

The Britney Spears Problem

Why getting it *almost* right is OK
and
Why scrambling the data may help



Oops I made it again...



I'm Popular because I'm Popular

- With respect to the internet, answering:
Which of these is the most popular web search?

is a much much easier question than answering:

What

ch?



The image shows a screenshot of the Google UK search engine interface. At the top is the Google logo with 'UK' underneath. Below the logo is a search bar containing the text 'i like to ta'. To the right of the search bar is a link for 'Advanced Search'. Below the search bar is a table of search results. The first result is highlighted in blue and reads 'i like to tape my thumbs to my hands to see what it would be like to be a dinosaur' with 13,400 results. The other results are: 'i like to take my time' (141,000,000 results), 'i like to take long walks on the beach' (25,000,000 results), 'i like to tape my thumbs to my hands' (273,000 results), and 'i like to take photos' (135,000,000 results). At the bottom right of the table is a 'close' link. Below the table is a footer with links: 'Advertising Programmes - Business Solutions - About Google - Go to Google.com'.

Search Query	Number of Results
i like to tape my thumbs to my hands to see what it would be like to be a dinosaur	13,400 results
i like to take my time	141,000,000 results
i like to take long walks on the beach	25,000,000 results
i like to tape my thumbs to my hands	273,000 results
i like to take photos	135,000,000 results

Straightforward Approach

- Let's assume Google received their engine-search requests via one long data stream that they could read-in in real time...
- The straightforward solution would be to append new words to an array containing all words that have already been encountered and update a corresponding counter
- ..., "Yo dog", "Girls gone wild", "Dog ate chocolate", ...
 $\{yo=1, dog=2, girls=1, gone=1, wild=1, ate=1, chocolate=1\}$

Need for a constant-space algorithm

- Deciding whether to append the new word or increment a past counter might require an expensive search through the array
- But **more importantly**, the size of the array would be *astronomical* with no maximum cap on memory



Image credit: The very Google servers pictured above (trippy right?)

Majority Rule

- Imagine if the English language was dumbed down to a few words, or better yet... the integers 1 to 9
- Also, assume that one number (let's say 4) had the *majority* of the number instances. (This means $>50\%$ of the numbers are actually 4)
- With the “majority rule” method we would have two pieces of memory:
 - 1) the most common number up to that point (**maj**)
 - 2) a ‘counter’ that we associate with that number (**count**)

Majority Rule

- The rule is that we increment when we stream across the number stored in memory, and decrement otherwise. Example:

... 4
maj=4
count=1

... 4 4
maj=4
count=2

... 2 4 4
maj=4
count=1

... 1 2 4 4
maj=4
count=0

3 1 2 4 4
maj = 3
count=1

Majority Rule

- In this case, if 4 had actually been the majority, maj would have =4 when the stream was complete.
- Method is guaranteed to find the majority *if there is one*, but the number stored in memory at algorithm completion is *not* guaranteed to be a number with >50% of the occurrences
- Extend this to use an m number of maj variables to find the $n/(m+1)$ frequency. Example: use $m=99$ to find if a word appears in 1% of web search queries. Actually pretty robust!

Almost Right



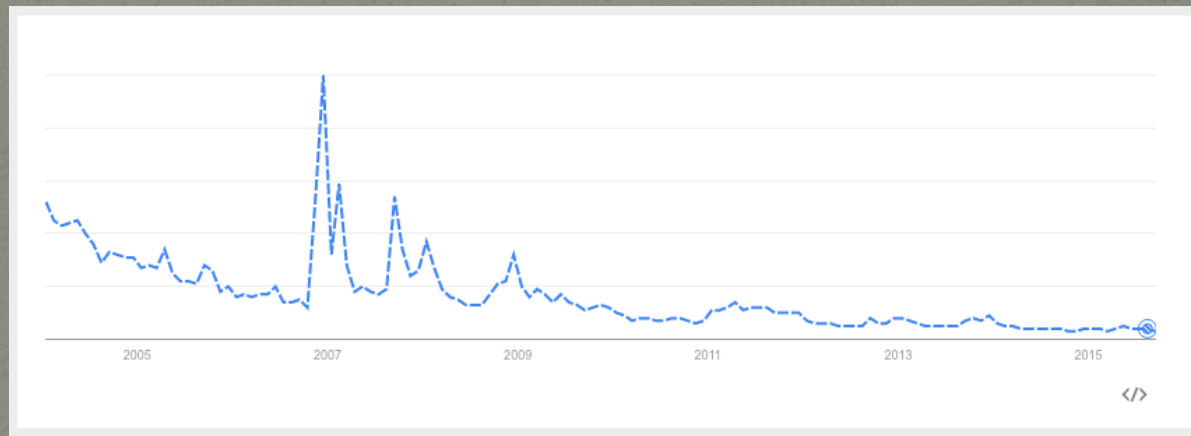
- Going back to the original straightforward method of appending to a huge array... what if we just removed the most infrequent elements every once in a while?
- This solution gives very good results, but we still have the unbounded space problem.
- This (along with Majority Rule) illustrates that we will not get the correct answer 100% of the time if we must obey the constant-space rule.
- But is that really all that bad?

Making a Hash

- A uniform random distribution actually has expected statistical properties (much like the standard normal distribution)
- A method used in computer science called “hashing” essentially bins and scrambles values that come from a unpredictable distribution to make them appear as if they are uniformly distributed.
- The bins can then be analyzed statistically to make generalizations about the data stream

Thanks, Britney!

You'll always be Number 1 in my book, even though the 90's misses you.



Reference:

Hayes, Brian. "The Britney Spears problem." *American Scientist* 96.4 (2008): 274.