CME 305: Discrete Mathematics and Algorithms

TA: Andreas Santucci

Midterm Practice Problems – 02/14/17

Probabilistic Method

Let X be a non-negative random variable with expectation $\mathbb{E}[X] = \mu \geq 0$ and variance σ^2. Prove that for all $\lambda > 0$,

$$\Pr(X \geq \lambda) \leq \frac{\lambda\mu + \sigma^2}{\lambda^2 + \sigma^2}.$$

We first recall Markov's Inequality, that for non-negative RV X, $\Pr(X \geq a) \leq \frac{\mathbb{E}[X]}{a}$.[1] Then,

$$\Pr(X \geq \lambda) = \Pr(\lambda X \geq \lambda^2) = \Pr(\lambda X + \sigma^2 \geq \lambda^2 + \sigma^2).$$

Now, define random variable $Y = \lambda X + \sigma^2$. Then $\mathbb{E}[Y] = \lambda\mu + \sigma^2$ by linearity of expectation. We apply Markov's inequality, with $a = \lambda^2 + \sigma^2$, and we're done.

Let F be a finite collection of binary strings of length at most k and assume no member of F is a prefix of another one. Let N_i be the number of strings of length i in F. Prove that

$$\sum_{i=1}^{k} \frac{N_i}{2^i} \leq 1.$$

Hint: consider generating a random binary string of length k.

Consider a random string of length exactly k, call it S. We generate each bit of S using a Bernoulli random variable with success probability $1/2$. We want to calculate $\Pr(S \in \mathcal{F})$. We know this probability at most one.

Consider the prefix's of S of length $1, 2, \ldots, k$, i.e. call S_1 the prefix of S containing only the first bit, and S_2 the prefix of S containing only the first two bits, and continue in this fashion to define S_k to be the prefix-string of length k (i.e. S itself).

Realize that because no member of F is a prefix of another one, that

$$\Pr(\{S_i \in F \cap S_j \in F\}) = 0$$

by the problem statement, and the fact that by construction S_i a prefix of S_j for $i < j$. Hence using a union bound (with non-overlapping events)

$$\Pr(\exists S_i \in F) = \Pr(\cup_{i=1}^{k}(S_i \in F)) = \sum_{i=1}^{k} \Pr(S_i \in F) = \sum_{i=1}^{k} \frac{N_i}{2^i} \leq 1,$$

[1]To show this, use the fact that $\mathbb{E}[\cdot]$ a monotone operator and $a \geq 0$:

$$\mathbb{1}_{\{x \geq a\}} a \leq x \implies \Pr(X \geq a) \leq \frac{\mathbb{E}[X]}{a}.$$

where the last inequality follows from the fact that $\Pr(\exists S_i \in F)$ is a probability, hence must be at most one. Notice that in general, union bound doesn't necessarily give a bound less than or equal to unit value, hence the argument for non-overlapping events is critical.

> Imagine n cars, each of which travels at a different speed. Initially, the cars are queued in uniform random order at the start of a semi-infinite, one-lane highway. Each car drives at the minimum of its maximum speed and the speed at which the car in front of it is driving. The cars will form "clumps". What is the expected number of clumps?

We are given a permutation $\pi[n]$. Consider the last car in the sequence, call it car i. It forms a clump $\iff$ it is slower than all cars in front of it.[2]

Since the car speeds are uniformly distributed, the ith car will be slower than all other cars in front with probability $1/i$. Hence we have a recursive relation.

$$f(i+1) = \underbrace{\frac{1}{i+1}(1+f(i))}_{\text{car } i+1 \text{ forms a clump}} + \underbrace{\frac{i}{i+1}(f(i))}_{\text{does not form a clump}} = \frac{1}{i+1} + f(i).$$

Realize that in the base case where we consider $\pi[1]$, the only car forms its own clump. Hence $f(1) = 1$. Hence by unrolling our recursion, we see that

$$f(n) = \sum_{i=1}^{n} \frac{1}{i} = H_n.$$

Alternate Solution Alternatively, we may consider the *fastest* car of the n-cars. Realize that it forms its own clump if and only if it is in front of all other cars; otherwise, there exists a car slower in front of it, and it gets absorbed in another clump. Since the speeds uniformly distributed, the fastest car is the first car with probability $1/n$. Having considered the fastest car, we now are left with $n-1$ cars, and we have our recursion.

Flow

> At lunchtime it's crucial for people to get to the food trucks as quickly as possible. The building is represented by a graph $G = (V, E)$ where each room, landing, or other location is represented by a vertex and each corridor or stairway is represented by an edge. Each corridor has associated capacity c, meaning that at most c people can pass through the corridor at once. Traversing a corridor from one end to the other takes one timestep and people can decide to stay in a room for the entire timestep.
>
> Suppose all people are initially in a single room s, and that the building has a single exit t. Give a polynomial time algorithm to find the fastest way to get everyone out of the building.

[2] $\rightarrow$) If the last car is traveling slower than all cars in front of it, then it will never catch up to any of the other cars. Hence the last car will remain by itself, forming its own clump. $\leftarrow$) If the last car forms a clump, then by definition its the leader of its own clump. Suppose it travels faster than a car in front of it. Then at some point, it must catch up to this car. But in this case the last car is not responsible for forming its own clump, and we get a contradiction. Hence if the last car forms a clump, it must be traveling slower than all other cars in front of it.

We solve this problem by modifying the input graph, then applying Ford Fulkerson's max flow algorithm.[3] We first fix a deadline time k, such that we need to empty the building by time k under the above conditions. Once we design our algorithm, we solve our problem by increasing k by unit value until we find the smallest such value where everybody can exit. Let the total number of people be P.

Constructing the Graph For each room $r \in V$, make $k+1$ copies of it $r_0, \ldots, r_k$; each will correspond to room r at a particular timestep. For each pair of rooms r and w with a hallway capacity of c_{rw} between them, create directed edges with weight c_{rw} each: $r_i \to w_{i+1}$ and $w_i \to r_{i+1}$, for each time $i = 0, 1, \ldots, k-1$. For each room r, add directed edge $r_i \to r_{i+1}$ with weight P for each time $i = 0, 1, \ldots, k-1$. Finally, add single room t, and connect t_k to t with edge capacity P. On this new graph, run FF with source s and target t.

If a plan exists, above algorithm returns flow of size P or more If we route units of flow as we route people in such a scheme, and then send all P units of flow from t_k to t, we clearly obtain a flow of size P. Since this flow feasible, and since FF finds a max flow, we know FF will find a flow with size greater than or equal to P.

If FF finds flow of P or more, this corresponds to a valid plan Note the flow must have value exactly P, since the cut defined by $\{t\}$ has size P. We obtain a routing as follows. For each timestep $i = 1, 2, \ldots, k$, look for all pairs of rooms r, w such that people are moving from $r_{i-1} \to w_i$ and from $w_{i-1} \to r_i$. Notice that although each capacity constraint respected (i.e. no more than c_{rw} people are crossing in either direction), the total number of people utilizing the hallway between r and w in both directions may be greater than c_{rw}. To remedy this, for all rooms r and w and each time-step k: consider the people "swapping" from r to w, suppose there are a of them; we leave them in place, and let the rest cross the hallway as normal. This operation doesn't change the number of people in every room after the ith timestep. The operation also preserves integer-weighted, non-negative, feasible flow. Hence the flow obtained after all swaps still has size P. We convert this into an evacuation plan by routing each person as a distinct unit of flow in the algorithm.

Runtime FF runs on a graph with m edges and max-flow f in $O(mf)$ work. If our original graph has m edges, our modified graph has km edges. Since the max-flow of this graph at most P, the runtime of FF on modified graph is $O(kmP)$. To address overcrowding in hallways (from flow in either direction), we consider m pairs of connected edges at each of the k time-steps and check if there is a swap between them. This phase takes $O(mk)$ time. Thus the algorithm upper-bounded by $O(kmP)$. Via original observation, increasing k one by one will give an algorithm running in poly time.

Trees

Let T be a tree. Let $\Delta(T) = d$, i.e. max-degree is d. Show that T has at least d leaves.

Fix attention to the node of max-degree. If this node is the root of the tree, it has d children. If it's not the root of the tree, perform a Breadth First Search starting from said node to explore the graph, in

[3]Solution by Arun Jambulapati.

the process construct a BFS tree.[4] Starting at the root, we traverse down the tree through each of the d children. Since it's a tree, the paths we proceed down must never intersect (if they do, we yield a cycle, contradicting that T is a tree). Our tree T finite, so eventually we must get to d leaves.

A minimum bottleneck spanning tree of a weighted graph G is a spanning tree of G that minimizes the maximum weight of any edge in the spanning tree. Prove that a minimum spanning tree is also minimum bottleneck spanning tree, but that the converse is not true.

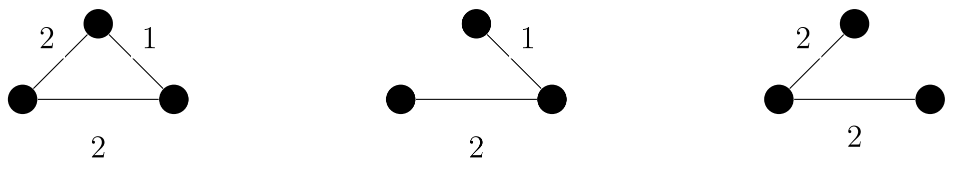
An MST is also an MBST, via cut property Notice that a MST respects the cut property, whence it's also an MBST. To see this, consider MST T, and MBST B. Let $e = (i, j)$ denote the heaviest weight edge used in T; we show that $e \in B$ as well, which shows that an MST also minimizes the max weight of any edge in the spanning tree, i.e. an MST also an MBST.

Consider the cut defined by removing e from MST T. We are left with a forest with two trees, call them L, R. By the cut property, each edge e' crossing the cut has weight at least that of e. Examine the edges of B which also cross the cut; they must too all have weight at least $c(e)$.[5] But we started with e as the bottleneck edge in the MST T. Hence

$$\text{weight}(e) \leq \text{weight}(e') \quad \text{for each } e' \in \substack{\text{cut defined by} \\ \text{removing } e \text{ from } T}$$

But we know that B has minimum bottleneck cost among trees T', i.e. bottleneck$(B) \leq$ bottleneck(T'), whence T also a MBST.

Not every MBST is an MST, via counter example Consider a weighted triangle with two edges of weight 2 and one edge of weight 1. The MST of this graph is any length two path containing the weight 1 edge, but the tree formed by weight 2 edges will have the same bottleneck cost (of 2) as any MST, but it has strictly more cost.



Weighted Graph G

An MST T with weight 3

An MBST B with bottleneck 2 but cost 4

Basic Graph Theory and Definitions

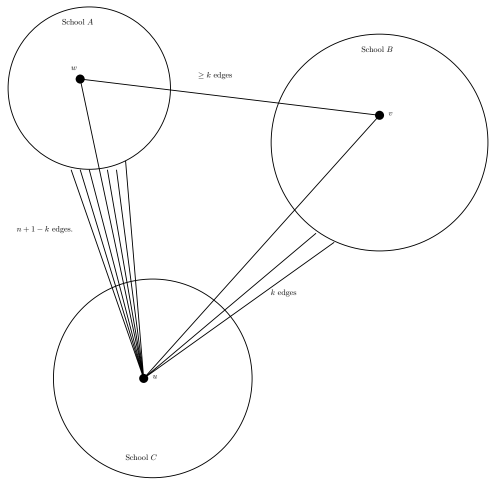
In a village there are three schools with n students in each of them. Every student from any of the schools is on speaking terms with at least $n + 1$ students from the other two schools. Show that we can find three students, no two from the same school, who are on speaking terms with each other.

[4]This is equivalent to picking up the graph from said node and "dangling" it, i.e. we "arbitrarily" make said max-degree node the root of the tree.

[5]If not, we get a contradiction that e is the smallest weight edge crossing the cut.

We have a tri-partite graph where we draw an undirected edge from node u in school A to node v in school $B \iff u$ is on speaking terms with v. We do this for each possible relationship across school boundaries.

Consider a node u in school A which has exactly $n+1-k$ neighbors in school B and k neighbors in school C, where we have chosen u such that k as small as possible. Consider a neighbor of u in C, call it v. It remains to show that v has a neighbor in B in common with one of u's $n+1-k$ neighbors in B. But we started with selecting u for the sole reason that k as small as possible, hence v must have at least k neighbors in B.



One of those neighbors of v must also be a neighbor of u using Pigeonhole principle; if there were no overlap between these two neighborhoods, it would imply that school B has more than n children, a contradiction. Hence we have realized a triangle (u, v, w) wherein we have found three students (no two from the same school) who are on speaking terms with each other.

A connected k-regular graph on n nodes is one in which all vertices have degree k. Prove that the diameter of a connected k-regular graph is $O(n/k)$.

Denote the diameter of the graph by d, and let u, v be two vertices satisfying that $\text{dist}(u, v) = d$. Perform a BFS starting from node u, and let ℓ_i denote the set of vertices w such that $\text{dist}(u, w) = i$. Denote by P the shortest-path from u to v.

Realize that since we started with a BFS tree rooted at u, and since $\text{dist}(u, v) = d = \text{diam}(G)$ is the longest shortest path, it must be the case that the BFS tree has exactly height d. Hence it must be the cast that

$$\cup_{i=0}^{d} \ell_i = V$$

i.e. all n nodes of our graph contained in the first d layers of our BFS tree.[6]

Realize that $\ell_i \cap P \subseteq (\ell_{i-1} \cup \ell_{i+1})$, i.e. the neighbors of ℓ_i which are also in our path P must be found only in the layer before and after.[7] Consider any layer ℓ_i in our tree for $1 < i < d$ (i.e. we exclude the root node in ℓ_0 and the last layer ℓ_d, itself containing v). Notice that since ℓ_i part of a BFS tree, that each node in ℓ_i must be connected to (at least one) parent in ℓ_{i-1}. Each node in ℓ_i must also be connected to exactly k

[6]If this weren't the case, and there existed some other node not in the first d layers, it means there is a shortest path of length $> d$ between u and said node, in which case we get a contradiction with the diameter of our graph being d.

[7]If not, we get a contradiction with P being a shortest path, since we can "shortcut" between layers, still traversing $u \rightsquigarrow v$.

neighbors, since our graph k-regular; in general, these k neighbors could themselves be in $\ell_{i-1}, \ell_i, \ell_{i+1}$ *only.* [8] Since we started with $i < d$, we know that there is also a "children layer" ℓ_{i+1}. We know that between these three layers, there must be at least k nodes. I.e. we have $d/3$ "epochs" in our tree, each containing at least k nodes. We know that $\frac{d}{3} \cdot k \leq n \iff d \leq \frac{3n}{k} \implies d = O(n/k)$.

Show that every graph on m edges has a subgraph on at least $m/2$ edges which is bipartite.

Consider our randomized max-cut algorithm, whereby we select each node for inclusion in our cut with probability $p = 1/2$. Then realize that an edge appears in our cut if and only if exactly one of its incident nodes is in the cut, which happens with probability $1/2$. [9] This holds for each edge, hence by linearity of expectation, there exists a cut of size $m/2$.

Consider the sub-graph induced by taking only these $m/2$ edges. Realize that for each edge, exactly one node incident to our cut and one node not in the cut, i.e. we have a bi-partition.

Prove that a graph with $2n$ nodes and minimum degree n must be connected.

Assume toward contradiction that the graph G split into $k > 1$ components. Fix attention to any two vertices u, v in different components. By definition of being in different components, they have no common neighbors, i.e. $|\Gamma(u) \cap \Gamma(v)| = 0$. Also note that $|\Gamma(u) \cup \Gamma(v)| \leq 2n - 2$ since if u, v are not directly connected then their neighborhoods exclude each other, and further our graph only has $2n$ nodes. We now use inclusion-exclusion principle,

$$2n - 2 \geq |\Gamma(u) \cup \Gamma(v)| = \underbrace{|\Gamma(u)|}_{\geq n} + \underbrace{|\Gamma(v)|}_{\geq n} - \underbrace{|\Gamma(u) \cap \Gamma(v)|}_{=0}$$

But this suggests that $2n - 2 \geq 2n$ which is a contradiction. Whence our graph must be connected.

[8] If it had a neighbor in $\cup_{j=0}^{i-2} \ell_j$, we get a contradiction with how we constructed our BFS tree. Note that it's OK to have neighbors in ℓ_i since our original graph not a tree. It's also OK to have neighbors in ℓ_{i+1} since each layer $1 < i < d$ in our tree has children. However, note that there can *not* be neighbors in any of $\ell_{i+2}, \ldots, \ell_d$, for if there were, said node should actually belong in layer ℓ_{i+1}, since it's reachable with a direct edge from ℓ_i.

[9] Fix attention to edge $e = (u, v)$. Realize that each of u, v could be in or out of the cut, hence there are four possibilities. Two possibilities involve exactly one of u or v chosen for inclusion in S; these are disjoint events each with probability $1/4$, hence with probability $1/2$ edge e is in the cut. With probability $1/4$, both $u, v \in S$ and with probability $1/4$, both $u, v \notin S$. This partition accounts for our sample space.