Applied Algorithms Lec 24: Review

Sam McCauley

December 9, 2021

Williams College

Admin

- MM2 and MM3 back tonight. (MM2 is basically done.)
- Assignment "8" due tonight at 10 (will run test script at 7, 9, 10)
- Evaluations at the end of today
- Office hours today in my office from 2:30–4 (I may be a tad late)
- Please go to colloquium tomorrow! (It's in-person!)
    - I know you're all very busy. But we (and the speaker) will really appreciate it!
    - Cool topic: internet measurements. Combination of networking/security/privacy. Very relevant research, and from a technical point of view.

Looking Back at the Class

About the Class

- Goal: bridge the gap between theory and practice
- How can theoretical models better predict practice?
- Useful algorithms you may not have seen
- Using algorithmic principles to become better coders!

Pantry Algorithms

- Algorithms that you should always have handy because they are incredibly useful
- Bloom filters, linear programming, suffix arrays
- What drives the course
- Algorithmic understanding of these ideas!

Course topics

Throughout the course: efficient coding

- `gcc` and compiler decisions/flags
- `gprof` and `cachegrind`
- Making code more efficient
- Intrinsics
- SIMD operations

Course topics

Part 1: Time and space

- Meet in the middle
- Hirshberg's algorithm
- External memory model
- Blocking, external memory sorting

Course topics

Part 2: Randomization

- Basic probabilistic algorithm analysis
- Practical hashing
- Bloom Filters, cuckoo filters
- Count-Min Sketch, HyperLogLog counting
- Locality-sensitive hashing

Course topics

Part 3: Linear Programming

- LP
- ILP/MIP
- Solving algorithmic problems using LP, ILP, MIP
- Simplex solver (for LP)
- Branch and bound solver (for ILP/MIP)
- Using `GLPK` to solve LPs, ILPs, MIPs

Course topics

Part 4: Strings and Trees

- Burrows-Wheeler Transform
- Suffix array usage
- SA-IS algorithm

Review

Optimizing Code

- Cost of different operations
- Branch mispredictions
- Loop unrolling
- Making your intentions clear to the compiler (i.e. storing length of a string in a separate variable when you know the string doesn't change)

SIMD Instructions

- Operations on 256 bits at a time
- Can greatly speed up certain types of code
- Disadvantages:
  - Operations are a bit slower each (but they operate on more data)
  - Have to load and store data (expensive)
- Let's look at one example

SIMD Instructions

```
int simdFindFirst0(int* A, int size){
    __m256i vector0 = _mm256_set1_epi32(0);
    for(int i = 0; i < size/8; i++) {
        __m256i a = _mm256_loadu_si256((__m256i*)(A+8*i));

        //note that this works for comparing any two vectors
        //(there may be a better way to only look for 0s)
        __m256i res = _mm256_cmpeq_epi32(a, vector0);

        //res now has all 1s in locations where they are the same; all 0s in locations where they differ
        //(note that this may be counterintuitive)

        //movemask stores the first bit of each byte of res in the integer
        int integerResult = _mm256_movemask_epi8(res);

        if(integerResult != 0) {
            //found correct sequence of 8 bytes.  Now search for correct answewr
            for(int j = 8*i; j < 8*i + 8; j++) {
                if(A[j] == 0) return j;
            }
        }
    }
    return -1;
}
```

Locality-Sensitive Hashing for Searching for Close Points

- Reminder: a locality-sensitive hash is likely to hash close items together, but unlikely to hash far items together
- We saw: want expected size of a bucket to be $O(1)$
- Strategy: hash all items into buckets. Do all-compare all within each bucket
- If the close pair is not found, choose a new hash function and repeat

LSH with Repetitions

0	1	2	3	4
(101, 37, 65)	(103,37,64)	(91,84,3)	(100,18,79)	

0	1	2	3	4
	(101,37,65) (103,37,64)	(91,84,3)		(100,18,79)

0	1	2	3	4
(101, 37, 65)		(103,37,64)		(91,84,3) (100,18,79)

LSH Analysis

- Reminder: to make sure buckets are small, we concatenate multiple (k) minHashes hashes together
- These k minHashes will determine what bucket each item goes in
- Let's say that each pair of items (other than the close pair) has similarity $1/3$; this means they collide (under *one* hash, not all k) with probability $1/3$
- Let's look at some item x_i. How big is its bucket?

LSH Analysis

- Let's look at some item x_i. How big is its bucket?
- Let's use linearity of expectation! For any $j \neq i$, x_i and x_j wind up in the same bucket only if all k hashes have the same value
- $\Pr(h(x_i) = h(x_j)) = 1/3^k$.
- Let B_i be a random variable denoting the number of items that hash to the same bucket as x_i. Then $X_i = X_{i1} + X_{i2} + \ldots + X_{in}$, where $X_{ij} = 1$ if i and j hash to the same bucket
- By linearity of expectation, $E[X_i] = \sum_{j \neq i} E[X_{ij}]$
- X_{ij} is a 0-1 random variable, so $\mathrm{E}[X_{ij}] = \Pr(X_{ij} = 1) = 1/3^k$
- Therefore, $E[X_i] = (n-1)(1/3)^k$.

Analysis from Lecture 12

- We want buckets of size $O(1)$
- So we solve $n(1/3)^k = O(1)$, which implies $k = \log_3 n$
- Many people found that smaller k led to better running time. Why is that?
- Each time we go to a bucket, need to hash the item; need to access the bucket (probably a cache miss); some other smaller overheads
- So the $O(1)$ is actually pretty big; leading to a smaller k
- Cache misses are a big part of this: for practice, let's look at how to optimize with cache misses in mind.

What about cache misses?

- Let's analyze this algorithm in external memory
- How many cache misses does it take for a bucket of size X?
- Are there assumptions about our cache parameters that will affect this analysis?
- First, let's say that $X \leq M$. How many cache misses does it take?
    - Can bring the entire bucket into cache and do all-compare all. $O(1 + X/B)$ cache misses.
    - Where is the 1 from?? Is there a case where we don't have that extra 1?

What about cache misses?

- Let's analyze this algorithm in external memory
- How many cache misses does it take for a bucket of size X?
- Are there assumptions about our cache parameters that will affect this analysis?
- Now, let's say that $X \gg M$. How many cache misses does it take?
- (Don't worry about the case where $X \approx M$. Just deal with the cases where it's significantly larger or smaller.)
    - For each item in X, we do a linear scan through the bucket
    - $X \cdot O(\frac{X}{B})$ cache misses $= O(X^2/B)$

Intuition: How big do we want buckets to be?

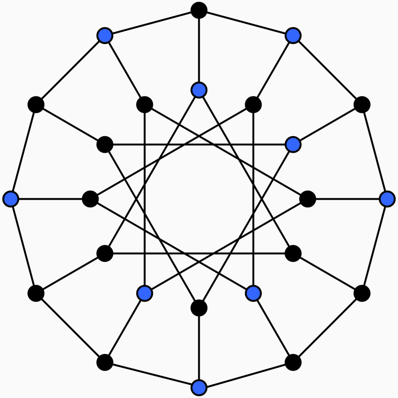
- It always takes at least one cache miss per bucket if $n \gg M$
- So let's shoot for $O(1)$ cache misses per bucket: expected size of a bucket is $O(B)$
- We set k:
  - $n(1/3)^k = O(B)$, so $k = \log_3 \frac{n}{B}$

Finishing the Analysis

- We have $k = \log_3 \frac{n}{B}$
- What is the expected number of repetitions?
- Probability that close pair is in the same bucket? (Let's say the close pair has similarity 3/4)
- $(3/4)^k$.
- So the expected number of repetitions until the close pair winds up in the same bucket is $1/(3/4)^k = (4/3)^k = (4/3)^{\log_3 \frac{n}{B}}$
- Let's simplify this on the board
- Hopefully we got: $(n/B)^{\log_3 4/3} \approx (n/B)^{.26}$

Questions about probability or external memory?

Independent Set (ILP practice)



- Given a graph G with nodes V and edges E
- Find the largest collection of vertices such that no two vertices in the collection are adjacent

Setting up the ILP

- Variables?
- Let $x_i = 1$ if vertex i is in the independent set; $x_i = 0$ otherwise. (Binary variables.)
- Objective?
- Maximize $\sum x_i$

Setting up the ILP

- Constraint?
- Need to make sure that no two vertices share an edge
- For every $e \in E$ (where $e = (v_i, v_j)$), have a constraint:
- $x_i + x_j \leq 1$.

ILP Wrapup

- Any questions about ILPs?
- May be an LP on the exam. Could we write an LP for this problem?
- No obvious way to do so; need x_i to be binary variables for $x_i + x_j \leq 1$ to force choosing one vertex
- (As you may know, this problem is NP hard, so it's likely impossible to write an LP for this problem.)

Review: Requested Topics

Bookshelf/Hirschberg's

- I was requested to go over how to use a Hirshberg's-like approach for other problems
- I started making some slides to go over the bookshelf problem
- They would definitely take at least 20 minutes to go over properly
- It's not on the final. So I don't want to take that time.

Hirschberg's takeaway

- (It is a cool algorithm that I do want you to know about in general)
- Also: Hirschberg's took more operations than normal edit distance, and required less space. But it ran faster. Why?
- Answer: cache-efficiency!
- Our Hirchberg's operations fit in cache, which sped things up
- Naive dynamic programming approach didn't fit in cache. Cache misses slowed us down.

Assgn 4 Prob 2

- Stream of 1,001,000 elements
- 1,000,000 elements only appear once $(a_1, \ldots, a_{1000000})$
- One element, q, appears 1000 times
- Goal for the next questions: determine number of columns, number of rows, number of bits for a Count-min sketch
- First: how does a count min sketch work? (Let's do it on the board)
- What are the guarantees?

Assign 4 Prob 2 par (a)

- Recall: 1,000,000 different a_i appear once; 1000 instances of q
- Let's say I query some a_i for an answer o_i, and some q for an answer o_q. How do I set my parameters so that 90% of the time, $o_q \geq o_i$?
- We know that $o_q \geq 1000$ by the CMS guarantees. So if we can make sure that 90% of the time, $o_i \leq 1000$ we are done
- CMS guarantees give: with probability $1 - \delta$, $o_i \leq 1 + 1001000\varepsilon$
- So we set $\delta = .1$, and $\varepsilon = 999/1001000$
- Number of rows $= \lceil \ln 1/\delta \rceil = 10$. Number of columns $= \lceil e/\varepsilon \rceil = 2724$. (Can tighten this a bit, but this is fine)
- Number of bits per element: $\lceil \log_2 1001000 \rceil = 20$

Assign 4 Prob 2 part (b)

- Recall: 1,000,000 different a_i appear once; 1000 instances of q
- Let's say I query *all* a_i for an answer o_i, and some q for an answer o_q. How do I set my parameters so that 90% of the time, $o_q \geq \max o_i$?
- Hint was to use the union bound

Assign 4 Prob 2 part (a)

- Recall: 1,000,000 different a_i appear once; 1000 instances of q
- CMS guarantees give: with probability $1 - \delta$, $o_i \leq 1 + 1001000\varepsilon$
- We still set $\varepsilon = 999/1001000$. But now we want this to happen more often
- We want the probability that $o_1 > o_q$, OR $o_2 > o_q$, OR $\ldots$ OR $o_{1000000} > q$ is at most .1
- By union bound: the probability that any of these happen is the sum of the probability that each happens
- So want $1000000\delta < .1$; $\delta = 1/10000000$.
- Number of rows $= \lceil \ln 1/\delta \rceil = 17$.

Assign 4 Prob 4 part (b)

- Have a stream of items with a majority item m that appears more than half the time
- How many rows do we need so that the count-min sketch is correct with constant probability?
- In other words: if we make any query q, we should be able to determine if $q = m$ or $q \neq m$ with constant probability.
- Let's go back to the count min sketch and see what this means

Looking at the CMS

- Our CMS is going to have one cell containing m
- This cell will have size at least $n/2$, where n is the length of the stream
- All other cells have size $< n/2$
- So on a query q: if $h(q)$ has size $> n/2$ we return $q = m$; otherwise we return $q \neq m$.

Analysis

- Let's say we have c entries in a row.
- If the correct answer is $q = m$, how does our CMS perform?
- Always gives the correct answer.
- If the correct answer is $q \neq m$, how does our CMS perform?
- Answers correctly if $h(q) \neq h(m)$
- So: answers correctly with probability $1/c$
- $c = O(1)$ is sufficient to obtain constant probability. (In fact $c = 2$ is enough.)

Any Questions about Assignment 4? Or CMS?

Any Other Questions?

Course Evaluations!

Course Evaluations

- Please do fill them out :)
- They're on Glow; course titled "Course Evaluations"
- Two kinds: main course evaluation that many people see (me, senior faculty in the department, admin); also "blue sheets" that only I see
- Hopefully clearly labelled

Course Evaluations Spiel

OPTIONAL SCRIPT FOR PROMOTING THE STUDENT COURSE SURVEY

Every term, Williams asks students to participate in end-of-semester course evaluations. Your feedback will help improve this course for other students taking it in the future, and help shape the **[department/program name]** curriculum.

You may skip questions that you don't wish to answer, and there is no penalty for choosing not to participate. All of your answers are confidential and I will only receive a report on your responses after I have submitted all grades for this course. While evaluations are open, I will receive information on how many students have filled out the evaluations, but I won't be told which of you have and haven't completed them. I won't know which responses are associated with which student unless you identify yourself in the comments.

To access the online evaluations, log into Glow (glow.williams.edu) using your regular Williams username and password (the same ones you use for your Williams email account). On your Glow dashboard you'll see a course called "Course Evaluations." Click on this and then follow the instructions on the screen. If you have trouble finding the evaluation, you can ask a classmate or reach out to Institutional Research at ir@williams.edu. The evaluations are open to you from now through the end of reading period. If you haven't filled it out by the beginning of reading period, you will start receiving email reminders.