DM582 Solutions

Mads Anker Nielsen
madsn20@student.sdu.dk

March 4, 2024

This document contains written solution to exercise problems from the course DM582 (spring 2024). The solutions given here might differ from the solutions discussed in class. In class, we place more emphasis on the intuition leading to the correct answer. Please do not consider reading these solutions an alternative to attending the exercise classes.

References to CLRS refer to the book *Introduction to Algorithms, 4th edition* by Cormen, Leiserson, Rivest, and Stein.

This document will inevitably contain mistakes. If you find some, please report them to me (Mads) so that I can correct them.

Sheet 3

KT, Exercise 1

Exercise

3-Coloring is a yes/no question, but we can phrase it as an optimization problem as follows. Suppose we are given a graph $G = (V, E)$, and we want to color each node with one of three colors, even if we aren't necessarily able to give different colors to every pair of adjacent nodes. Rather, we say that an edge (u, v) is satisfied if the colors assigned to u and v are different. Consider a 3-coloring that maximizes the number of satisfied edges, and let c^* denote this number. Give a polynomial-time algorithm that produces a 3-coloring that satisfies at least $\frac{3}{2}c^*$ edges. If you want, your algorithm can be randomized; in this case, the expected number of edges it satisfies should be at least $\frac{3}{2}c^*$.

Suggested solution

We consider the number of edges satisfied by the straightforward randomized algorithm which colors each vertex with one of 3 colors independently and uniformly at random. For each edge (u, v), the probability that the edge is satisfied is exactly the probability that u has a color different from v, which is $\frac{2}{3}$. Let X_e be the indicator random variable which takes the value 1 if e is satisfied and 0 otherwise. Let $X = \sum_{e \in E} X_e$ be the total number of satisfied edges. By linearity of expectation,

$$E(X) = \sum_{e \in E} E(X_e) = \sum_{e \in E} \frac{2}{3} = \frac{2}{3}|E| \geq \frac{2}{3}c^*$$

where the last inequality holds since c^* is at most the number of edges in the graph.

KT, Exercise 2

Exercise

Consider a county in which 100,000 people vote in an election. There are only two candidates on the ballot: a Democratic candidate (denoted D) and a Republican candidate (denoted R). As it happens, this county is heavily Democratic, so $80,000$ people go to the polls with the intention of voting for D, and $20,000$ go to the polls with the intention of voting for R. However, the layout of the ballot is a little confusing, so each voter, independently and with probability $1/100$ votes for the wrong candidate – that is, the one that he or she didn't intend to vote for. (Remember that in this election, there are only two candidates on the ballot.) Let X denote the random variable equal to the number of votes received by the Democratic candidate D, when the voting is conducted with this process of error. Determine the expected value of X, and give an explanation of your derivation of this value.

Suggested solution

Let X_D be the random variable which denotes the number of democratic voters who vote for D and let X_R be the random variable which denotes the number of republican voters who vote for D. Then $X = X_D + X_R$. Now,

$$E[X_D] = 0.99 \cdot 80000$$

since the probability that a democratic voter votes for D is 99%. Similarly,

$$E[X_R] = 0.01 \cdot 20000$$

since the probability that a republican voter votes for D is 1%. By linearity of expectation,

$$E[X] = E[X_D] + E[X_R] = 0.99 \cdot 80000 + 0.01 \cdot 20000 = 79400.$$

KT, Exercise 3

Exercise

In Section 13.1, we saw a simple distributed protocol to solve a particular contention-resolution problem. Here is another setting in which randomization can help with contention resolution, through the distributed construction of an independent set.

Suppose we have a system with n processes. Certain pairs of processes are in conflict, meaning that they both require access to a shared resource. In a given time interval, the goal is to schedule a large subset S of the processes to run – the rest will remain idle – so that no two conflicting processes are both in the scheduled set S. We'll call such a set S conflict-free.

One can picture this process in terms of a graph $G = (V, E)$ with a node representing each process and an edge joining pairs of processes that are in conflict. It is easy to check that a set of processes S is conflict-free if and only if it forms an independent set in G. This suggests that finding a maximum-size conflict-free set S, for an arbitrary conflict G, will be difficult (since the general Independent Set Problem is reducible to this problem). Nevertheless, we can still look for heuristics that find a reasonably large conflict-free set. Moreover, we'd like a simple method for achieving this without centralized control: Each process should communicate with only a small number of other processes and then decide whether or not it should belong to the set S.

We will suppose for purposes of this question that each node has exactly d neighbors in the graph G. (That is, each process is in conflict with exactly d other processes.)

(a) Consider the following simple protocol.

Each process P_i independently picks a random value x_i; it sets x_i to 1 with probability $\frac{1}{2}$ and sets x_i to 0 with probability $\frac{1}{2}$. It then decides to enter the set S if and only if it chooses the value 1, and each of the processes with which it is in conflict chooses the value 0.

Prove that the set S resulting from the execution of this protocol is conflict-free. Also, give a formula for the expected size of S in terms of n (the number of processes) and d (the number of conflicts per process).

(b) The choice of the probability $\frac{1}{2}$ in the protocol above was fairly arbitrary, and it's not clear that it should give the best system performance. A more general specification of the protocol would replace the probability $\frac{1}{2}$ by a parameter p between 0 and 1, as follows.

Each process P_i independently picks a random value x_i; it sets x_i to 1 *with probability p and sets x_i to* 0 *with probability* $1-p$. *It then decides to enter the set S if and only if it chooses the value* 1*, and each of the processes with which it is in conflict chooses the value* 0*.*

In terms of the parameters of the graph G, give a value of p so that the expected size of the resulting set S is as large as possible. Give a formula for the expected size of S when p is set to this optimal value.

Suggested solution

For $v \in V$ corresponding to a process P_i, let x_v denote the value x_i chosen by process P_i.

(a) We start by showing that the resulting set S is indeed conflict-free (independent). Suppose S is not conflict-free and let $u, v \in S$ be such that v and u are in conflict ($uv \in E$). $u \in S$ so $x_u = 1$ and $x_{u'} = 0$ for all neighbors u' of u. In particular, $x_v = 0$. But then $v \notin S$, a contradiction.

For all $v \in V$ let X_v be the indicator random for the event that $v \in S$. Then $X_v = 1$ iff $x_v = 1$ and $x_{v'} = 0$ for all neighbors v' of v. v has exactly d neighbors and $x_{v'}$ is set to 0 independently and with probability $\frac{1}{2}$ for all neighbors v' of v. Thus,

$$E[X_i] = P[v \in S] = \frac{1}{2}\left(\frac{1}{2}\right)^d = \frac{1}{2^{d+1}}.$$

Let $X = \sum_{v \in V} X_v$. Then X is a random variable whose value is the size of S. By linearity of expectation,

$$E[X] = \sum_{v \in V} E[X_v] = \frac{n}{2^{d+1}}.$$

(b) Using the notation from the previous part, we have

$$E[X_v] = P[v \in S] = p\,(1-p)^d$$

and

$$E[X] = \sum_{v \in V} E[X_v] = np\left(1-p\right)^d .$$

Thus, we must pick p to maximize $p(1-p)^d$ in order to maximize $E[X]$. We can do this taking the derivative of $p(1-p)^d$ with respect to p and solving for 0. The equation to solve is

$$\frac{\mathrm{d}}{\mathrm{d}p} p(1-p)^d = -(1-p)^{d-1}(dp + p - 1) = 0$$

which has the solution $p = 0$ or $p = \frac{1}{d+1}$. Picking $p = 0$ is not optimal since this results in $E[X] = 0$. Thus, the optimal value of p is $p = \frac{1}{d+1}$. Substituting this value into the formula for $E[X]$, we get

$$E[X] = n\left(\frac{1}{d+1}\right)\left(1 - \frac{1}{d+1}\right)^d .$$

We notice that $\left(1 - \frac{1}{d+1}\right)^d$ approaches $\frac{1}{e}$ as d becomes large.

KT, Exercise 4

Exercise

A number of peer-to-peer systems on the Internet are based on overlay networks. Rather than using the physical Internet topology as the network on which to perform computation, these systems run protocols by which nodes choose collections of virtual "neighbors" so as to define a higher-level graph whose structure may bear little or no relation to the underlying physical network. Such an overlay network is then used for sharing data and services, and it can be extremely flexible compared with a physical network, which is hard to modify in real time to adapt to changing conditions.

Peer-to-peer networks tend to grow through the arrival of new participants, who join by linking into the existing structure. This growth process has an intrinsic effect on the characteristics of the overall network. Recently, people have investigated simple abstract models for network growth that might provide insight into the way such processes behave, at a qualitative level, in real networks.

Here's a simple example of such a model. The system begins with a single node v_1. Nodes then join one at a time; as each node joins, it executes a protocol whereby it forms a directed link to a single other node chosen uniformly at random from those already in the system. More concretely, if the system already contains nodes $v_1, v_2, \ldots, v_{k-1}$ and node v_k wishes to join, it randomly selects one of $v_1, v_2, \ldots, v_{k-1}$ and links to this node.

Suppose we run this process until we have a system consisting of nodes $v_1, v_2, \ldots, v_n$; the random process described above will produce a directed network in which each node other than v_1 has exactly one outgoing edge. On the other hand, a node may have multiple incoming links, or none at all. The incoming links to a node v_j reflect all the other nodes whose access into the system is via v_j; so if v_j has many incoming links, this can place a large load on it. To keep the system load-balanced, then, we'd like all nodes to have a roughly comparable number of incoming links. That's unlikely to happen here, however, since nodes that join earlier in the process are likely to have more incoming links than nodes that join later. Let's try to quantify this imbalance as follows.

(a) Given the random process described above, what is the expected number of incoming links to node v_j in the resulting network? Give an exact formula in terms of n and j, and also try to express this quantity asymptotically (via an expression without large summations) using $\Theta(\cdot)$ notation.

(b) Part (a) makes precise a sense in which the nodes that arrive early carry an "unfair" share of the connections in the network. Another way to quantify the imbalance is to observe that, in a run of this random process, we expect many nodes to end up with no incoming links.

Give a formula for the expected number of nodes with no incoming links in a network grown randomly according to this model.

Suggested solution

(a) For $j, i \in [n]$ let E_{ji} be the event that node v_j has an incoming link from v_i and let X_{ij} be the indicator random variable for this event. Let $X_j = \sum_{i=1}^{n} X_{ji}$. Then X_j is a random variable whose value is the number of incoming links to node v_j.

For $i \leq j$, $P[E_{ji}] = 0$. For $i > j$, we have

$$P[E_{ji}] = \frac{1}{i-1}.$$

By linearity of expectation,

$$\begin{aligned} E[X_j] &= \sum_{i=1}^{n} E[X_{ji}] \\ &= \sum_{i=j+1}^{n} \frac{1}{i-1} \\ &= \sum_{i=j}^{n-1} \frac{1}{i} \\ &= H_{n-1} - H_{j-1} \\ &= \theta(\ln n) - \theta(\ln j) \\ &= \theta\left(\ln \frac{n}{j}\right). \end{aligned}$$

where $H_n = \sum_{i=1}^{n} \frac{1}{i}$, denotes the n-th harmonic number.

(b) Node v_j has no incoming arcs iff the event $\overline{E_{ji}}$ occurs for every $i \in [n]$. Let N_j be the indicator random variable for this event. Using that

$P[E_{ji}] = \frac{1}{i-1}$ for $i > j$ and $P[E_{ji}] = 1$ for $i \leq j$ and that the events are mutually independent we obtain

$$\begin{aligned} E[N_j] &= \prod_{i=j+1}^{n} \left(1 - \frac{1}{i-1}\right) \\ &= \prod_{i=j}^{n-1} \left(1 - \frac{1}{i}\right) \\ &= \left(\frac{j-1}{j}\right)\left(\frac{j}{j+1}\right)\cdots\left(\frac{n-2}{n-1}\right) \\ &= \frac{j-1}{n-1} \end{aligned}$$

Now, let $N = \sum_{j=1}^{n} N_j$. Then N is a random variable whose value is the number of nodes with no incoming links. By linearity of expectation

$$\begin{aligned} E[N] &= \sum_{j=1}^{n} E[N_j] \\ &= \sum_{j=1}^{n} \frac{j-1}{n-1} \\ &= \frac{1}{n-1} \sum_{j=1}^{n} j - 1 \\ &= \frac{1}{n-1} \frac{n(n-1)}{2} \\ &= \frac{n}{2}, \end{aligned}$$

so we expect half the nodes to have no incoming link.

Exercise from course website

Exercise

In this problem, we consider an undirected graph $G = (V, E)$to model the problem where there are small devices, each of which uses wireless communication to communicate with d of the other nearby devices. The vertices in V are the devices, and there are edges between vertices representing pairs of devices within communication range of each other. Thus, each vertex has exactly d neighbors. Some of the devices should be given an uplink transmitter so they can send data to the main station. We want to use as few uplink transmitters as possible while still getting all of the data from all of the devices. Thus, in our graph, we want a subset $S \subseteq V$ such that for all $v \in V$, either $v \in V$ or v has a neighbor u such that $u \in V$. Such a set S is called a *dominating set.* If the devices corresponding to vertices in S all get uplink transmitters, then all devices can either send data to the main station directly, or send data to a device that can send data directly.

Finding a minimum-sized dominating set is an NP-hard problem (this means that no sub-exponential algorithm is known), so one does not expect to find a polynomial-time deterministic algorithm to solve it.

In this problem, consider a randomized algorithm that chooses a set S of $k = \frac{cn\ln(n)}{d+1}$ vertices from V uniformly at random. The constant c will be discussed later. You should show through the following that S is a dominating set with high probability.

In the following, assume that the vertices in S are chosen one at a time and choosing the same vertex more than once is allowed. A vertex v is said to dominate a vertex u if they are the same vertex or u is a neighbor of v. A set S' dominates u if some vertex in S' dominates u.

1. What can you say about the minimum size a dominating set in G must have?

2. Let $D[v, t]$ be the event that the tth random vertex chosen dominates v. Find the probability of $D[v, t]$.

3. Let D_V be the event that S dominates v. What is the probability of the complement of this event, $\overline{D_v}$?

4. Show that $\left(\frac{1}{e}\right)^{c\ln(n)}$ is an upper bound on the probability of $\overline{D_v}$ and that $\left(\frac{1}{e}\right)^{c\ln(n)} = \frac{1}{n^c}$.

5. Let A be the event that S is a dominating set. Use the Union Bound to give an upper bound on the probability of $\overline{A}$.

6. Discuss what value c should have.

Suggested solution

1. Adding a vertex to S increases the number vertices dominated by S by at most $d+1$. Thus, the minimum size of a dominating set is $n/(d+1)$.

2. The tth vertex chosen dominates v if it is v or a neighbor of v. There are $d+1$ vertices that dominate v and each vertex is chosen uniformly at random among all n vertices. Thus,
$$P[D[v,t]] = \frac{d+1}{n}.$$

3. We notice that $\overline{D_v}$ occurs iff $\overline{D[v,t]}$ for $t = 1, 2, \ldots, k$.[1] Thus,
$$P[\overline{D_v}] = \left(1 - \frac{d+1}{n}\right)^k$$

4. Rewriting the expression for $P[\overline{D_v}]$ from 4. we get
$$\begin{aligned} P[\overline{D_v}] &= \left(1 - \frac{d+1}{n}\right)^{\frac{cn\ln(n)}{d+1}} \\ &= \left(\left(1 - \frac{d+1}{n}\right)^{\frac{n}{d+1}}\right)^{c\ln(n)} \\ &\leq \left(\frac{1}{e}\right)^{c\ln(n)} \\ &= \frac{1}{e^{c\ln(n)}} \\ &= \frac{1}{n^c} \end{aligned}$$

   where we use the fact that $(1 - a^{-1})^a \leq 1/e$ for $a \geq 1$.

[1]Suppose we round k up to the nearest integer

5. $\overline{A}$ occurs iff some vertex is not covered by S. Thus, $\overline{A} = \cup_{v \in V} \overline{D_v}$ and the Union Bound gives

$$P[\overline{A}] \leq \sum_{v \in V} P[\overline{D_v}] \leq n \frac{1}{n^c} = \frac{1}{n^{c-1}}.$$

6. If we want a non-zero probability of S being a dominating (and we want to use the given bound to argue that this is the case) we must pick $c > 1$. This gives $P[\overline{A}] < 1$ and consequently $P[A] > 0$. One might wish to pick c larger in order to increase the likelihood of success at the expense of solution quality.