Massachusetts Institute of Technology
6.856J/18.416J: Randomized Algorithms
David Karger

April 9, 2021

Problem Set 6 Solutions

Problem 1. We break ties between equal-weight vertices according to the index of the vertices (the larger-index vertex is unmarked).

(a) Note that each iteration, at least one node remains marked and is added to S (particularly, the smallest-weight smallest-index vertex). Therefore, there are a finite number of iterations. Moreover, at each iteration the marked vertices form an independent component and are not connected to any of the existing vertices in S. Thus, the algorithm yields an independent set. It is maximal since we stop when the graph is empty (and if the set would not be maximal, then there would be at least one more vertex remaining in the graph and the algorithm could not have stopped).

(b) Note that equal weights occur with probability at most $n \cdot n/n^4 = n^{-2}$.

Assuming there are no equalities, v stays marked if it has the lowest weight among the $d+1$ weights of v and $\Gamma(v)$. This happens with probability $1/(d+1)$. Thus, v stays marked with probability $\geq 1/(d+1) - n^{-2}$.

(c) We prove that a constant fraction of edges are deleted in an iteration. Consider that all the weights are different (this happens with probability $\geq 1 - n^{-2}$).

Next, consider a "good vertex" v (as per definition 12.3 from the book); v has degree d. We will argue that the probability that v is in $\Gamma(S)$ with probability $\Omega(1)$. Note that v will certainly be in $\Gamma(S)$, if one of it's lower-degree neighbors are in S. To lower bound this probability, we consider two events:

(E1) at least one of the lower-degree neighbors of v get assigned a score of less than n^4/d. Let u be such a neighbor of v.

(E2) all neighbors of u have score higher than n^4/d.

We note that the probability of event (E1) is at least $1-(1-1/d)^{d/3} \geq 1-e^{-1/3}$. Conditioned on (E1), the probablility of (E2) is at least $(1-1/d)^d \geq 1/4$ (for $d \geq 2$). Note that when conditioned on (E1), the neighbors of u don't necessarily get independent and uniform score over $[n^4]$, since there may be common neighbors of u and v. However, this dependence only increases the probability of (E2). Thus, the overall probability that a "good" v is in $\Gamma(S)$ is at least $(1/4)(1 - e^{-1/3})$ (except for the case of $d = 0$, when the vertex v remains marked with probability 1).

Thus, probability that $v \in S \cup \Gamma(S)$ is at least $(1/4)(1-e^{-1/3})-n^{-2}$. Subsequently, a good edge is deleted with probability at least $(1/4)(1-e^{-1/3})-n^{-2}$. Therefore,

the expected number of deleted good edges is a constant fraction of m. Since number of good edges is $\geq m/2$, a constant fraction of edges will be deleted in an iteration. By theorem 1.3 (or by the analysis in the book), the expected number of iterations is $O(\log m) = O(\log n)$.

Thus, the algorithm above is an RNC algorithm (all the operations are clearly implementable in parallel $O(\log n)$ time).

Slightly different analysis: We analyze events (E1') and (E2'), which are the same as (E1) and (E2) except we set the threshold at $n^4/2d$ (factor of two smaller). We can then analyze probability of (E2') using a union bound as follows: a neighbor of u has lower score than u probability $1/2d$ so union bounding over all neighbors of u (there are only at most d of them), this has probability at most $1/2$. Hence, with probability at least $1/2$ all neighbors of u have higher score, and u will stay marked.

Different analyses are also possible.

Problem 2. We note that after randomly partitioning vertices into two groups, a particular edge is in the cut if its two incident vertices are in opposite groups. This occurs with probability $1/2$ as long as the two incident vertices choose groups independently from one another. In general, all edges are in the cut with probability $1/2$ as long as all vertices choose groups pairwise independently. We saw in lecture that we can produce n pairwise independent bits using only $O(\log n)$ fully independent bits. That is, there exists a pairwise-independent hash family $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}\}$ such that for all $i \neq j$, it holds that $\Pr[h(i) \neq h(j)] = 1/2$.

One choice of such a hash family is as follows. Let $s = s_0 \dots s_{\lceil \log n \rceil}$ be our random seed. For $i \in [n]$, let $i_1 \dots i_{\lceil \log n \rceil}$ be the binary representation of i. The hash function $h_s(i) = \bigoplus_j (s_j \wedge i_j)$. Note that $|\mathcal{H}| = O(n)$. This hash family is pairwise independent because: for any $i \neq j$, the event $h(i) = h(j)$ occurs with probability exactly $1/2$, since i and j always differ on at least one bit.

For any $h \in \mathcal{H}$, we can define a cut, $\mathrm{Cut}(h) = (h^{-1}(0), h^{-1}(1))$, that is, vertices v with $h(v) = 0$, on one side and vertices v with $h(v) = 1$ on the other side. It is easy to see that for any pairwise-independent hash family $\mathcal{H}$, it holds that,

$$\mathbb{E}_{h \sim \mathcal{H}} |\mathrm{Cut}(h)| \;=\; \sum_{(i,j) \in E(G)} \Pr[h(i) \neq h(j)] \;=\; m/2$$

We conclude by observing that trying all of the $O(n)$ different seeds of length $\lceil \log n \rceil$ and taking the largest cut produces a cut of size at least $m/2$ with certainty, since the expectation of the cut size over the choice of seed is at least $m/2$. Furthermore, this computation can be done in NC: for each seed, we can count the number of cut edges using logarithmic depth, then we can compute the cut size for all seeds and compare in parallel using only logarithmic additional depth. The number of processors needed is equal to the number of edges in the graph times the number of different seeds, which is polynomial.

Problem 3. (a) We modify the weights of edges as follows: multiply all weights by $2m^2 + 10$, add to each edge a random perturbation drawn from $\{1 \dots 2m\}$. Find the min-weight perfect matching, wrt to the new weights, using the RNC algorithm described in the class.

We claim that the min-weight perfect matching wrt to new weights is:

- also a min-weight perfect matching wrt the initial weights
- is unique with probability at least $1/2$.

To see the correctness of first claim, let M be the value of the min-weight perfect matching (wrt to initial weights), and M' the value of the min-weight perfect matching (wrt to new weights). Note that $(2m^2+10)M \leq M' \leq (2m^2+10)M + 2m^2 < (2m^2+10)(M+1)$. This means that if a minimum weight perfect matching is found (wrt to new weights), then it has minimum weight perfect matching wrt to initial weights too.

For the uniqueness, we want to apply the Isolating lemma. Here, we might be tempted to apply the lemma only to the added perturbations sampled from $\{1, ..., 2m\}$. Unfortunatley, we run into trouble if we naively apply the lemma to all matchings, as there might be a matching with a uniquely smallest perturbation, but it might not correspond to one with the minimum total weight (initial weight scaled + perturbation). Intuitively, the perturbation *should* still produce a unique minimum, and there are two ways to formalize this:

- The beauty of the Isolating lemma is that it is agnostic to the families of items under consideration, so we can just restrict the set of matchings that we apply the Isolating Lemma to: it suffices to only consider the family of matchings that are minimum weight perfect matchings wrt to initial weights. One of those has a uniquely smallest perturbation, and hence will have a uniquely smallest minimum total weight.
- We can apply the Isolating lemma to the *total* new weights (rather than just the perturbations). This is because if we inspect the proof of the lemma, the random weights didn't actually need to be exactly from $\{1, ..., 2m\}$–the weight of each edge just needed to be sampled uniformly from a set of at least $2m$ possible values.

We've shown that with probability $\geq 1/2$, there is a unique minium weight matching wrt new weights, so the algorithm from class can be used to find the perfect matching, and it will also be min-weight wrt to initial weights.

(b) If an edge has weight w then the corresponding Tutte matrix entry is given value 2^w for the determinant calculation. Thus, if the input weights have a polynomial number of bits, then the number in the matrix will have an exponential number of *bits* so we will not be able to perform arithmetic operations on the efficiently.

(c) Given the graph G, build a complete bipartite graph G' where edge (i, j) has weight 1 if the $(i, j) \in G$ and weight 2 if $(i, j) \notin G$. Any matching in G that has

k edges corresponds to a matching in G' of total weight $k + 2(n - k) = 2n - k$. Thus a matching of minimum weight in G' corresponds to a maximum matching in G.

Problem 4. We collapse all 0 length edges and merge their endpoints since distances don't change modulo these edges (we can transform paths in the new graph back to the to paths in the original graph trivially). Let the distance matrix be D.

Consider nodes i, j, and k, where i and k are neighbors. We know that D_{ij} and D_{ik} can differ by at most B. This suggests it will be enough to work with distances modulo $2B + 1$.

Let $A^{(r)}$ be the "adjacency" matrix for edges of length r (i.e. the (i, j) entry is 1 iff i and j have an edge of length r). Let $R^{(d)}$ the distance matrix of paths of length d modulo $2B + 1$ (i.e. the (i, j) entry is 1 iff i and j are distance d modulo $2B + 1$ apart).

Suppose we know that i and j are at distance d, and i's next hop to j is some neighbor k at distance r away. Then, if we do a Binary Witness Matrix multiplication (BWM) of $A^{(r)}$ and $R^{(d-r)}$, then k will be a valid witness for the (i, j) entry of the product. Conversely, for any other witness k' that we might get, we know k' has an edge of length r to i and $D_{k'j} \equiv d - r \pmod{2B + 1}$. But since $D_{k'j}$ can't be more than B units different from $D_{i,j}$, we know that in fact $D_{k'j} = d - r$, and so k' is in fact on the next hop from i to j. In other words, we have shown that the (i, j) entry of BWM of $A^{(r)}$ and $R^{(d-r)}$ will reveal a next hop k going from i to j if $D_{ij} = d$ and $D_{ik} = r$.

In order to do this for all possible distances d modulo $2B + 1$ and for all hop sizes from 1 to B, we just need to compute BWM of $A^{(r)}$ and $R^{(d)}$ for each choice of $1 \leq r \leq B$ and $0 \leq d < 2B + 1$. After each BWM for a particular choice of r and d, we will check for each (i, j) entry in the result whether we have a witness k and if $D_{ij} \equiv r + d \pmod{2B + 1}$ and if so update the successor ("next hop") matrix $H_{ij} = k$.

The total time is dominated by $O(B^2)$ invokations of BWM for $n \times n$ matrices, for a total of $O(B^2 MM(n) \log^2 n)$.

Problem 5.

Problem 6. (a) Let v be a vertex with degree $d \geq D/2$. v is deleted in a round if at least one of its d neighbors adds itself to the independent set, which happens with probability at least $1 - (1 - p)^d \geq 1 - (1 - 1/2D)^{D/2} \geq 1 - e^{-1/4}$. The talking node has at most D neighbors, so by union bound, it adds itself to the independent set with probability at least $1 - pD = \frac{1}{2}$. Thus, the probability that v is deleted is at least $\frac{1}{2}(1 - e^{-1/4})$, which is constant as desired.

We say v is bad if it has degree at least $D/2$ and is not deleted, and we say v good otherwise. After each round, v turns from bad to good with probability at least $\frac{1}{2}(1 - e^{-1/4})$. Thus, after $O(\log n)$ rounds, v is good with high probability. By taking a union bound over all vertices with degree at least $D/2$, we conclude that all of them will be good with high probability after $O(\log n)$ rounds, which means the maximum degree of the graph will be $D/2$.

(b) We repeat the previous part $O(\log n)$ times. Each time the maximum degree of the graph is divided by two with high probability, so after $O(\log n)$ times, the maximum degree of the remaining graph becomes 0, when we can add all vertices to the independent set.

(c) As in the hint, we simulate one talking-while-listening round with several talking-or-listening rounds. Each round, every vertex which wishes to talk will do so with probability 1/2. We will repeat this at least $3 \log n$ times.

In each round, for any pair of neighboring vertices i and j for which i wants to talk, let X_{ijk} be the indicator variable event that i successfully communicates its message to j in round k. Note that $X_{ijk} = 1$ if and only if i talks and j doesn't. Since i talks with probability exactly 1/2 and j talks with probability at most 1/2 (perhaps j doesn't wish to talk at all), we have $\mathbb{E}[X_{ijk}] \geq 1/4$. Noting that the X_{ijk} are independent for fixed i, j and varying k, we can apply the Chernoff bound to compute $\Pr\left(\sum_k^{3 \log n} X_{ijk} < 1\right) \leq n^{-3}$. There are $O(n^2)$ pairs i, j so by the union bound, the probability that there is *any* pair i, j such that $X_{ijk} = 0$ for every k is at most n^{-1}, and thus the talking-while-listening round succeeds with high probability.