then buy if a \((c+1)^{st}\) trip happens
- How well this strategy would do?
EXAMPLE: SKI RENTAL

- If \( t \leq c \), then it is optimal – minimum possible amount is spent.
- If \( t > c \), then the cost is exactly twice the optimal cost!
- The strategy can be optimal for some situations and in the worst case it incurs a cost that is twice the optimal cost.
- This worst case ratio between the cost incurred by the online strategy and the optimal cost is called the ‘COMPETITIVE RATIO’.
Is there a better strategy given the rules of the game?

A strategy is simply a value ‘k’: the number of times to rent before buying

Cost of strategy:

\[ tx \text{ for } t \leq k \]
\[ kx + y \text{ for } t > k \]

Clearly, there is no value of k that is guaranteed to achieve optimal cost in all cases
Any $k$ is non-optimal for the case $t=k+1$

Optimal cost = $tx = (k+1)x$

Online cost = $kx + y$

$kx+y = kx+cx \geq (k+2)x > (k+1)x = tx$

This is typical of online problems

Without future knowledge, there is no online algorithm that is always optimal
EXAMPLE: SKI RENTAL

- It is not hard to see that no strategy can have a competitive ratio that is less than 2
- The worst case ratio between the online cost and the optimal cost is
  \[ \frac{kx+y}{\min(tx,y)} \text{ OR } \max(kx+y/tx, \frac{kx+y}{y}) \]
- If \( k=0 \), then for \( t=1 \), first ratio is \( y/x \) which by assumption is at least 2
- If \( kx \leq y \), then the ratio is at least 2 when \( t=k \) (first ratio in the max)
- If \( kx > y \), then the ratio is at least 2 when \( t>k \) (second ratio in the max)
EXAMPLE: SKI RENTAL

- Renting costs $20 a day
- Buying costs $300
Omniscient strategy (if you know in advance you will ski \( x \) days):

- If \( x < 15 \), optimal policy is to rent.
- If \( x > 15 \), optimal policy is to buy the first day.
- If \( x = 15 \), both policies are the same.

An Online strategy is described by a threshold \( z \):

- Rent for up to \( z \) days, then buy, if still skiing.
EXAMPLE: SKI RENTAL

Offline Solution

- If Tamon knew today that he would be skiing for \( d \) days (Instance \( I_d \)), his problem is easy
- If \( 20d \leq 300 \) then rent

Else buy

- Offline optimum cost
  - \( \text{OPT} (I_d) = \min (20d, 300) \)

...BUT Tamon does not know \( d \)!!
EXAMPLE: SKI RENTAL

General Online Ski Rental Algorithm $A_x$

- Rent for up to $x$ days
- Then buy, if still skiing

How to evaluate the cost of an online algorithm?
General Online Ski Rental Algorithm $A_x$

(Rent for $x$ days, then buy)

- If Tamon ends up skiing $d$ days, his actual cost is

$$C(A_x, I_d) = \begin{cases} 
20d & \text{if } d < x \\
20x + 300 & \text{otherwise}
\end{cases}$$

Whereas he could have only paid

$$OPT(I_d) = \min(20d, 300)$$

... but we don’t know which case will apply!
GENERALIZATION OF SKI RENTAL PROBLEM

- Ski rental is relevant not only to the management of sports equipment
- Applicable to wide variety of resource allocation problems
- For example: power management in a laptop computer
- Laptop powers down the hard drive when it isn’t in use, because running a hard drive consumes battery power
- It takes significant amount of power and time, however, to restart the hard drive
- If the user of the laptop doesn’t use the hard drive for a while, how long the laptop should wait to powering it down?
- A typical online problem!!