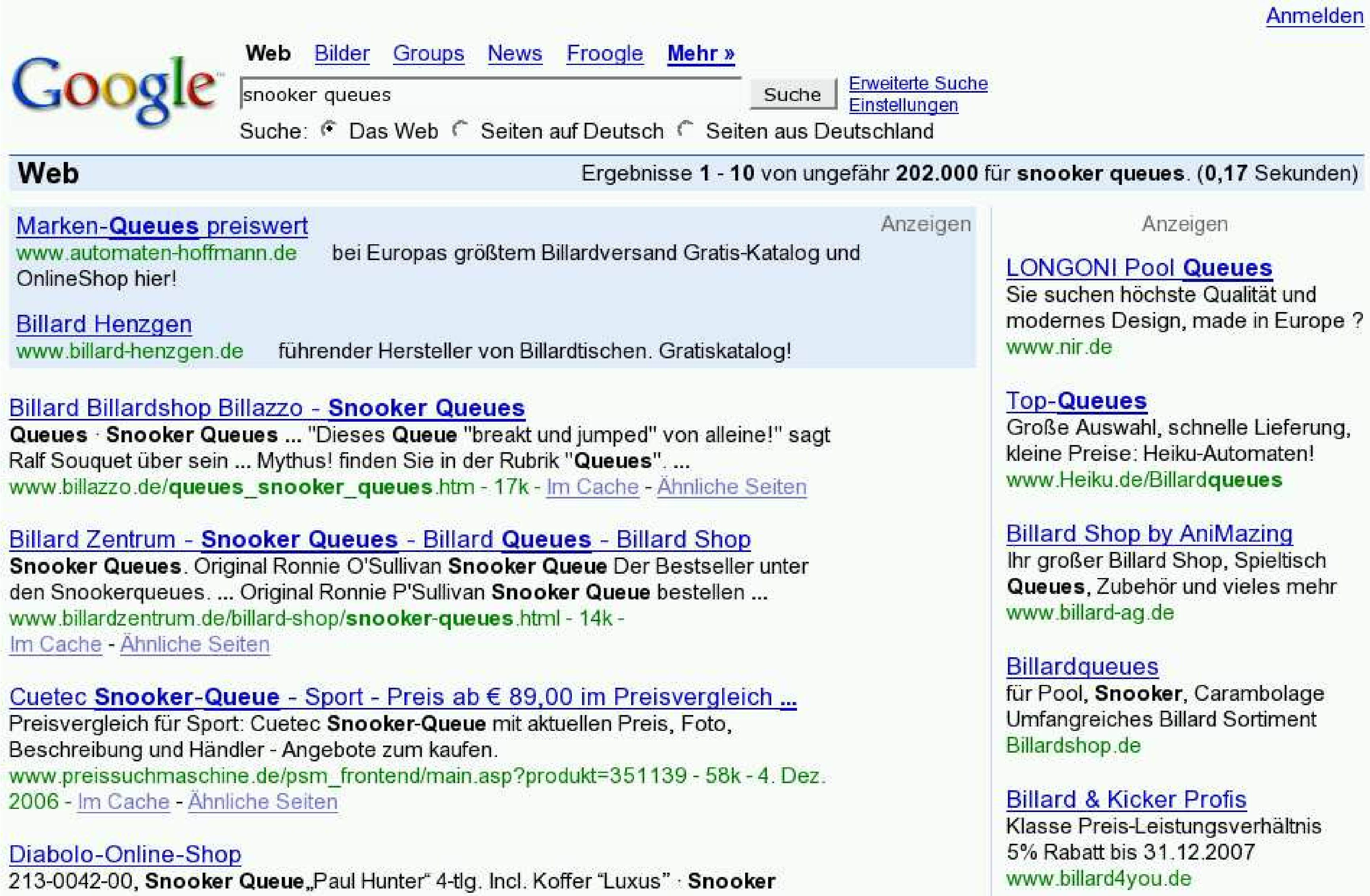
Online algorithms

- Information is revealed to the algorithm in parts
- Algorithm needs to process each part before receiving the next
- There is **no information** about the future (in particular, no probabilistic assumptions!)
- How well can an algorithm do compared to an algorithm that knows everything?
- Lack of knowledge vs. lack of processing power

Example: *Google ad auctions*

- Google is popular because of its PageRank algorithm
- However, it can only earn money through ads
- Ads are linked to specific keywords
- Advertisers select keywords, maximum price, and daily budget
- They pay only if
  - their ad is shown with the keyword
  - someone clicks on the ad
  - the daily budget has not yet been reached


Anmelden

Google

Web Bilder Groups News Froogle **Mehr »**

snooker queues | Suche | Erweiterte Suche | Einstellungen

Suche: Das Web Seiten auf Deutsch Seiten aus Deutschland

Web Ergebnisse **1** - **10** von ungefähr **202.000** für **snooker queues**. (**0,17** Sekunden)

Anzeigen

Marken-**Queues** preiswert
www.automaten-hoffmann.de bei Europas größtem Billardversand Gratis-Katalog und OnlineShop hier!

Billard Henzgen
www.billard-henzgen.de führender Hersteller von Billardtischen. Gratiskatalog!

Billard Billardshop Billazzo - **Snooker Queues**
Queues · **Snooker Queues** ... "Dieses **Queue** "breakt und jumped" von alleine!" sagt Ralf Souquet über sein ... Mythus! finden Sie in der Rubrik "**Queues**". ...
www.billazzo.de/**queues_snooker_queues**.htm - 17k - Im Cache - Ähnliche Seiten

Billard Zentrum - **Snooker Queues** - Billard **Queues** - Billard Shop
Snooker Queues. Original Ronnie O'Sullivan **Snooker Queue** Der Bestseller unter den Snookerqueues. ... Original Ronnie P'Sullivan **Snooker Queue** bestellen ...
www.billardzentrum.de/billard-shop/**snooker-queues**.html - 14k -
Im Cache - Ähnliche Seiten

Cuetec **Snooker-Queue** - Sport - Preis ab € 89,00 im Preisvergleich ...
Preisvergleich für Sport: Cuetec **Snooker-Queue** mit aktuellen Preis, Foto, Beschreibung und Händler - Angebote zum kaufen.
www.preissuchmaschine.de/psm_frontend/main.asp?produkt=351139 - 58k - 4. Dez. 2006 - Im Cache - Ähnliche Seiten

Diabolo-Online-Shop
213-0042-00, **Snooker Queue**„Paul Hunter“ 4-tlg. Incl. Koffer “Luxus” · **Snooker**

Anzeigen

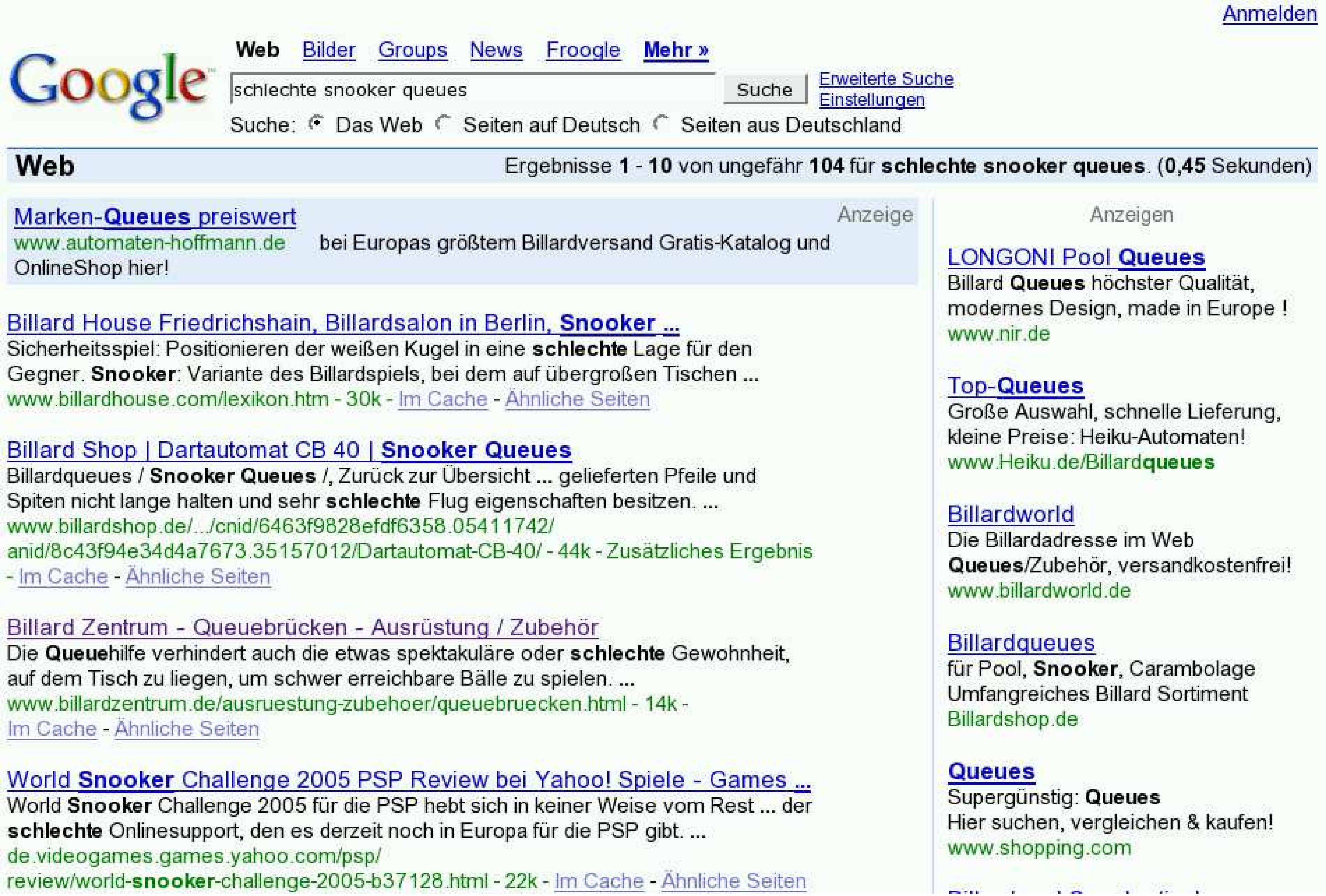
LONGONI Pool **Queues**
Sie suchen höchste Qualität und modernes Design, made in Europe ?
www.nir.de

Top-**Queues**
Große Auswahl, schnelle Lieferung, kleine Preise: Heiku-Automaten!
www.Heiku.de/Billard**queues**

Billard Shop by AniMazing
Ihr großer Billard Shop, Spieltisch **Queues**, Zubehör und vieles mehr
www.billard-ag.de

Billardqueues
für Pool, **Snooker**, Carambolage Umfangreiches Billard Sortiment
Billardshop.de

Billard & Kicker Profis
Klasse Preis-Leistungsverhältnis 5% Rabatt bis 31.12.2007
www.billard4you.de


Anmelden

Google

Web Bilder Groups News Froogle **Mehr »**

schlechte snooker queues | Suche | Erweiterte Suche | Einstellungen

Suche: Das Web Seiten auf Deutsch Seiten aus Deutschland

Web Ergebnisse **1** - **10** von ungefähr **104** für **schlechte snooker queues**. (**0,45** Sekunden)

Marken-**Queues** preiswert Anzeige
www.automaten-hoffmann.de bei Europas größtem Billardversand Gratis-Katalog und OnlineShop hier!

Billard House Friedrichshain, Billardsalon in Berlin, **Snooker** ...
Sicherheitsspiel: Positionieren der weißen Kugel in eine **schlechte** Lage für den Gegner. **Snooker**: Variante des Billardspiels, bei dem auf übergroßen Tischen ...
www.billardhouse.com/lexikon.htm - 30k - Im Cache - Ähnliche Seiten

Billard Shop | Dartautomat CB 40 | **Snooker Queues**
Billardqueues / **Snooker Queues** /, Zurück zur Übersicht ... gelieferten Pfeile und Spiten nicht lange halten und sehr **schlechte** Flug eigenschaften besitzen. ...
www.billardshop.de/.../cnid/6463f9828efdf6358.05411742/anid/8c43f94e34d4a7673.35157012/Dartautomat-CB-40/ - 44k - Zusätzliches Ergebnis - Im Cache - Ähnliche Seiten

Billard Zentrum - Queuebrücken - Ausrüstung / Zubehör
Die **Queue**hilfe verhindert auch die etwas spektakuläre oder **schlechte** Gewohnheit, auf dem Tisch zu liegen, um schwer erreichbare Bälle zu spielen. ...
www.billardzentrum.de/ausruestung-zubehoer/queuebruecken.html - 14k - Im Cache - Ähnliche Seiten

World **Snooker** Challenge 2005 PSP Review bei Yahoo! Spiele - Games ...
World **Snooker** Challenge 2005 für die PSP hebt sich in keiner Weise vom Rest ... der **schlechte** Onlinesupport, den es derzeit noch in Europa für die PSP gibt. ...
de.videogames.games.yahoo.com/psp/review/world-**snooker**-challenge-2005-b37128.html - 22k - Im Cache - Ähnliche Seiten

Anzeigen

LONGONI Pool **Queues**
Billard **Queues** höchster Qualität, modernes Design, made in Europe !
www.nir.de

Top-**Queues**
Große Auswahl, schnelle Lieferung, kleine Preise: Heiku-Automaten!
www.Heiku.de/Billard**queues**

Billardworld
Die Billardadresse im Web
Queues/Zubehör, versandkostenfrei!
www.billardworld.de

Billardqueues
für Pool, **Snooker**, Carambolage
Umfangreiches Billard Sortiment
Billardshop.de

Queues
Supergünstig: **Queues**
Hier suchen, vergleichen & kaufen!
www.shopping.com

Second price auction

- Advertisers only need to beat competitors: they pay the second price
- When no competitors are left, price drops to 0
- Google wants to maximize revenue : keep advertisers solvent!
- **Question** for Google:

which ads should be shown

each time a certain keyword is looked for?

Optimizing the revenue

Which ads should be shown? Google needs to take into account

- Clickthrough probability for a given ad
- Remaining budget of each advertiser
- Future queries on the same day...

A typical online problem!

Online matching

- Advertisers, bids and budgets are known in advance
- Queries occur over time (during the day)
- Each query must be matched to some ads without knowledge of future queries

Result: it is possible to give an algorithm which gets

$$1 - \frac{1}{e} \approx 0.62312$$

of the optimal offline revenue (knowing all queries in advance)

Competitive analysis

- Idea: compare online algorithm ALG to offline algorithm OPT
- Worst-case performance measure
- Definition:

$$C_{ALG} = \sup_{\sigma} \frac{\text{ALG}(\sigma)}{\text{OPT}(\sigma)}$$

(we look for the input that results in worst relative performance)

- Goal:

find ALG with minimal C_{ALG}

A typical online problem: ski rental

- Renting skis costs 50 euros, buying them costs 300 euros
- You do not know in advance how often you will go skiing
- Should you rent skis or buy them?

A typical online problem: ski rental

- Renting skis costs 50 euros, buying them costs 300 euros
- You do not know in advance how often you will go skiing
- Should you rent skis or buy them?
- Suggested algorithm: buy skis on the sixth trip
- Two questions:
  - How good is this algorithm?
  - Can you do better?

Upper bound for ski rental

- You plan to buy skis on the sixth trip
- If you make five trips or less, you pay optimal cost (50 euros per trip)
- If you make at least six trips, you pay 550 euros
- In this case OPT pays at least 300 euros
- Conclusion: algorithm is $\frac{11}{6}$-competitive: it never pays more than $\frac{11}{6}$ times the optimal cost

Lower bound for ski rental

- Suppose you buy skis earlier, say on trip $x < 6$.

  You pay $300 + 50(x-1)$, OPT pays only $50x$

$$\frac{250+50x}{50x} = \frac{5}{x} + 1 \geq 2.$$

- Suppose you buy skis later, on trip $y > 6$.

  You pay $300 + 50(y-1)$, OPT pays only 300

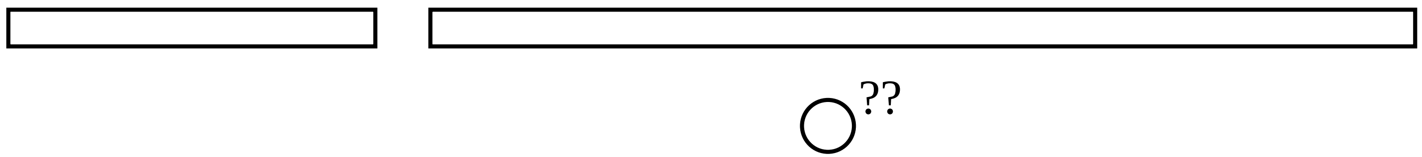
$$\frac{250+50y}{300} = \frac{5+y}{6} \geq 2.$$

- Idea: do not pay the large cost (buy skis) until you have paid the same amount in small costs (rent)

The cow path problem

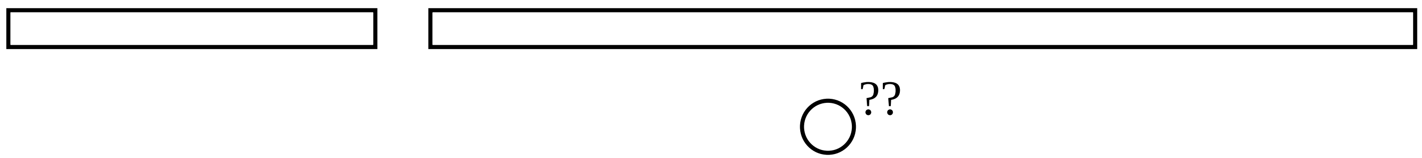
The cow path problem

- A cow wants to get past a fence
- There is a hole in this fence, but the cow does not know where it is
- How can it find the hole quickly?

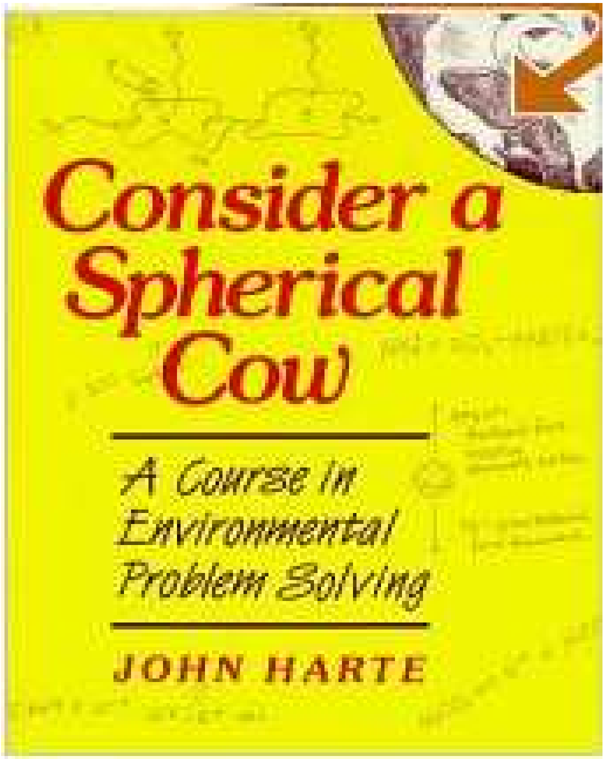


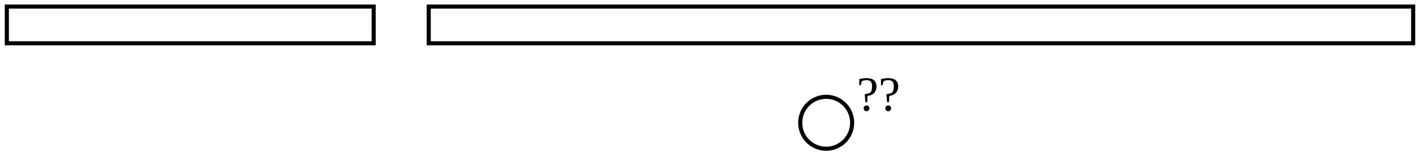
The cow path problem

- A cow wants to get past a fence
- There is a hole in this fence, but the cow does not know where it is
- How can it find the hole quickly?
- Idea: use a doubling strategy



The cow path problem





Algorithm

- Let $d = 1$, side = Right
- Repeat until hole is found:
  - Walk distance d to current side
  - If hole not found: return to starting point, double the value of d and flip side

Analysis

- What is the worst that could happen?

Analysis

- What is the worst that could happen?
- Answer: hole is slightly beyond a point where you turn around
- Denote its distance from the starting point by $2^i + \varepsilon$, where ε is very small.
- $2^i + \varepsilon$ is the distance that OPT walks.

Analysis

Total distance walked by the cow is

$$2(1 + 2 + \cdots + 2^{i+1}) + 2^i + \varepsilon$$

Analysis

Total distance walked by the cow is

$$
\begin{aligned}
& 2(1+2+\cdots+2^{i+1})+2^i+\varepsilon \\
= \quad & 2(2^{i+2}-1)+2^i+\varepsilon
\end{aligned}
$$

Analysis

Total distance walked by the cow is

$$
\begin{aligned}
& 2(1+2+\cdots+2^{i+1}) + 2^i + \varepsilon \\
= \quad & 2(2^{i+2}-1) + 2^i + \varepsilon \\
< \quad & 8 \cdot 2^i + 2^i
\end{aligned}
$$

Analysis

Total distance walked by the cow is

$$
\begin{aligned}
& 2(1+2+\cdots+2^{i+1})+2^i+\varepsilon \\
= \quad & 2(2^{i+2}-1)+2^i+\varepsilon \\
< \quad & 8\cdot 2^i+2^i+\varepsilon \\
< \quad & 9\cdot \text{OPT} \qquad\qquad \text{OPT}=2^i+\varepsilon
\end{aligned}
$$

The adversary

- An online problem can be seen as a game between two players
- One player is the online algorithm
- The other is the adversary
- The adversary tries to make things difficult for the online algorithm
- The algorithm minimizes the competitive ratio, the adversary maximizes it

List update

- Suppose you have a collection of ℓ unsorted files
- When you need a certain file, you can only find it by going through the list from the start
- After finding a file, you may move it closer to the start of the list
- Goal: minimize overall search time
- Linked-list data structure
- NP-hard (Ambühl, 2000)

Upper bound

- Every request costs at most ℓ, the length of the list
- OTP pays at least 1 for each request
- Thus, no algorithm has a competitive ratio above ℓ

Three algorithms

- Move-To-Front (MTF): move requested item to start of list
- Transpose (TRA): exchange requested item with item before it
- Frequency Count (FC): sort items by amount of requests for them

Three algorithms

- Move-To-Front (MTF): move requested item to start of list

  Seems like an overreaction

- Transpose (TRA): exchange requested item with item before it

  More conservative

- Frequency Count (FC): sort items by amount of requests for them

  Requires bookkeeping

TRA is bad

- Consider list

x_1	x_2	$\ldots$	$x_{\ell-1}$	x_ℓ

- Request sequence: $x_\ell, x_{\ell-1}, x_\ell, x_{\ell-1}, \ldots$
- TRA pays ℓ for every request

Before request $1, 3, 5, \ldots$

x_1	x_2	$\ldots$	$x_{\ell-1}$	x_ℓ

Before request $2, 4, 6, \ldots$

x_1	x_2	$\ldots$	x_ℓ	$x_{\ell-1}$

TRA is bad

- Consider list

x_1	x_2	$\ldots$	$x_{\ell-1}$	x_ℓ

- Request sequence: $x_\ell, x_{\ell-1}, x_\ell, x_{\ell-1}, \ldots$
- TRA pays ℓ for every request
- OPT moves both items to start of list, and then pays at most 2 per request
- Long sequence of requests: TRA is not better than $\ell/2$-competitive
- *Locality of reference* makes request sequences similar to these likely

FC is bad

- Let $k > \ell$, initial list is $x_1, \ldots, x_\ell$
- Request sequence: k times x_1, $k-1$ times x_2,...
- In general, item x_i is requested $k+1-i$ times

Example:

List					Requests
x_1	x_2	x_3	x_4	x_5	$x_1, x_1, x_1, x_1, x_1,$
					$x_2, x_2, x_2, x_2,$
					$x_3, x_3, x_3,$
					$x_4, x_4,$
					x_5

FC is bad

- Let $k > \ell$, initial list is $x_1, \ldots, x_\ell$
- Request sequence: k times x_1, $k-1$ times x_2,...
- In general, item x_i is requested $k+1-i$ times
- FC never moves any item
- Total cost is

$$\sum_{i=1}^{\ell} i \cdot (k+1-i) = \frac{k\ell(\ell+1)}{2} + \frac{\ell(1-\ell^2)}{3}$$

FC is bad

- Total cost of FC is $\frac{k\ell(\ell+1)}{2} + \frac{\ell(1-\ell^2)}{3}$
- Optimal: move each page to front on first request
- Cost is $\sum_{i=1}^{k}\{i + (k-i)\} = k\ell$

List					Requests	Optimal cost
x_1	x_2	x_3	x_4	x_5	$x_1, x_1, x_1, x_1, x_1,$	5
x_2	x_1	x_3	x_4	x_5	$x_2, x_2, x_2, x_2,$	$2+3=5$
x_3	x_2	x_1	x_4	x_5	$x_3, x_3, x_3,$	$3+2=5$
x_4	x_3	x_2	x_1	x_5	$x_4, x_4,$	$4+1=5$
x_5	x_4	x_3	x_2	x_1	x_5	5

Free and paid transpositions

- A transposition is the switching of two consecutive items
- When a file is found, we can move it closer to the start of the list for free (free transpositions)
- Generally, we might also use paid transpositions: move items although they are not requested

Offline: paid transpositions are necessary!

Paid transpositions are required offline

Initial list

x_1	x_2	x_3

Request sequence x_3, x_2, x_3, x_2

Only free transpositions

Current list			Request	Cost
x_1	x_2	x_3	x_3	3
x_3	x_1	x_2	x_2	3
x_2	x_3	x_1	x_3	2
x_2	x_3	x_1	x_2	1

With paid transpositions

Current list			Request	Cost
x_1	x_2	x_3	–	1
x_2	x_1	x_3	x_3	3
x_2	x_3	x_1	x_2	1
x_2	x_3	x_1	x_3	2
x_2	x_3	x_1	x_2	1

Move-To-Front

For an algorithm ALG, we define

- $\text{ALG}_P(\sigma)$ = number of paid transpositions (cost 1)
- $\text{ALG}_F(\sigma)$ = number of free transpositions (cost 0)
- $\text{ALG}_C(\sigma)$ = total cost other than paid transpositions

We show

$$\text{MTF}(\sigma) \leq 2 \cdot \text{OPT}_C(\sigma) + \text{OPT}_P(\sigma) - \text{OPT}_F(\sigma) - n$$

Potential functions

- It is often hard to analyze the performance of an online algorithm on a long input sequence at once
- Analysis per request also usually does not work: OPT may have very low cost for some requests
- Solution: potential functions
- Idea: keep track of configurations of OPT and ALG
- If an action of ALG makes its configuration more similar to OPT, it is allowed to cost more
- Very powerful technique

Potential function

An *inversion* in the list of MTF with respect to the list of OPT is an ordered pair (x, y) for which

- x precedes y on list of MTF
- y precedes x on list of OPT

Let t_i = cost of MTF for request i.

Let Φ_i = number of inversions after request i.

Φ_i is a potential function.

We define amortized costs

$$a_i = t_i + \Phi_i - \Phi_{i-1}$$

Potential function

- Φ_i is always nonnegative
- $\Phi_0 = 0$ (lists are the same at the start)

$$
\begin{aligned}
\text{MTF}(\sigma) &= \sum_{i=1}^{n} t_i = \sum_{i=1}^{n} (a_i - \Phi_i + \Phi_{i+1}) \\
&= \Phi_0 - \Phi_n + \sum_{i=1}^{n} a_i \\
&= \sum_{i=1}^{n} a_i - \Phi_n
\end{aligned}
$$

Potential function

- We can bound MTF(σ) by bounding the amortized costs
- For request i and for OPT, let s_i be the search cost, and P_i (F_i) the number of paid (free) transpositions
- We show

$$a_i \le (2s_i - 1) + P_i - F_i$$

- Thus we relate the amortized costs to the optimal decisions
- Summing this for all i gives the theorem

From claim to theorem

If

$$a_i \le (2s_i - 1) + P_i - F_i$$

then

$$\sum_{i=1}^{n} a_i \le 2\sum_{i=1}^{n} s_i - n + \sum_{i=1}^{n} P_i - \sum_{i=1}^{n} F_i$$

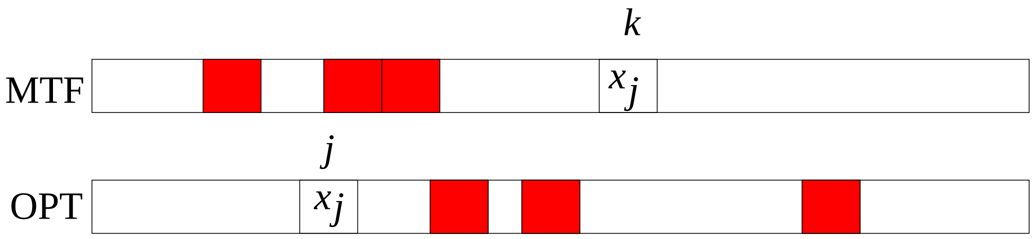
Therefore

$$\begin{aligned} \mathsf{MTF}(\sigma) &= \sum_{i=1}^{n} a_i - \Phi_n \\ &\le 2 \cdot \mathsf{OPT}_C(\sigma) + \mathsf{OPT}_P(\sigma) - \mathsf{OPT}_F(\sigma) - n \end{aligned}$$

which is what we wanted to show

Comparing the lists

- Consider the number of inversions involving the current request x_j
- x_j is at position j in list of OPT, position k in list of MTF
- Suppose there are v items that are before x_j in list of MTF and behind x_j in list of OPT
- Then OPT has at least $k-1-v$ items before x_j
- Thus, $k-1-v \leq j-1$



Comparing the lists (2)

- MTF moves x_j to front of list : $k-1-v$ new inversions created, v inversions removed
- Contribution to amortized cost:

$$k+(k-1-v)-v=2(k-v)-1\leq 2j-1=2s_i-1$$

- A paid exchange adds at most 1 to the potential function
- A free exchange contributes -1
- This proves the claim $a_i\leq(2s_i-1)+P_i-F_i$

Randomized algorithms

- A randomized algorithm is allowed to use random bits in its decision-making
- We compare its expected cost to the optimal cost
- The adversary knows the probability distribution(s) and chooses the input sequence in advance (oblivious adversary)
- ALG is c-competitive against an oblivious adversary if

$$\mathbb{E}(\text{ALG}(\sigma)) \leq c \cdot \text{OPT}(\sigma) + \alpha,$$

where α is a constant that does not depend on the input σ

Comparison to approximation algorithms

- Here, we are not interested in running times
- Purpose of randomization is only to decrease competitive ratio
- Compare weighted vertex cover:
  - deterministic 2-approximation solved linear program (time $O(n^{3.5}L)$)
  - randomized 2-approximation only flipped at most n coins (time $O(n)$)
- Disadvantage: random bits are not so easy to find

The BIT algorithm

- For each item x on the list, BIT uses one bit $b(x)$
- At the start, each bit is set to 0 or 1, independently and uniformly
- Whenever an element x is requested:
  - flip bit $b(x)$
  - If $b(x) = 1$, move x to front, else do nothing

Potential function

- Let $w(x, y)$ be the weight of inversion (x, y) = the number of times y is accessed before y passes x in the list of BIT
- We have $w(x, y) = b(y) + 1$ ($= 1$ or 2)
- Define potential Φ as
$$\boxed{\Phi = \sum_{\text{inversions (x,y)}} w(x, y).}$$
- We have $\Phi_0 = 0$ and $\Phi_n \geq 0$
- For amortized costs $a_i = \text{BIT}_i = \Phi_i - \Phi_{i-1}$ we have

$$BIT(\sigma) = \sum_{i=1}^{n} \text{BIT}_i = \Phi_0 - \Phi_n + \sum_{i=1}^{n} a_i$$

Events

There are two types of events in the sequence:

- A paid exchange by OPT
- All other operations from BIT and OPT to serve a request

For both types, we can show that the amortized cost of BIT is at most $7/4$ times the optimal cost

Here we only discuss the first type of event

Paid exchange of OPT

- Suppose event i in the sequence is a paid exchange of OPT
- Then OPT$_i = 1$ and BIT$_i = 0$
- The exchange might create a new inversion of weight 1 or 2
- If it creates no new inversion, $\Phi_i = \Phi_{i-1}$ and we are done
- Else, recall that $w(x, y) = b(y) + 1$
- $b(y)$ is 0 or 1, both with probability $1/2$, at the start of the algorithm and therefore throughout
- Thus $\mathbb{E}(a_i) = \frac{1}{2}(1 + 2) \leq \frac{3}{2} \cdot$ OPT$_i$