Randomness and Computation 18/19
Week 6 tutorial sheet (12-1pm, Tues 26th, Wed 27th February)

1. Recall the Coupon Collector problem where we repeatedly draw random footballer cards uniformly at random from a sample pool of of n footballers, in the "with replacement" setting. We have previously used our knowledge of geometric r.v.s to show that the expected number of purchases to acquire all cards is $n \cdot H(n) \sim n \cdot \ln(n)$. Then we analysed the variance of the process etc etc.

   We now show how to obtain similar results without considering the geometric distribution, and by applying Chernoff bounds. We will first analyse the number of trials to get an specific card, contrary to the set-up of our original analysis.

   (a) Suppose we consider a specific footballer card of interest, and for $j = 1, \ldots,$ define the Bernoulli variable Z_j (with probability n^{-1}) to be 1 if we draw that footballer on the j-th purchase, 0 otherwise. Let $Z = \sum_{j=1}^{m} Z_j$ be the number of time we obtain that footballer over m purchases. Clearly $E[Z] = m/n$.

   Use a one-sided Chernoff bound to show that if we have a number of samples slightly bigger than $3n \cdot \ln(n)$, more precisely $m \geq 3n(\ln(n) + \ln(k))$, this is enough to have $\Pr[Z < 1] \leq n^{-1}k^{-1}$.

   We will approach this with Chernoff bounds, our goal being to bound $\Pr[Z < \frac{n}{m}E[Z]]$ (as $\frac{n}{m}E[Z] = 1$). Using Theorem 4.5 from the book, we know that setting $\delta = (1 - \frac{n}{m})$, we have

$$\Pr[Z \leq 1] \;\leq\; \Pr[Z \leq (1-\delta)E[Z]] \;\leq\; e^{-\frac{m}{n}(1-\frac{n}{m})^2/2}$$

   We want to have $e^{-\frac{m}{n}(1-\frac{n}{m})^2/2}$ less than $n^{-1}k^{-1}$, which is the case if and only if

$$-\tfrac{m}{2n}(\tfrac{m-n}{m})^2 \leq -\ln(n) - \ln(k),$$

   which is true if and only if

$$\tfrac{m}{2n}(\tfrac{m-n}{m})^2 \geq \ln(n) + \ln(k),$$

   which is true if and only if

$$\tfrac{(m-n)^2}{m} \geq 2n(\ln(n) + \ln(k)),$$

   which is true if and only if

$$m + \tfrac{n^2-2nm}{m} \geq 2n(\ln(n) + \ln(k)).$$

   It will certainly suffice to show that

$$m - \tfrac{2nm}{m} \;=\; m - 2n \geq 2n(\ln(n) + \ln(k)).$$

   (ignoring the help from the positive $\frac{n^2}{m}$ term). We are now asking whether $m \geq 2n(\ln(n) + \ln(k)) + 2n$. Well, we have $m \geq 3n(\ln(n) + \ln(k))$ by definition and we know

that for any $n \geq 8$, we will have $\ln(n) > 2$, meaning that the third $n(\ln(n)+\ln(k))$ term of m (after $2n(\ln(n) + \ln(k))$ has been cancelled on both sides) will be at least $2n$ for every $n \geq 8$, as required.

alternative solution: Jonathan (from my Wednesday tutorial) had an alternative way of approaching this wrt to the original number of samples $m \geq n(\ln(n) + \ln(k))$ that I initially wrote on the sheet. He was able to use the complex form of the Chernoff bound from Thm 4.5 (the first bound) together with some properties of limits to (almost) show the $n^{-1}k^{-1}$ result for this lower value of m. Recall that 1. of Thm 4.5 states that for a r.v. X which is a sum of Poisson variables and which has $E[X] = \mu$, that for any $0 < \delta < 1$, we have

$$\Pr[X \leq (1-\delta)\mu] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu}.$$

For us, we have Z with expected value m/n (as n^{-1} is the probability of getting the card on a single trial) and our concern is that we might fail to succeed on each of the m trails ... and end up with 0 of the cards. Jonathan notes that we don't just need to set $\delta = (1 - \frac{n}{m})$ (which has the effect of making $(1-\delta)\mu$ equal 1, but in fact, as our concern is having $Z = 0$, we might consider any value of δ between $1 - \frac{n}{m}$ and 1. His idea is to actually consider the series of upper bounds as $\delta \to 1$:

$$\lim_{\delta \to 1} \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu} = \lim_{\delta \to 1} \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{m/n}.$$

Then he uses the fact that taking the limit of a fractional form, that if both numerator and denominator have definite limits, the overall limit equals the fraction of the numerator limit over the denominator limit. Note that $\frac{m}{n}$ is $\ln(n) + \ln(k)$ for our definition, and multiplying in, we get

$$\left(\frac{\lim_{\delta \to 1} e^{-\delta(\ln(n)+\ln(k))}}{\lim_{\delta \to 1}(1-\delta)^{(1-\delta)(\ln(n)+\ln(k))}}\right).$$

For the numerator, we can simplify to $\lim_{\delta \to 1}(n^{-1}k^{-1})^{\delta}$ which is $n^{-1}k^{-1}$. For the denominator, we can use the rules of limits to see it tends to 0^0, which has limit 1.
So we get a limit to the Chernoff upper bound of $n^{-1}k^{-1}$.

I like this second solution a lot. Thanks Jonathan!

(b) Now show that if we have when each individual card has probability $1 - n^{-1}k^{-1}$ (for some value $k > 2$, reasonable size n) of being generated over m random purchases, that the probability of getting all cards is at least $1 - k^{-1}$.

For this part of the analysis, we are dealing with n different random Z variables (one for each different footballer). With this new analysis, however, these high-level variables are not independent. So we will use the Union Bound. The "bad event" for a particular Z is failing to get 1 card with that footballer. We have shown (for the m in part (a)) that the probability of that bad event is $\leq n^{-1}k^{-1}$. Then, the probability of failing to get *some* footballer is at most $nn^{-1}k^{-1}$, which is k^{-1}.

(c) Compare and contrast this analyses with the analysis in the slides for lectures (using geometric random variables and Chebyshev).

I mean difference in the approach, and also the bound of samples to achieve equivalent probability guarantees (Chernoff approach is better). Just use $3n(\ln(n) + \ln(k))$ in redoing the Chebyshev calculations from the slides and see what you get.

2. (Rough solution to second Q) Imagine that we draw the n inputs to BucketSort independently and uniformly at random from $\{0,1\}^k$. Hence ...

   The first-m-bits of the inputs are independently uniform from $\{0,1\}^m$. Each a_i has probability $\frac{1}{2^m}$ of entering any bucket. Bucket Sort can be seen as a "balls-in-bins" experiment.

   Running time is $\Theta(n)$ for the linear scan of 1. The *expected* running time for 2.-3. will be $E[\sum_{b\in\{0,1\}^m} c \cdot (X_b^2)]$, where X_b is the number of inputs landing in bucket b, and $c > 0$ is the fixed constant of the $O(n^2)$ algorithm.

   We want to evaluate $E[\sum_{b\in\{0,1\}^m} c \cdot (X_b^2)] = \sum_{b\in\{0,1\}^m} c \cdot E[X_b^2]$.

   We are now going to use an unexpected "trick" where we exploit the "second moment" of Binomial random variables to bound the $E[X_b^2]$.

   Realise each X_b is a binomial random variable $B[n, \frac{1}{2^m}]$ with

$$E[X_b^2] = n(n-1)2^{-2m} + n2^{-m}.$$

   Multiplying by 2^m (for each $b \in \{0,1\}^m$), and by c, this gives expected time for 2.-3. at most

$$c \cdot \left(n^2 2^{-m} + n\right).$$

   Choose m carefully to satisfy $m \geq \lg(n)$ and we see that this ensures the expected number of steps for 2.-3. is at most $2 \cdot c \cdot n$.

3. Consider a function $F : \{0,1,\ldots,n-1\} \to \{0,1,\ldots,m-1\}$ and suppose we know that for $0 \leq x, y \leq n-1$, $F((x+y) \bmod n) = (F(x) + F(y)) \bmod m$. The only way we know to evaluate $F(\cdot)$ is to examine the values in an array where the $F(\cdot)$ values have been stored (with entry i holding the value of $F(i)$). Unfortunately, a system failure has corrupted up to a $1/5$-fraction of the entries of the array, so we no longer have reliable values in all positions. We now design a simple randomized algorithm that, given an input $z \in \{0,\ldots,n-1\}$, outputs a value that equals $F(z)$ with probability at least $1/2$.

   note: we are not allowed to assume that the corruptions have taken place uniformly at random; it is possible that they could have ocurred along a contiguous sub-block of the array, or maybe just at even positions of the array. So we must take a worst-case approach to the analysis.

   solution: Our algorithm relies on the additive (modulo m) property and a simple sampling rule: regardless of the value z input, we will randomly draw a value y from $\{0,1,\ldots,n-1\}$ uniformly at random. After having drawn y, we will define $x =_{\text{def}} (z-y) \bmod n$. Note that

this ensures $z = (x + y) \bmod n$. We will then "lookup" the values of $F(x)$ and $F(y)$ from the table, add $F(x)$ and $F(y)$ together, and take the remainder with m as the final answer for $F(z)$.

The probability that this value returned is the true value $F(z)$ is exactly the probability that *both* $F(x)$ and $F(y)$ were *not corrupted*. Let's consider the fact that at most $\frac{n}{5}$ of the n entries of F have been corrupted; this means that in considering the n different (x, y) *pairs*, at most $\frac{2n}{5}$ of these might be somehow corrupted (either x corrupted, or y corrupted, or both). So $\frac{3n}{5}$ of the n (x, y) pairs for z might be corrupted, and given that we drew y randomly from all possible n values, this means the probability we have an uncorrupted pair is $\geq \frac{3}{5} > \frac{1}{2}$. This result is independent of the particular z which was chosen.

Now suppose we are allowed to repeat the initial algorithm times before we return a result. When it says "repeat the algorithm" this implies a resampling of the y (and hence a new (x, y) pair) each time. The approach should be to view the three results returned for $F(z)$ and if ≥ 2 of those are the same, return that max value; otherwise, if all are different, randomly return any of them.

What is the probability of a correct answer? It is certainly at least as high as the probability that *at most one answer is corrupted*, which is $\frac{3}{5}^3 + 3\frac{2}{5}\frac{3}{5}^2$. If exactly two of the results are corrupted, which happens with probability $3\frac{3}{5}\frac{2}{5}^2$, there is still probability at most $\frac{1}{3}$ of the right answer being returned - unfortunately, this is not guaranteed, as we may be tricked if the two corrupted values match. So we can only say that the total probability of the right answer returned will be at least

$$\frac{9}{25}\frac{1}{5}(3 + 6) \;=\; \frac{81}{125},$$

which is almost .65, an improvement on the original result.

4. Our variation on the "Max-Cut" (or $\frac{|E|}{2}$-cut) algorithm will chose a random *balanced* cut of the graph (instead of a random cut of variable size, as we did in lecture 4). Specifically, we choose $S \subset V$ to be any random subset of size $\lfloor \frac{|V|}{2} \rfloor$ and take the cut $(S, V \setminus S)$ to be our cut. Let $n = |V|$.

   We now analyse the expected size of this cut, in two cases.

   n even: In this case, S will be any subset of size exactly $n/2$, and the same for $V \setminus S$. Let $e = (u, v) \in E$. Then the probability that u and v are on different sides of the cut is exactly

$$\frac{2\binom{n-2}{n/2-1}}{\binom{n}{n/2}} = \frac{2(n-2)!}{(n/2-1)!(n/2-1)!}\frac{(n/2)!(n/2)!}{n!} = \frac{2(n/2)^2}{n(n-1)} = \frac{n}{2n-2} > \frac{n}{2n-1}.$$

   n odd: In this case, S will be any subset of size exactly $(n-1)/2$, with $V \setminus S$ of size $(n+1)/2$. Then for $e = (u, v) \in E$, the probability that u and v are on different sides of the cut is exactly

$$\frac{2\binom{n-2}{(n-3)/2}}{\binom{n}{(n-1)/2}} = \frac{2(n-2)!}{\frac{n-3}{2}!\frac{n-1}{2}!} \cdot \frac{\frac{n-1}{2}!\frac{n+1}{2}!}{n!} = \frac{2}{n(n-1)}\frac{n+1}{2}\frac{n-1}{2} = \frac{n+1}{2n}$$

which is also greater than $\frac{n}{2n-1}$, as required.

Now we consider the set E of all edges, and note that the number of edges in the random cut $(S, V \setminus S)$ is $E[X] = E[\sum_{\{u,v\} \in E} X_{u,v}]$, where $X_{u,v}$ is the indicator variable that is 1 when u and v end up on opposite sides of the cut. Applying linearity of expectation this is $\sum_{\{u,v\} \in E} E[X_{u,v}]$, which is $|E|E[X_{u,v}]$ for any $\{u, v\}$ (our analysis above did not depend on the particular edge we were interested in). This is at least $|E|\frac{n}{2n-1}$, as required.

5. The "Max Cut" approximation algorithm developed in question 4 is simple: we randomly choose a subset $S \subset V$ of size $\lceil \frac{n}{2} \rceil$ to be the "left side" of the cut (with $V \setminus S$ being the "right side"). We were able to show that this "balanced cut" approach gives an expected cut satisfying $E[|C_S|] \geq |E| \cdot \frac{|V|}{2|V|-1}$ in either the odd or the even case.

   Note the exact expectation for the balanced cut differs depending on whether $n = |V|$ is even or odd, with the factor on $|E|$ being $\frac{n}{2(n-1)}$ for the even case and $\frac{n^2-1}{2n(n-1)}$ in the odd case. The particular value is not that important, what is important is that $E[C_S]$ is *exactly* equal to $|E|$ multiplied by this known factor (and of course that the factor is at least $\frac{n}{2n-1}$). We will need to use the exact-ness in our analysis below.

   We now show how to derandomize this improved algorithm to obtain a deterministic algorithm which *always* returns a cut of size *at least* the original expected value; ie, $|E|\frac{|V|}{2|V|-1}$.

   Our derandomisation is different to the derandomisation of the standard Max-Cut approximation of Lecture 4 because here we will need to ensure that the "sizes" of the partial Cut stay balanced throughout (exactly the same size, or difference of 1 vertex). A single step of our derandomisation will consider a "side" of the cut and assign a vertex to add to that side, alternating between sides at each step. At any given stage of the derandomisation we will have the following partial solution/parameters - $L \subset V, R \subset V \setminus L$ such that $|L| - 1 \leq |R| \leq |L|$. We write the expected size of a random *balanced extension* (conditional on fixed L, R) as $E[C_S \mid L, R]$ and we will ensure that this value is always at least $|E|\frac{|V|}{2|V|-1}$ at any intermediate stage of our derandomisation.

   We start by assigning an arbitrary vertex $v*$ to the left side of the cut - assume there is a "best max cut" which achieves the value $|E|\frac{|V|}{2|V|-1}$ with this vertex on the left-hand side (we can always remove this assumption in a similar way to the derandomization step below).

   We will use k to represent the size of L at the current step. When $|R| = k$ also, our next step will be to find a vertex to add to L, and when $|R| = (k-1)$ our next step will be to find a vertex to add to R (note this is independent of whether $n = |V|$ for our original graph was even or odd). By doing this, we ensure that our partial solution is balanced and that it will always be extendable to an overall balanced cut. We will "choose the next vertex" (for whichever "side" we are working with) that maintains expected cut size.

   Now suppose we at an intermediate stage where L, R are the current (almost-balanced) partial assignments. First assume $|R| = k$, hence our next step is to add a vertex to L. For every possible $w \in V \setminus (L \cup R)$, we will consider $L \cup \{w\}, R$, and must calculate the value of $E[C_S \mid L \cup \{w\}, R]$. There are a large number of different balanced cuts that extend $L \cup \{w\}, R)$ and

these various balanced cuts will overlap with the balanced cuts that extend $(L \cup \{w'\}, R)$ for different w'. This is different to the derandomisation of the standard Max Cut algorithm where we compare two disjoint events (w getting added to left or to the right). However every $w \in V \setminus (L \cup R)$ lies on the left side for exactly the same number of balanced extensions of (L, R) as for any other $w' \in V \setminus (L \cup R)$, so each $E[C_S \mid L \cup \{w\}, R]$ ($w \in V \setminus (L \cup R)$) has a common factor in the expansion of $E[C_S \mid L, R]$, and we are guaranteed that there is one $E[C_S \mid L \cup \{w\}, R]$ which is at least as big as $E[C_S \mid L, R]$. We will find that w by computing the $E[C_S \mid L \cup \{w\}, R]$ values and comparing them, add that w to the left, then start again to choose a vertex for the right-side.

First calculate $m_{L,R} = |E \cap (L \times R)|$, the number of edges definitely belonging to the cut, conditional on (L, R). Note this does not depend on w and can be computed in $O(n^2)$ time.

The value $m_{L,R}$ must be added in the calculation of every $E[C_S \mid L \cup \{w\}, R]$. Also for every specific $w \in V \setminus (L \cup R)$, we must add $m_{w,R} = |Nbd(w) \cap R|$, the number of edges which enter the cut after adding w to the left.

Consider each $w \in V \setminus (L \cup R)$ in turn. Let $\widehat{V} = V \setminus (L \cup R \cup \{w\})$, $\widehat{n} = |\widehat{V}|$. Let $\widehat{E} = \{\{u, v\} : u, v \in \widehat{V}\}$ and note that these are not the only edges affecting the value of $E[C_S \mid L \cup \{w\}, R]$, also relevant are the edges between $L \cup R \cup \{w\}$ and $\widehat{V}$.

The edges of $\widehat{E}$ are the induced edges of the graph on $\widehat{V}$ and since the extension of $(L \cup \{w\}, R)$ will define a balanced cut on $\widehat{V}$ (this time with the possibly larger side on the right) the contribution to the expectation from this will be exactly

$$|\widehat{E}| \cdot \mathrm{fac}(\widehat{n})$$

where $\mathrm{fac}(\widehat{n})$ is either $\frac{\widehat{n}}{2(\widehat{n}-1)}$ or $\frac{\widehat{n}^2-1}{\widehat{n}(\widehat{n}-1)}$ depending on whether $\widehat{n}$ is odd or even. Note the constant in front of $|\widehat{E}|$ will be the same for any particular w (though $|\widehat{E}|$ might not) and it creates an exact equality for the expectation on this group of nodes/edges, not a lower bound.

Consider the edges between $\widehat{V}$ and L, and between $\widehat{V}$ and R. Define $m_L(\widehat{V}) = \sum_{v \in \widehat{V}} |Nbd(v) \cap R|$, $m_R(\widehat{V}) = \sum_{v \in \widehat{V}} |Nbd(v) \cap L|$. We know that after adding w to L, the left side will have space for $\lfloor \frac{\widehat{n}}{2} \rfloor$ more vertices and the right-side for $\lceil \frac{\widehat{n}}{2} \rceil$. So the contribution to the expectation of these un-placed vertices wrt the existing L, R is

$$\frac{\lfloor \widehat{n}/2 \rfloor}{\widehat{n}} \cdot m_L + \frac{\lceil \widehat{n}/2 \rceil}{\widehat{n}} \cdot m_R.$$

The overall value of $E[C_S \mid L \cup \{w\}, R]$ is then exactly

$$m_{L,R} + |Nbd(w) \cap R| + \mathrm{fac}(\widehat{n}) \cdot |\widehat{E}| + \frac{\lfloor \widehat{n}/2 \rfloor}{\widehat{n}} \cdot m_L + \frac{\lceil \widehat{n}/2 \rceil}{\widehat{n}} \cdot m_R.$$

Note that the task of computing *all* these time will take $O(1)$ (time to update $m_{L,R}$ after the previous vertex added), $|Nbd(w) \cap R|$ for each of the $\widehat{n}+1$ vertices, hence $O(n \cdot n)$ to do this for all w, and $O(n^2)$ time to compute m_L and m_R (first compute the values without any w

deleted in $O(n^2)$, and then adjust the values by subtracting the $|\mathsf{Nbd}(w) \cap R|$ or $|\mathsf{Nbd}(w) \cap L|$ each time ... $O(n)$ for each w). Similarly $|\widehat{E}|$ can be computed for all w by first computing $|E \cap (V \setminus L) \times (V \setminus R)|$ in $O(n^2)$ and then adjusting to subtract the w-adjacent edges each time (overall $O(n \cdot n)$). We need to compute $\mathrm{fac}(\widehat{n})$ once, $\Theta(1)$. Overall we can compute the optimal w, and update (L, R) in $O(n^2)$ time. We will need to do this until every vertex has been allocated, so $O(n^3)$ to build the cut which achieves the expectation.

Note that although I explain this in terms of adding to left/right, the fact that I allow for odd/even n means that we don't know whether the extension on $(L \cup \{w\}, R)$ will need to add equal number of vertices to left/right (if the original n was odd), or one more to R (if the original n was even). Hence we have covered both cases in our analysis and the only change for "adding to R" is that we swap the role of L and R. We must always alternate the choice of a vertex for L versus R to ensure we stay inside our target space of balanced cuts.

Mary Cryan, 26th Feb