Kangaroos, Card Tricks and Discrete Logarithms

Steven Galbraith

Mathematics Department, University of Auckland

Outline

- Explaining the card trick.
- A computational problem about searching a list.
- The Pollard kangaroo algorithm.
- Open questions.

Thanks: LMS and NZMS.

Alexander Aitken

- Born in Dunedin, NZ.
- Studied Otago Boys' High School and Otago University.
- Served in WWI at Gallipoli and the Somme.
- PhD Edinburgh 1926.

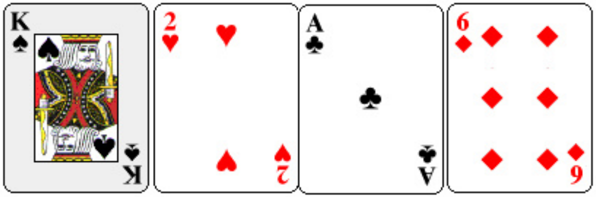
Alexander Aitken

"Professor Aitken's first year mathematics lectures were rather unusual. The fifty minutes were composed of forty minutes of clear mathematics, five minutes of jokes and stories and five minutes of tricks."

The Kruskal Trick

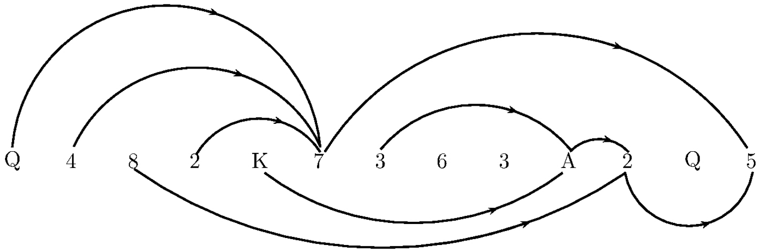
- Random walk along deck of cards from left to right.
- Number on the current card tells how many cards to the right to move.
- King, Queen, Jack count as 5.
- Ace means one step.

Crucial property



- Once two walks land on the same card, their paths are identical.

Crucial property



The Kruskal Trick

- When dealing the cards I run this walk starting from the first card.
- The last card in the deck visited by this walk is “distinguished”.
- The contestant starts a new walk at a “random” card on the left hand side.
- I win if the contestant’s walk visits any of the cards I visited in my walk.
- The contestant wins if their walk avoids all my cards.

The Kruskal Trick

- Average step length is about 5.38.
- Average length of walk is about $52/5.38 \approx 9.66$ steps.
- Probability my contestant wins is roughly

$$\left(1 - \tfrac{1}{5.38}\right)^{9.66} \approx 0.137 < \tfrac{1}{7}.$$

- I win with probability greater than $6/7$.

Finding an integer in a list

- Fix $n \in \mathbb{N}$ and let $A = \{1, 2, \ldots, n\}$.
- Let $F : A \to \mathbb{Z}$ be an injective function.
- **Computational problem:** Given an integer b, determine whether $b \in F(A)$ and, if so, compute $a \in A$ such that $F(a) = b$.

Finding an integer in a list

- Fix $n \in \mathbb{N}$ and let $A = \{1, 2, \ldots, n\}$.
- Let $F : A \to \mathbb{Z}$ be an injective function.
- **Computational problem:** Given an integer b, determine whether $b \in F(A)$ and, if so, compute $a \in A$ such that $F(a) = b$.

- **This is not a sorted list.**

Finding an integer in a list

- Fix $n \in \mathbb{N}$ and let $A = \{1, 2, \ldots, n\}$.
- Let $F : A \to \mathbb{Z}$ be an injective function.
- **Computational problem:** Given an integer b, determine whether $b \in F(A)$ and, if so, compute $a \in A$ such that $F(a) = b$.

- **This is not a sorted list.**

- **Theorem:** If F is arbitrary and one only has *oracle access* to F, then it is necessary to perform $\Omega(n)$ evaluations of the function to solve this problem on average.

More Structure

- Recall $A = \{1, 2, \ldots, n\}$.
- Suppose the injective function $F : A \to \mathbb{Z}$ satisfies $F(x + y) = F(x) + F(y)$ for all $x, y \in A$ such that $x + y \leq n$.
- Can one do better than the previous theorem?

More Structure

- Recall $A = \{1, 2, \dots, n\}$.
- Suppose the injective function $F : A \to \mathbb{Z}$ satisfies $F(x + y) = F(x) + F(y)$ for all $x, y \in A$ such that $x + y \leq n$.
- Can one do better than the previous theorem?

- Suppose $b = F(a)$ for some $a \in A$.
- Let $M = \lceil \sqrt{n} \rceil$.
- Then $a = a_0 + a_1 M$ for some integers $0 \leq a_0, a_1 < M$.
- Since $b = F(a) = F(a_0) + F(Ma_1)$ we have

$$F(a_0) = b - F(Ma_1).$$

More Structure

- Given b we wish to find a_0, a_1 such that
$$F(a_0) = b - F(Ma_1).$$
- So compute and store $L = \{(F(a_0), a_0) : 0 \leq a_0 < M\}$
- Then compute $b - F(Ma_1)$ for $a_1 = 0, 1, 2, \cdots$ checking each time if the value lies in L.
- If no match then there is no solution. If there is a match then the solution is $a_0 + a_1 M$.

The baby-step-giant-step algorithm

- Attributed to Shanks (1973), though Nechaev states it was known to Gel'fond in 1962.
- It requires $O(\sqrt{n})$ evaluations of the function F and requires storing $O(\sqrt{n})$ integers.
- Can one do better?

The baby-step-giant-step algorithm

- Attributed to Shanks (1973), though Nechaev states it was known to Gel'fond in 1962.
- It requires $O(\sqrt{n})$ evaluations of the function F and requires storing $O(\sqrt{n})$ integers.
- Can one do better?

- **Theorem:** (Shoup, 1997) If F is arbitrary (satisfying $F(x+y) = F(x) + F(y)$) and one only has *oracle access* to F, then it is necessary to perform $\Omega(\sqrt{n})$ evaluations of F to solve the problem.

Low storage algorithm

- Baby-step-giant-step requires storing $O(\sqrt{n})$ integers.
- Can one get the same running time but with lower storage?
- Before explaining such an algorithm, I should reveal that the function F I am most interested in is

$$F(i) = ig = g + g + \cdots + g$$

  where $g \in G$ is an element of a finite abelian group of order at least n.
- Hence, F is a non-injective function $F : \mathbb{Z} \to G$.
  But F is injective on $\{1, 2, \ldots, n\}$.
- Note that $F(x + y) = F(x) + F(y)$ for all $x, y \in \mathbb{Z}$.

Discrete Logarithm Problem

- Let $g \in \mathbb{Z}$ and p a prime, the function $F(a) = g^a \mod p$ is the canonical such function.
- The discrete logarithm problem is: Given $F(a)$ to determine a.
- This computational problem is fundamental to public key cryptography.
- In multiplicative groups of finite fields there exist algorithms for this problem that are much more efficient than the ones in this talk.
- For the group of points on an arbitrary elliptic curve over a finite field, no better algorithm is known.
- For elliptic curves it is traditional to write the group operation additively as $F(a) = aP$.

Low storage algorithm

- Given $F : A \to G$ such that $F(x + y) = F(x) + F(y)$, and given $b \in G$. Find $a \in A$ (if it exists) such that $F(a) = b$.
- Fix a set $\{s_1, \ldots, s_k\} \subset A$.
- Choose a function $H : G \to \{1, 2, \ldots, k\}$.
  Essentially this is just a "random" partition of G.
- Set $u_0 = F(\lfloor n/2 \rfloor) \in G$. Perform the following "random walk" for $i = 0, 1, 2, \cdots$

$$u_{i+1} = u_i + F(s_{H(u_i)}).$$

- This is a "random walk" along G, whose steps are determined by the values u_i.
- Note that $u_i = F(a_i)$ where $a_0 = \lfloor n/2 \rfloor$
  and $a_{i+1} = a_i + s_{H(u_i)}$.

This is the Kruskal walk

- We have defined a walk $u_{i+1} = F(a_i + s_{H(u_i)})$.
- Let C be the set of 52 cards and $F : \{1, 2, \ldots, 52\} \to C$ the function that puts them in a row.
  (We have now induced a binary operation $+$ on C by $F(x) + F(y) := F(x + y)$.)
- $u_i = F(a_i)$ means that the a_i-th card is the card u_i.
- The value $s_{H(u_i)}$ is the numerical value of the card (or 5 if the card is K, Q or J).
- The next step of the walk depends only on the card u_i, not its position.

Low storage algorithm

- Run the “random walk”, starting from $u_0 = F(a_0)$ where $a_0 = \lfloor n/2 \rfloor \in A$.
- From time to time, store a “distinguished point” u_i.
- In parallel, run another “random walk” starting at $v_0 = b$ and with the same rule

  $$v_{i+1} = v_i + F(s_{H(v_i)}).$$

  We also keep track of $b_i \in \mathbb{Z}$ such that $v_i = b + F(b_i)$.
- If $v_j = u_i$ for some integers i and j then the walks follow the same path.
- The “collision” is detected when the same “distinguished point” is visited twice.

Low storage algorithm

- At that distinguished point we have

$$F(a_i) = u_i = v_j = b + F(b_j)$$

  and so $b = F(a_i) - F(b_j) = F(a_i - b_j)$ and the problem is solved.
- This is the “kangaroo algorithm” invented by John Pollard in 1978.

A standard result in probability

Lemma: Let $m > 0$ and let E be an event that occurs with probability $1/m$. The expected number of independent trials until E occurs is m.

Optimal average running time

- Let m be the mean step size of the jumps.
  The set $\{s_1, \ldots, s_k\}$ mentioned earlier is chosen so that

$$\frac{1}{k} \sum_{j=1}^{k} s_j \quad \approx \quad m.$$

- The rear kangaroo requires, on average, $n/(4m)$ steps to catch up with front Kangaroo.
- From this point, roughly 1 in every m elements has been visited by the front kangaroo.
- Probability that the rear kangaroo lands on a “footprint” of the front kangaroo at each step is roughly $1/m$.
- By the previous lemma, the expected number of steps until a collision is m.

Optimal average running time (heuristic)

- So the total running time is $2(n/(4m) + m)$ steps (plus a little more to get to the next distinguished point after the collision)
- This is minimised by taking $m = \frac{1}{2}\sqrt{n}$.
- The average case running time is $2\sqrt{n}$ steps of the walk. (Technically: heuristic $(2 + o(1))\sqrt{n}$ group operations.)
- The algorithm may not terminate (either because there is no solution, or because the walks do not collide).
- This analysis was given by van Oorschot and Wiener in 1996.

Can one do better?

- Is $(2 + o(1))\sqrt{n}$ group operations optimal for this problem (still requiring low storage)?
- When $n = \#G$ then one can do better using a random walk algorithm based on the birthday paradox.
- The Pollard rho algorithm requires $(\sqrt{\pi n/2} + o(1)) \approx (1.25 + o(1))\sqrt{n}$ group operations on average (heuristic).
- It is based on collisions of the form $aP + bQ = cP + dQ$ for $P, Q \in G$

Pollard row

Pollard row

Can one do better?

- For the case $n \ll \#G$ there was no improvement in nearly 15 years until recent joint work with Pollard and Ruprai.
- Using three kangaroos we get an algorithm requiring $(1.819 + o(1))\sqrt{n}$ group operations.
- The idea is to shift the interval $[0, n]$ to $[-n/2, n/2]$ and to run kangaroos from $b = F(a)$, $-b = F(-a)$ and $u_0 = F(a_0)$ for suitable choice of a_0.
- Using further techniques and a slightly different algorithm (based on a variant of the birthday paradox) we get a running time of $(1.661 + o(1))\sqrt{n}$ operations.
- Recently my student Alex Fowler gave some evidence that this running time cannot be improved when using kangaroos.

Birthday paradox

- Suppose we sample uniformly at random from a set of size N. The expected number of trials until an element is sampled twice is $\sqrt{\pi N/2}$.
- When $N = 365$ this expected number is ≈ 23.94.
- Now sample uniformly at random from a set of size N and record each element in one of two lists.
  The expected number of trials until an element appears in both lists is $\sqrt{\pi N}$.
- The expected number of people in a room before there is a boy and a girl with the same birthday is ≈ 33.86.

Birthday puzzle

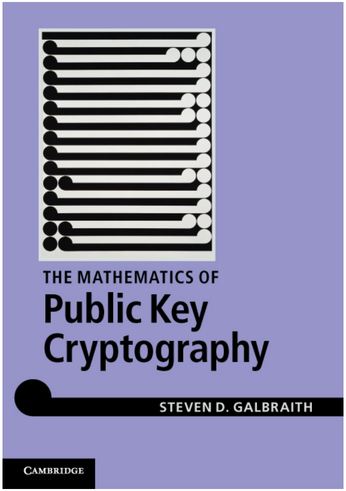
- In my hotel there is a meeting of the “boys born in January” club, and a meeting of the “random girls” club.
- I wish to invite some boys and girls into the lobby so that I have a boy and a girl with the same birthday.
- I wish to minimise the total number of people in the lobby.
- What ratio of each should I invite? (As $31 \to \infty$.)

Open questions and future work

- What is the best-possible constant c such that there exists a low-storage algorithm for the problem with running time $c\sqrt{n}$, when $n \ll \#G$?
- Can one match the $(1.25 + o(1))\sqrt{n}$ of Pollard rho?
- Can one do better in elliptic curve groups?
- What is the best average-case running time for a variant of the baby-step-giant-step algorithm?
  (Recent work by Bernstein and Lange gives the "grumpy giants" algorithm, but leaves open the possibility of better algorithms.)
- Similar ideas can be used to find paths between vertices in an expander graph in sub-linear time with low storage, but handling small cycles in the walks is inconvenient.

Thank you for your attention

For further details...


THE MATHEMATICS OF

Public Key Cryptography

STEVEN D. GALBRAITH

CAMBRIDGE