CSC2420: Algorithm Design, Analysis and Theory Spring 2019

Allan Borodin

March 21, 2019

Week 10

Announcements:

- If you did not get an acknowledgement from me about your project please let me know. I think I only have 6 project proposals thus far and I want to make sure that I haven't missed any.
- I plan to add all or most of the remaining assignment problems by tomorrow.

Todays agenda

Todays agenda: Mainly today will be a brief introduction to a number of topics filling in some additional information in the following week(s).

- The ROM model and some applications.
- Walk Sat type algorithms
- Sublinear time algorithms
- Streaming algorithms

The random order model

We left off last lecture just mentioning the ROM model. As mentioned, the ROM model was first introduced in the classical secretary problem. It has now been adopted to many problem settings, and has led to a number of related topics.

There is a basic observation (next slide) that was made by Karande, Mehta and Tripathi (2011) in the context of studying the performace of the Ranking algorithm in the ROM model. This observation applies more generally to any "online problem".

Note: Unfortunatley, when we say "online", most people mean "online" where the input sequence is given by an adversary (i.e. worst case analysis). But here I mean "online algorithm" in the sense that input items arrive one at a time and the algorithm must make an immediate decision for each item and where the input sequence can come from any external (without the algorithm being able to control the sequence) source.

The ROM and i.i.d input models

As we mentioned previously, in the ROM model an adversary chooses a set of inputs , say $\{I_1, I_2, \ldots, I_n\}$ and then "nature" creates a random input sequence $a_{\pi(1)}, a_{pi(2)}, \ldots, a_{\pi(n)}$ where π is chosen with uniform probability over the set of all $n!$ permutations.

In the i.i.d. model, the input is generated by randomly and independently selecting each input item I_j from a distribution $\mathcal{D}$.

Fact (Karande et al): Suppose algorithm $\mathcal{A}$ for an online problem $\mathcal{P}$ has (expected) competitive ratio c with respect to the ROM model. (Note: that this is the worst ratio over any possible set of input items.) Then $\mathcal{A}$ achieves competitive ratio at least c for any (unknown) distribution.

Proof of the basic "ROM imples i.i.d " fact

The proof is remarkably simple given how useful is this fact. Consider the algorithm on problem instances consisting on n input items. Partition the input instanes into classes, each of size $n!$ such that the class is made up of the $n!$ ways to permute some set of input items. Each input sequence in a class occurs with the same probability. Thus each class becomes an instance of the random order model and hence algorithm $\mathcal{A}$ has competitive ratio at least c on each class. We csn then take the expectation over the differnent clssses to obtain the desired result.

In particular then, for the online bipartite matching problem, since the Ranking algorithm can be viewed as a simple deterministic greedy algorithm (call it algorithm $\mathcal{G}$ in the ROM model, it folows that $\mathcal{G}$ has competitive ratio $1 - \frac{1}{e}$ in the unknown i.i.d. input model.

Here the greedy algorithm $\mathcal{G}$ assumes the offline nodes have some (implied) ordering π and $\mathcal{G}$ assigns each online node to the first (wrt π) offline node (if any) that is still not yet matched.

What is known about the bipartite matching problem in the ROM, known and unknown i.i.d. models

We know that any positive (existance of a good ratio) ROM result implies the same for the unknown i.i.d. input model and this in turn implies a ratio for the known the i.i.d model. And, of course, the contrapositive tells us that any inapproximations for the known i.i.d. model imply the same negative result for the ROM model.

For i.i.d. inputs , there is a distinction between integral types (where the expected number of arrivals of any given input type is integral) vs arbitrary types.

At this point in time, as far as I know, the best results in the unknown i.i.d. model follow from what is known for the ROM model. However, much better competitive ratios have been shown for the known i.i.d. input model especially as regards integral arrival rates.

What is known about the bipartite matching problem in the ROM and known i.i.d. models

- In the ROM model, Mahdian and Yan (2011) show that the (randomized) Ranking algorithm obtains the ratio .696 improving upong the fist result in this regard by Karande et al. who also show that ranking is no better that .727 in the ROM model.
- The best known i.i.d. result (for non-integral arrival rates) is .702 by Manshadi et al (2012) who also have the best inappoximation .823 for any BMM algorithm in the known i.i.d. model.
- For the known i.i.d. input model with integral arrival rates, Brubach et al (2016) establish a .7299 competitive ratio slightly improving upon the .7293 result due to Jaillet and Lu (2013). Furthermore, the .7299 result holds for the (offline) vertex weighted version of the problem.
- Finally, with regard to known i.i.d., integral types, Brubach et al have a .705 competitve ratio for the general edge weighted version of bipartite matching which substantially beats the previous best .668 by Hauptler et al (2011).

The classical secretary problem

As stated, there are n candidates (n is known) who arrive (in the ROM model). Each candidate has a positive value which is revealed upn arrival. Here first is the algorithm that achieves a $\frac{1}{e}$ bound on the probability that the algorithm will choose the candidate having the best value.

The secretary problem algorithm

$i := 1$
$best := 0$
While $i \leq n/e$
  If $b_i > best$ **then** $best := v_i$
  $i := i + 1$
EndWhile
While $i \leq n - 1$
  If $v_i > best$ **then** return v_i and halt
Return v_n % This is the case where we did not find a value better than the best value in the first n/e candidates.

The approximation bound

The approximation bound

The "sample and take next best" algorithm chooses the best value with probability at least $\frac{1}{e}$ for all input instances in the ROM model. It follows that the expected value of the algorithm is at least that fraction of the best value.

Given how classical and interesting is this result, there are many proofs of the result as well as many variants. The history of the problem is by its self quite fascinating.
Here is a very simple argument (but not a tight analysis) for a weaker bound. Instead of sampling n/e candidates, sample $n/2$ candidates and then choose the first candidate in the second half that is better than the best in the first half.

Claim: The probability that this algorithm chooses the best candidate is *at least* $\frac{1}{4}$.

Proof of the weaker $\frac{1}{4}$ bound

Proof of the weak bound: Given the random ordering of the inputs, with probability $\frac{1}{4}$ the second best candidate is in the first $n/2$ inputs and the best candidate is in the last $n/2$ candidates.

Why is this argument not a tight analysis?

Proof of the weaker $\frac{1}{4}$ bound

Proof of the weak bound: Given the random ordering of the inputs, with probability $\frac{1}{4}$ the second best candidate is in the first $n/2$ inputs and the best candidate is in the last $n/2$ candidates.

Why is this argument not a tight analysis?

What we were saying in the above argument is the following: If we let A^1 be the event that second best candidate is in first half and
B^1 is the event that best candidate is in the second half.
Then the above was almost a precise bound for

$$Prob[A^1 \wedge B^1] = Prob[A^1|B^1] \cdot Prob[B^1]$$

But this is actually an underestimate as thee are other events where we will obtain the maximum value candidate. For example, let A^2 be the event that the third best candidate occurs in the first half, and B^2 be the event that the best candidate and the second best occur in the second half with the second best coming after the best. And, of course, we could continue this with more events in which the simple algorithm would succeed in choosing the best candidate.

The idea of the $\frac{1}{e}$ bound

In the analysis, we don't worry about picking the last element if we run out of candidates.

- The first observation is that the the optimal solution is obtained by considering the the optimal stopping rule in the followng online scenario: An algorithm has to decide on when to stop considering candidates amongst the first i seen based on just the relative value of the i^{th} canddidate having seen the first i canddiates with no information about the future.
- In the random order model we then need to understand
  $p_i = Prob$[selecting the best candidate at position i]
- This is equal to the probability:
  $Prob[i$ picked $|$ being best so far$] \cdot Prob[$ best so far$]$
  The first Probability is at most $(1 - \sum_{j<i} p_j)$
  The second probability is $1/i$.
- That is, we need to : Maximize ip_i
  subject to $p_i \leq (1 - \sum j < ip_j)\frac{1}{i}$.

The idea of the $\frac{1}{e}$ analysis continued

- This was just the primal LP we gave last lecture.
  Some solutions then analyze this LP and its dual and show that in the limit as $n \to \infty$, the optimal solution is n/e.
- Some other solutions just more directly anaylze the inequality and use the approximation $p_i = i \int_i^n \frac{1}{t} dt = -i \ln i$ and the differentate to obtain the solution.
- So far I haven't found a purely combinatorial analysis. However, the following extension to edge weighted bipartite matching does provide a combinatorial analysis for the expected weight of a solution.
  Aside: Since we are considering the worst case ROM result (i.e. over all inout]it sets) this is also a tight bound on the expected value. Namely, we can choose an input set where the value of the best candidate is arbitrarily large while the other candidates have arbitrarily small values.

Extending the expected bound to the edge weighted bipartite matching problem

The secretary problem can be seen as a special case of the edge weighted bipqrtite matching problem where there is only one offline node.

For the ROM model, Kesselheim et al (2013) extend the sampling idea in the secretary problem to obtain the same $\frac{1}{e}$ approxination of the optimal weight solution.

Algorithm 1. Bipartite online matching

Input : vertex set R and cardinality $n = |L|$
Output: matching M

Let L' be the first $\lfloor n/e \rfloor$ vertices of L;
$M := \emptyset$;
for each subsequent vertex $\ell \in L - L'$ **do** `// steps` $\lceil n/e \rceil$ `to` n
    $L' := L' \cup \ell$;
    $M^{(\ell)}$:= optimal matching on $G[L' \cup R]$; `// e.g. by Hungarian method`
    Let $e^{(\ell)} := (\ell, r)$ be the edge assigned to ℓ in $M^{(\ell)}$;
    if $M \cup e^{(\ell)}$ is a matching **then**
        add $e^{(\ell)}$ to M;

Proof sketch of the Kesselheim et al extension to edge weighted bipartite matching

Aside: First note that L is the set of online nodes and in other papers, L is often the offline nodes.
As discuussed in their paper, they way they think about the random choice of an online node occuring in the k^{th} iteration (wrt to the random permutation of the ROM nodes in L) is that a a random set of size k is chosen and then iteratively each node is drawn from this set and removed.

They then want to determine a bound on the contribution of the ℓ^{th} online node for $\ell \in \{\lceil n/e \rceil, \ldots, n\}$. Denote this by $\mathbb{E}[A_\ell]$.

Main lemma

$$\mathbb{E}[A_\ell] \geq \frac{\lfloor n/e \rfloor}{\ell - 1} \cdot \frac{OPT}{n}$$

Continution of the $\frac{n}{e}$ proof for edge weighted bipartite matching

Let r be the offline node to which ℓ is matched (and let this matching edge be e^ℓ in the optimal offline matching for M^ℓ using the notation in the algorithm. The proof of the main lemma shows that the probability

1. $Prob[e^\ell]$ will be available is at least $\frac{\lfloor n/e \rfloor}{\ell - 1}$.
2. If chosen the expected value of $e^\ell \geq \frac{OPT}{n}$.

Given the previous main Lemma, the bound for the expected value for the matching M computed by the Kesselheim et al algorithm is

$$\mathbb{E}[w(M)] \geq \sum_{\ell=\lceil n/e \rceil}^{n} \frac{\lfloor n/e \rfloor / e}{\ell - 1} \cdot \frac{OPT}{n} = \frac{\lfloor n/e \rfloor}{n} \sum_{\ell=\lceil n/e \rceil}^{n-1} \frac{1}{\ell} \cdot OPT$$

Some easy approximations then show that $\mathbb{E}[w(M)] \geq \frac{1}{e} - \frac{1}{n}$.

Many other extension of the basic secretary problem

There are many other extenions of the secretary problem as well as other application of the ROM model to other types of problems.

- Matroid secretary problems. The most basic matroid is the uniform matroid (i.e. choose k candidates for some k)
- Knapsack secretary

A related set of problems is to have the input items generated from a known i.i.d. stochastic setting or more generally from an i.d. (i.e. independent but not necessarily identical) setting where the i^{th} candidate is drawn from a distribution $\mathcal{D}_i$. There are a variety of such problems which are called *prophet inequality* problems.

Walk-Sat type algorthms

Continuing the theme of conceptually simple algorithms, I want to mention algorithms based on random walks in a graph for the k-SAT problem. In some sense the basic idea is close to the random changing of variables.

We already saw that randomly settng the propositional variables in a propositional formula can wind up satifying a large number of clauses especially clauses with many literals.

But the same naive idea is very unlikely to satisfy all clauses even if the formula is satisfiable. For exact k-CNF, the expected number of clauses satisfied will be a $\frac{2^k-1}{2^k}$ fraction of all clauses and the probability of saifying all clauses will drop exponentially with m, the number of clauses.

But lets start with a random truth assignment (or even start with an arbutrary truth assignment). What variables might we want to change if our initial solution is not a satisfying assignment?
Let's start with a tractable case, 2-SAT which we we know has a polynomial time algorithm. But note that Max-2-SAT is NP-hard.

The random walk algorithm for 2-Sat

- First, here is the idea of the deterministic polynomial time algorithm for 2-Sat: We can first eliminate all unit clauses. We then reduce the problem to the directed $s - t$ path problem. We view each clause $(x \vee y)$ in F as two directed edges $(\bar{x}, y)$ and $(\bar{y}, x)$ in a graph G_F whose nodes are all possible literals x and $\bar{x}$. Then the formula is satisfiable iff there does not exist a variable x such that there are paths from x to $\bar{x}$ and from $\bar{x}$ to x in G_F.
- There is also a randomized algorithm for 2-SAT (due to Papadimitriou [1991]) based on a random walk on the line graph with nodes $\{0, 1, , n\}$. We view being on node i as having a truth assignment τ that is Hamming distance i from some fixed satisfying assignment τ^* if such an assignment exists (i.e. F is satisfiable).
- Start with an arbitrary truth assignment τ. If $F(\tau)$ is true then we are done; else find an arbitrary unsatisfied clause C and randomly choose one of the two variables x_i occurring in C and now change τ to τ' by setting $\tau'(x_i) = 1 - \tau(x_i)$.

The expected time to reach a satisfying assignment

- When we randomly select one the the two literals in C and complement it, we are getting close to τ^* (i.e. moving one edge closer to node 0 on the line) with probability at least $\frac{1}{2}$. (If it turns out that both literal values disagree with τ^*, then we are getting closer to τ^* with probability $= 1$.)
- As we are proceeding in this random walk we might encounter another satisfying assignment which is all the better.
- It remains to bound the expected time to reach node 0 in a random walk on the line where on each random step, the distance to node 0 is reduced by 1 with probability at least $\frac{1}{2}$ and otherwise increased by 1 (but never exceeding distance n). This perhaps biased random walk is at least as good as the case where we randomly increase or decrease the distance by 1 with probability equal to $\frac{1}{2}$.

Claim:

The expected time to hit node 0 is $O(n^2)$.

An elementary proof

Let $T_{i+1,i}$ be the expected time to go from node $i+1$ to node i.
We have $T_{i+1,i} \leq \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \left(T_{i+2,i+1} + T_{i+1,i}\right)$
Rearranging we have $T_{i+1,i} \leq 1 + T_{i+2,i+1}$
This implies $T_{i+1,i} \leq (n-i)$ since $T_{n,n-1} = 1$.

Thus $T_{n,0} \leq T_{n,n-1} + \cdots\cdots + T_{1,0} \leq \frac{n(n-1)}{2}$.

Even though we have a proof, it is still instructive to see how the random walk bound on the line follows from a more general analysis of random walks on a finite connected graph.

A proof using the basics of finite Markov chains

- A finite Markov chain M is a discrete-time random process defined over a set of states S and a matrix $P = \{P_{ij}\}$ of transition probabilities.
- Denote by X_t the state of the Markov chain at time t. It is a memoryless process in that the future behavior of a Markov chain depends only on its current state: $Prob[X_{t+1} = j | X_t = i] = P_{ij}$ and hence $Prob[X_{t+1} = j] = \sum_i Prob[X_{t+1} = j | X_t = i] Prob[X_t = i]$.
- Given an initial state i, denote by r_{ij}^t the probability that the first time the process reaches state j occurs at time t;
  $r_{ij}^t = Pr[X_t = j \text{ and } X_s \neq j \text{ for } 1 \leq s \leq t-1 | X_0 = i]$
- Let f_{ij} the probability that state j is reachable from initial state i;
  $f_{ij} = \sum_{t>0} r_{ij}^t$.
- Denote by h_{ij} the expected number of steps to reach state j starting from state i (hitting time); that is, $h_{ij} = \sum_{t>0} t \cdot r_{ij}^t$
- Finally, the *commute time* c_{ij} is the expected number of steps to reach state j starting from state i, and then return to i from j; $c_{ij} = h_{ij} + h_{ji}$

Stationary distributions

- Define $\mathbf{q}^t = (q_1^t, q_2^t, \ldots, q_n^t)$, the state probability vector (the distribution of the chain at time t), as the row vector whose i-th component is the probability that the Markov chain is in state i at time t.
- A distribution π is a stationary distribution for a Markov chain with transition matrix P if $\pi = \pi P$.
- Define the underlying directed graph of a Markov chain as follows: each vertex in the graph corresponds to a state of the Markov chain and there is a directed edge from vertex i to vertex j iff $P_{ij} > 0$. A Markov chain is *irreducible* if its underlying graph consists of a single strongly connected component. We end these preliminary concepts by the following theorem.

Theorem: Existence of a stationary distribution

For any finite, irreducible and aperiodic Markov chain,

(i) There exists a *unique* stationary distribution π.

(ii) For all states i, $h_{ii} < \infty$, and $h_{ii} = 1/\pi_i$.

Back to random walks on graphs

- Let $G = (V, E)$ be a connected, non-bipartite, undirected graph with $|V| = n$ and $|E| = m$. A uniform random walk induces a Markov chain M_G as follows: the states of M_G are the vertices of G; and for any $u, v \in V$, $P_{uv} = 1/deg(u)$ if $(u, v) \in E$, and $P_{uv} = 0$ otherwise.
- Denote by $(d_1, d_2, \ldots, d_n)$ the vertex degrees. M_G has a stationary distribution $(d_1/2m, \ldots, d_n/2m)$.
- Let $C_u(G)$ be the expected time to visit every vertex, starting from u and define $C(G) = \max_u C_u(G)$ to be the *cover time* of G.

Theorem: Aleliunas et al [1979]

Let G be a connected undirected graph. Then

1. For each edge (u, v), $C_{u,v} \leq 2m$,
2. $C(G) \leq 2m(n-1)$.

- It follows that the 2-SAT random walk has expected time at most $2n^2$. to find a satisfying assignment in a satisfiable formula.

Extending the random walk idea to k-SAT

- The random walk 2-Sat algorithm might be viewed as a drunken walk (and not an algorithmic paradigm). Or we could view the approach as a local search algorithm that doesn't know when it is making progress on any iteration but does have confidence that such an exploration of the local neighborhood likely to be successful over time.
- We want to extend the 2-Sat algorithm to k-SAT. However, we know that k-SAT is NP-complete for $k \geq 3$ so our goal now is to improve upon the naive running time of 2^n, for formulas with n variables.
- In 1999, Following some earlier results, Schöning gave a very simple (a good thing) random walk algorithm for k-Sat that provides a substantial improvement in the running time (over say the naive 2^n exhaustive search) and this is still almost the fastest (worst case) algorithm known.
- This algorithm was derandomized by Moser and Scheder [2011].
- Beyond the theoretical significance of the result, this is the basis for various Walk-Sat algorithms that are used in practice.

Schöning's k-SAT algorithm for $k \geq 3$

The algorithm is similar to the 2-Sat algorithm with the difference being that one does not allow the random walk to go on too long before trying another initial random starting assignment. The result is a one-sided error alg running in time $\tilde{O}[(2(1 - /1k)]^n$; i.e. $\tilde{O}(\frac{4}{3})^n$ for 3-SAT, etc.

Randomized k-SAT algorithm

Choose a random assignment τ
Repeat 3n times % n = number of variables
If τ satisfies F then stop and accept
Else Let C be an arbitrary unsatisfied clause
  Randomly pick and flip one of the literals in C
End If

Claim

If F is satisfiable then the above succeeds with probability p at least $(2 - \frac{2}{k})^{-n}$. The expected time is then the inverse of this probability.

Skecth of the claim

We will sketch the idea along the lines of the elementary proof for the case of $k = 2$ glossing over some details.

Assume that the formula is satisfiable. So then again we can think of the analysis of finding some fixed satisfying assignment τ (where it can only help if we stumble across another satisfying assignment before hitting τ.

Now for a given starting assignment, let p_i be the probability that starting at node i, a random walk will eventually reach node 0 (i.e. and hence find τ). Now the thing we are glossing over is that we are ending the random walk process in $3n$ steps and not letting it run indefinitely. The claim is that this will not alter the probability substantially.

Sketch of claim proof continued

We can replace the inequalities by equalites in bounding p_i and by the same reasoning as in the elementary proof, we get the recurrence:

$$p_{i+1} = \frac{1}{k} p_{i-1} + \frac{k-1}{k} p_{i+1}$$

Rewriting:

$$p_{i+1} = \frac{k}{k-1} p_i - \frac{1}{k-1} p_{i-1}$$

So now we have a recurrence that we want to solve by induction. But we need a basis for the recurrence and in particular we need p_1.

Recognizing that going from node 2 to node 1 is the same as going from node 1 to node 0, we get $p_1 = \frac{1}{k} + \frac{k-1}{k}(p_1)^2$
One of the roots of this quadratic equation is $\frac{1}{k-1}$ which leads us to try the recurrence $p_i = \frac{1}{(k-1)^i}$. This is easily then shown to hold by induction.

Finishing sketch of proof of claim

This is the anslysis for a given initial assignment at node i. We need to average this over all possible assignments which is then

$$\sum_{i=0}^{n} \binom{n}{i} \frac{1}{(-1)^i} = (2 - \frac{2}{k})^{-n}$$

More general, we may want to solve k variable CSP's with d-ary variables.

A similar anaysis establishes a $\tilde{O}\big(d(1 - \frac{1}{k})\big)^n$ time complexity.

How good are these bounds for k-SAT?

For 3-SAT, the Schöning bound is $\tilde{O}(\frac{4}{3})^n$.

It is a basic open problem as to whether or not we can achieve a^n for some $a < \frac{4}{3}$ (or for some a substantially better than $\frac{4}{3}$.

A more basic question is whether or not 3-SAT requires a^n for some $a > 1$?

The Exponential Time Hypothesis (ETH) states that a^n for some $a > 1$ is necessary.

We also observe that as k increases the base of the time complexity is limiting to 2. The basic question for arbitrary CNF formulas (i.e. arbitrary k) is whether or not time complexity can be a^n for some $a < 2$.

The Strong Exponential Time Hierarchy (SETH) states that SAT cannot be done in time a^n for $a < 2$.

These are much stronger assumptions thn $P \neq NP$ and (perhaps surprisingly) relates to the topic of *fine-grained complexity*.

Sublinear time and sublinear space algorithms

We now consider other contexts in which randomization is (with few exceptions) provably essential. In particular, we will study sublinear time algorithms and then the (small space) streaming model.

- An algorithm is sublinear time if its running time is $o(n)$, where n is the length of the input. As such an algorithm must provide an answer without reading the entire input.
- Thus to achieve non-trivial tasks, we almost always have to use randomness in sublinear time algorithms to sample parts of the inputs.
- The subject of sublinear time algorithms is a big topic and we will only present a very small selection of hopefully representative results.
- The general flavour of results will be a tradeoff between the accuracy of the solution and the time bound.
- This topic will take us beyond search and optimization problems.

A deterministic exception: estimating the diameter in a finite metric space

- We first conisder an exception of a "sublinear time" algorithm that does not use randomization. (Comment: "sublinear in a weak sense".)
- Suppose we are given a finite metric space M (with say n points x_i) where the input is given as n^2 distance values $d(x_i, x_j)$. The problem is to compute the diameter D of the metric space, that is, the maximum distance between any two points.
- For this maximum diameter problem, there is a simple $O(n)$ time (and hence sublinear in n^2, the number of distances) algorithm; namely, choose an arbitrary point $x \in M$ and compute $D = \max_j d(x, x_j)$. By the triangle inequality, D is a 2-approximation of the diameter.
- I say sublinear time in a weak sense because in an implicitly represented distance function (such as d dimensional Euclidean space), the points could be explicitly given as inputs and then the input size is n and not n^2.

Sampling the inputs: some examples

- The goal in this area is to minimize execution time while still being able to produce a reasonable answer with sufficiently high probability.
- We will consider the following examples:
  1. Finding an element in an (anchored) sorted linked list [Chazelle,Liu,Magen]
  2. Estimating the average degree in a graph [Feige 2006]
  3. Estimating the size of some maximal (and maximum) matching [Nguyen and Onak 2008] in bounded degree graphs.
  4. Examples of property testing, a major topic within the area of sublinear time algorithms. See Dana Ron's DBLP for many results and surveys.

Finding an element in an (anchored) sorted list

- Suppose we have an array $A[i]$ for $1 \leq i \leq n$ where each $A[i]$ is a pair (x_i, p_i) with $x_1 = \min\{x_i\}$ and p_i being a pointer to the next smallest value in the linked list. That is, $x_{p_i} = \min\{x_j | x_j > x_i\}$. (For simplicity we are assuming all x_j are distinct.)
- We would like to determine if a given value x occurs in the linked list and if so, output the index j such that $x = x_j$.

A $\sqrt{n}$ algorithm for searching in an anchored sorted linked list

If $x < x_1$, then x is not in the list.
Let $R = \{j_i | 0 \leq i \leq \sqrt{n}\}$ be $\sqrt{n}$ randomly chosen indices plus the index 1.
Access these $\{A[j_i]\}$ to determine k such that x_k is the largest of the accessed array elements less than or equal to x.
From $A[k]$, search forward $2\sqrt{n}$ steps in list to see if and where x exists

Claim:

This is a one-sided error algorithm that (when $x \in \{A[i]\}$) will fail to return j such that $x = A[j]$ with probability at most $1/2$.

Estimating average degree in a graph

- Given a graph $G = (V, E)$ with $|V| = n$, we want to estimate the average degree d of the vertices.
- We want to construct an algorithm that approximates the average degree within a factor less than $(2 + \epsilon)$ with probability at least $3/4$ in time $O(\frac{\sqrt{n}}{poly(\epsilon)})$. We will assume that we can access the degree d_i of any vertex v_i in one step.
- Like a number of results in this area, the algorithm is simple but the analysis requires some care.

The Feige algorithm

Sample $8/\epsilon$ random subsets S_i of V each of size (say) $\frac{\sqrt{n}}{\epsilon^3}$
Compute the average degree a_i of nodes in each S_i.
The output is the minimum of these $\{a_i\}$.

The analysis of the approximation

Since we are sampling subsets to estimate the average degree, we might have estimates that are too low or too high. But we will show that with high probability these estimates will not be too bad. More precisely, we need:

1. Lemma 1: $Prob[a_i < \frac{1}{2}(1-\epsilon)\bar{d}] \leq \frac{\epsilon}{64}$
2. Lemma 2: $Prob[a_i > (1+\epsilon)\bar{d}] \leq 1 - \frac{\epsilon}{2}$

The probability bound in Lemma 2 is amplified as usual by repeated trials. For Lemma 1, we fall outside the desired bound if any of the repeated trials gives a very small estimate of the average degree but by the union bound this is no worse than the sum of the probabilities for each trial.

End of Week 10 lecture

We ended here having just introduced sublinear time algorithms.

Next week we will continue the discussion of sublinear time and then move on to streaming algorithms. I will soon post an advance set of slides for Wweek 11.