CS60029
RANDOMIZED ALGORITHM DESIGN

YAO'S LEMMA

Bratin Mondal (21CS10016)

Voddula Karthik Reddy (21CS30058)

Department of Computer Science and Engineerinng

Indian Institute of Technology Kharagpur

MOTIVATION

1. How can we go about proving lower bounds for worst case costs incurred of randomized algorithms ?
2. How do online(Randomized?) and offline algorithms compare with each other?

FEW CONVENTIONS AND DEFINITIONS

We assume that we have a class of deterministic algorithms $\mathcal{A}$ and a class of instances $\mathcal{X}$. We assume both classes are finite. Algorithm $a \in \mathcal{A}$ on instance $x \in \mathcal{X}$ incurs cost $c(a, x) \in \mathbb{R}$. A randomized algorithm is can viewed as probability distribution over the set of deterministic algorithms $\mathcal{A}$. So, let A be a randomized algorithm (which is now a random variable), then A 's worst-case cost is $\max_{x \in \mathcal{X}} \mathbb{E}[c(A, x)]$.

Definition 1.0 (Competitive Ratio) An online Algorithm ALG is said to be is *α-competitive* if there is a constant c such that for all finite input sequences I we have,

$$ALG(I) \leq \alpha \cdot OPT(I) + c$$

if c=0, we call ALG is *strictly α-competitive*

YAO'S LEMMA

Theorm 1.0 (Yao's Principle). Let A be a random variable with values in $\mathcal{A}$ and let X be a random variable with values in $\mathcal{X}$. Then,

$$\max_{x \in \mathcal{X}} \mathbb{E}[c(A, x)] \geq \min_{a \in \mathcal{A}} \mathbb{E}[c(a, X)]$$

LHS is simply worst case performance of randomized algorithm A. RHS is average-case performance of the best deterministic algorithm in our class. The distribution over instances is arbitrary.

Proof. Let us first write the expectations as sums over all possible outcomes of X and A.

$$\mathbb{E}[c(A, x)] = \sum_{a \in \mathcal{A}} \mathbf{Pr}[A = a] c(a, x) \quad \text{and} \quad \mathbb{E}[c(a, X)] = \sum_{x \in \mathcal{X}} \mathbf{Pr}[X = x] c(a, x)$$

Now we use that the weighted average of a sequence is always upper-bounded by its maximum value. In our case, the weights are $\mathbf{Pr}[X = x]$. As $\sum_{x \in \mathcal{X}} \mathbf{Pr}[X = x] = 1$, we have

$$\max_{x \in \mathcal{X}} \mathbb{E}[c(A, x)] = \max_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} \mathbf{Pr}[A = a] c(a, x) \geq \sum_{x \in \mathcal{X}} \mathbf{Pr}[X = x] \sum_{a \in \mathcal{A}} \mathbf{Pr}[A = a] c(a, x).$$

We can now reorder the sums and get

$$\sum_{x \in \mathcal{X}} \mathbf{Pr}[X = x] \sum_{a \in \mathcal{A}} \mathbf{Pr}[A = a] c(a, x) = \sum_{a \in \mathcal{A}} \mathbf{Pr}[A = a] \sum_{x \in \mathcal{X}} \mathbf{Pr}[X = x] c(a, x)$$

Finally, we can use that $\sum_{a \in \mathcal{A}} \mathbf{Pr}[A = a] = 1$ the same way as above to obtain

$$\sum_{a \in \mathcal{A}} \mathbf{Pr}[A = a] \sum_{x \in \mathcal{X}} \mathbf{Pr}[X = x] c(a, x) \geq \min_{a \in \mathcal{A}} \sum_{x \in \mathcal{X}} \mathbf{Pr}[X = x] c(a, x) = \min_{a \in \mathcal{A}} \mathbb{E}[c(a, X)]$$

USE CASE'S

Yao's principle is a powerful tool which can be used to get worst case lower bounds for Randomsied algorithms, especially to those algorithms which give the correct result always but cost asscoicated with them might vary(Las Vegas Algorihtms).

- In particular worst case cost of any randomized(online) is given by LHS, while RHS is the best possible(lowest cost) deterministic algorithm(online),RHS can be manipulated by changing distribution of X, we wish to take X such that all deterministic algorithms perform poorly in average. A case of interest occurs when we are able to make expected the cost all deterministic algorithms the same. If we know optimal offline solution, competitive ratio can be calculated accordingly.
- Next, we demonstrate few examples using the above method.

Application to online set-cover

Theorem 1.1. No algorithm for Online Set Cover is strictly α-competitive for $\alpha < 1 + \frac{1}{2}\log_2 n$.

Proof. We consider n that are powers of two. Define the set system as follows. Take a complete binary tree of $2n-1$ vertices. This tree has height $h = \log_2 n + 1$ and n leaves.

Potential elements e to be covered correspond to nodes in the tree. Only some of these potential elements will be present in our universe U of elements which have to be covered. The set U corresponds to the elements on one root-leaf path.

The sets $S \in \mathcal{S}$ correspond to leaves of the tree, where S contains all ancestors in the tree that are in U. We set $c_S = 1$ for all $S \in \mathcal{S}$.

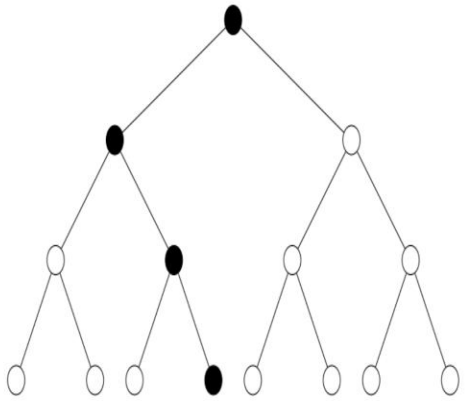


Figure 1: The binary tree used in the construction of the lower bound. The black nodes correspond to a potential sequence.

- Cost of optimal offline algorithm is 1 i.e. choose the req set.
- Let Distribution of X be uniform
- Without loss of generality at each step only req no of sets are chosen
- Equivalently at each step t with probability 1/2 the new element belongs to the set chosen and with probability ½ it does not belong to it

so $\mathbb{E}[C_t] = \frac{1}{2}$.

Overall, we get for any $a \in \mathcal{A}$

$$\mathbb{E}[c(a, X)] = \sum_{t=1}^{h} \mathbb{E}[C_t] = 1 + (h-1)\frac{1}{2} = 1 + \frac{1}{2}\log_2 n.$$

Ski Rental Problem

Statement Let x be the number of days that a person plans to ski. Renting skis costs \$1 per day, and buying skis once costs \$$B$. Each day, the person must decide whether to continue renting skis for one more day or buy a pair of skis. x is not known in advanced.

Objective The objective is to design an algorithm that minimizes the ratio between the cost incurred in the absence of prior knowledge of d and the cost that would be incurred optimally if x was known in advance.

For the offline optimum OPT,

$$C(OPT, x) = \begin{cases} x & \text{if } x < B \\ B & \text{otherwise} \end{cases}$$

Deterministic Algorithms

Deterministic algorithms differ only in how long they wait until buying the skies. If a particular deterministic algorithm $A(a)$ waits until a $a \geq 0$ days before buying on day a+1 (if this day exists). Then cost of the algorithm $A(a)$ is

$$C(A(a), x) = \begin{cases} x & \text{if } x \leq a \\ a + B & \text{otherwise} \end{cases}$$

The expected cost for this algorithm for a distribution of $x \in X$

$$\mathbb{E}[c(A(a), X)] = \sum_{x=1}^{a} x\mathbf{Pr}[X = x] + (a + B)\mathbf{Pr}[X > a] = \sum_{x=1}^{a} \mathbf{Pr}[X \geq x] + B\mathbf{Pr}[X > a]$$

Choosing the Distribution

$$\mathbb{E}[c(A(a), X)] = \mathbb{E}[c(A(a-1), X)]$$

$$\sum_{x=1}^{a-1} \mathbf{Pr}[X \geq x] + B\mathbf{Pr}[X > a-1] = \sum_{x=1}^{a-1} \mathbf{Pr}[X \geq x] + \mathbf{Pr}[X \geq a] + B\mathbf{Pr}[X > a]$$

$$B\mathbf{Pr}[X \geq a] = \mathbf{Pr}[X \geq a] + B\mathbf{Pr}[X \geq a+1]$$

$$\mathbf{Pr}[X \geq a+1] = \left(1 - \frac{1}{B}\right)\mathbf{Pr}[X \geq a]$$

$$\text{Set } \mathbf{Pr}[X \geq x] = \left(1 - \frac{1}{B}\right)^{x-1} \text{ for all } x \geq 1$$

Expected Costs

$$\mathbb{E}[c(OPT, x)] = \sum_{x=1}^{B-1} x\mathbf{Pr}[X = x] + B\mathbf{Pr}[X \geq B] = \sum_{x=1}^{B} \mathbf{Pr}[X \geq x]$$
$$= \sum_{x=1}^{B} \left(1 - \frac{1}{B}\right)^{x-1} = B\left(1 - \left(1 - \frac{1}{B}\right)^{B}\right)$$

$$\mathbb{E}[c(A(a), X)] = \sum_{x=1}^{a} \left(1 - \frac{1}{B}\right)^{x-1} + B\left(\frac{1}{B}\right)^{a} = B$$

Competitive Ratio of Randomized Algorithm

Lemma. *For a fixed B, no randomized algorithm for Ski Rental is $\alpha-$competitive for $\alpha < \left(1-\left(1-\frac{1}{B}\right)^B\right)^{-1}$*

Proof. Suppose a randomized algorithm A is $\alpha-$competitive for $\alpha < \left(1-\left(1-\frac{1}{B}\right)^B\right)^{-1}$.This implies

$$\mathbb{E}[c(A,X)] \leq \alpha \mathbb{E}[c(OPT,X)]$$

As we know $\mathbb{E}[c(A,X)] = \sum_{a\in\mathcal{A}} \mathbf{Pr}[A=a]\mathbb{E}[c(A,X)] = B$ and $\mathbb{E}[c(OPT,x)] = B\left(1-\left(1-\frac{1}{B}\right)^B\right)$ which is a contradiction □

Theorem 7 *There is a simple on-line randomized algorithm A for the spin-block problem which is strongly $e/(e-1)$-competitive against a weak adversary.*

$$\lim_{B\to\infty}\left(1-\left(1-\frac{1}{B}\right)^B\right)^{-1} = \frac{e}{e-1}$$

Karlin et al.

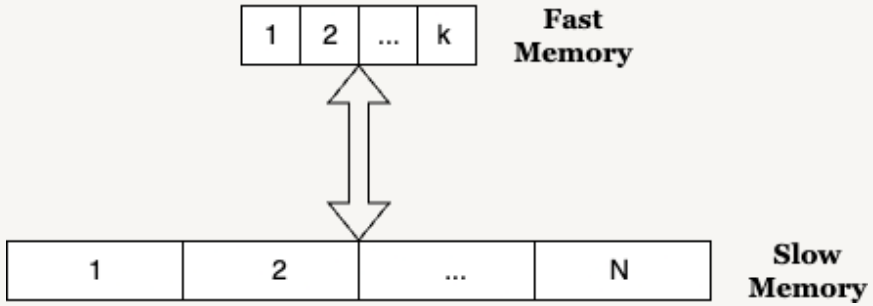
https://courses.csail.mit.edu/6.895/fall03/handouts/papers/karlin.pdf

Paging Problem

Statement Consider two-level computer memory system. $k < N$

At each time step t a request for some page P_{i_t} comes. A **hit** occurs if P_{i_t} is already in cache. Otherwise, a **miss** occurs. Then the system incurs one page fault and P_{i_t} must be fetched from slow memory to cache.

Objective Decide which k pages to retain in the cache, at each point of time, to minimize misses (maximize hits)



Offline Setting

LFD (LONGEST-FORWARD-DISTANCE) On miss, evict the page whose next request is farthest away

Claim: For any finite sequence σ chosen from a set of $k+1$ pages, $c(LFD, \sigma) \leq \frac{|\sigma|}{k}$

Phase 0 : empty sequence
Phase i : maximal sequence that immediately follows phase $i-1$ and contains at most k distinct page requests

Online Deterministic Algorithms

Lemma. *There is no deterministic algorithm for the paging problem that is α-competitive for $\alpha < k$*

Proof. Adversary always requests the page which is not currently present in cache. Implies miss on every request. □

Lower Bound for Randomized Algorithms

Coupon Collector Problem Number of random attempts needed to collect a complete set of n distinct items, when each item is selected with equal probability.
Let x_i denote the number of attempts needed to collect the i-th distinct element after collecting $(i-1)$-distinct elements and X denote the total number of attempts to collect all the elements.

$$\mathbb{E}[x_i] = \frac{n}{n-i+1}$$

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^{n} x_i\right] = \sum_{i=1}^{n} \mathbb{E}[x_i] = \sum_{i=1}^{n} \frac{n}{n-i+1} = n\sum_{j=1}^{n} \frac{1}{j} = nH(n)$$

Renewal Theory A renewal process represents process that counts number of times a certain event restarts itself within some time interval
$\{X_i : i \in \mathbb{N}\}$ be sequence of positive i.i.d Random Variables. Let $\mathbf{Pr}[X_i = 0] < 1$ and $S_n = \sum_{i=1}^{n} X_i$ for $n \in \mathbb{N}$, $S_0 = 0$. $N(t) = \sup\{n : S_n \leq t\}$
Formally Renewal Process is a stochastic process $\{N(t) : t \geq 0\}$

S_n : time of the n'th renewal
X_n : time between $(n-1)$ and n'th renewal $= S_n - S_{n-1}$
$N(t)$: Number of renewals in the interval $[0, t]$
$M(t)$: Renewal function $= \mathbb{E}[N(t)]$

Elementary Renewal Theorem: Given a sequence of i.i.d Random Variables $X_i, i \geq 1$ with $0 < \mathbb{E}[X_1] \leq \infty$, then

$$\lim_{t\to\infty} \frac{M(t)}{t} = \frac{1}{\mathbb{E}[X_1]}$$

Lemma. *Any randomized algorithm for the paging problem has competitive ratio* $\geq \ln k$*, where* k *is the size of the cache*

Universe size $|u| = k + 1$
$\mathcal{D}$: uniform distribution over the $k + 1$ files
Claim: For any deterministic algorithm A:

$$\mathbb{E}_{\mathcal{D}}[c(A, \sigma)] = \frac{|\sigma|}{k+1}$$

Upper Bound on $\mathbb{E}_{\mathcal{D}}[c(LFD, \sigma)]$:

Partition σ into phases P_i,where P_i consists of requests made at time $\theta_i, \theta + 1, .., \theta_{i+1} - 1$,
where $\theta_1 = 1$, and $\theta_{i+1} = \min \{r : \{\sigma_{t_i}, \sigma_{t_i+1}, .., \sigma_r\} = [k + 1]\}$
Thus P_i contains exactly k distinct file requests are made. $|P_i| = \theta_{i+1} - \theta_i$
In LFD, if one fault is incurred in a phase then no more faults can occur

Number of faults incurred by LFD until time $|\sigma| \leq$ Number of completed phases by time $|\sigma| + 1$

$$\mathbb{E}_{\mathcal{D}}[c(LFD, \sigma)] \leq 1 + \mathbb{E}[\max\{l : \theta_l \leq j\}]$$

Now D is uniform and i.i.d, $|P_i| = \theta_{i+1} - \theta_i$ is also i.i.d.
Apply elementary renewal theorem, with renewal intervals. $X_i = P_i = \theta_{i+1} - \theta_i$

$$\lim_{j\to\infty} = \frac{(1 + \mathbb{E}[\max\{l : \theta_l \leq j\}])}{j} = \frac{1}{\mathbb{E}[|P_1|]}$$

When $|\sigma| \to \infty$, the number of completed phases normalized with $|\sigma|$ converges to the reciprocal of the expected length of a phase.

$$\lim_{j\to\infty} \frac{c(LFD, \sigma)}{j} \leq \frac{1}{\mathbb{E}[|P_1|]}$$

From the definition of phase, $|P_1| + 1$ is identically distributed as the finish time T_c in the coupon collector problem.

$$\mathbb{E}(|P_1|) = (k+1)H_{k+1} - 1 = (k+1)H_k$$

So, we can bound the expected cost of LFD as

$$\mathbb{E}[c(LFD, \sigma)] \leq \frac{j}{\mathbb{E}[P_1]} \leq \frac{j}{(k+1)H_k}$$

Also

$$\min_{\mathcal{A}} \mathbb{E}_D(c(A, \sigma) = \frac{j}{k+1}$$

So competitive ratio of any online randomized algorithm can be bounded as

$$\mu_R \geq \frac{\frac{j}{k+1}}{\frac{j}{(k+1)H_k}} \geq H_k \geq \ln k$$

MARKING $2 \ln k$-competitive randomized algorithm

Algorithm 1 Marking Algorithm

```
while True do
    for i ← 1 to k do
        Set marker bit m_i = 0
    end for
    while Not all m_i = 1 for i = 1, ..., k do
        Accept a page request
        if Requested page is in cache location i then
            Set m_i = 1
        else
            Choose an unmarked cache location j at random
            Evict cache location j and bring the new page into j
            Set m_j = 1
        end if
    end while
end while
```

References:

1. https://tcs.cs.uni-bonn.de/lib/exe/fetch.php/teaching/ws1819/vl-aau/lecturenotes05.pdf
2. https://www.csa.iisc.ac.in/~barman/AlgUncertain/static/lectureNotes/Online1IntroPaging.pdf
3. https://www.csa.iisc.ac.in/~barman/AlgUncertain/static/lectureNotes/L5l6SkiRental.pdf
4. https://cse.iitkgp.ac.in/~palash/Courses/2022AlgorithmicGameTheory/Files/2022AlgorithmicGameTheoryLectureNotes.pdf
5. https://www.cse.iitd.ac.in/~naveen/courses/CSL758/scribes/lec23.pdf
6. https://www.mpi-inf.mpg.de/fileadmin/inf/d1/teaching/summer16/random/yaosprinciple.pdf
7. https://youtu.be/0DNoIb24xi4?si=GZNCjXHFEE72HFi5
8. https://www.youtube.com/live/oK3QUvUUHHI?si=KpRYu4THNBZaEsGG

Thank You