CS 174: Combinatorics and Discrete Probability Fall 2012

Homework 2

Problem 1. *(Exercise 1.23 from MU)* There may be several different min-cut sets in a graph. Using the analysis of the randomized min-cut algorithm, argue that there can be at most $n(n-1)/2$ distinct min-cut sets.

Solution: Suppose there are c distinct min-cut sets. Let E_i be the event that the algorithm described in thm 1.8 outputs the i^{th} min-cut. Thm 1.8 says this happens with probability $\geq \frac{2}{n(n-1)}$. The E_i are disjoint, since the algorithm outputs at most one min-cut, so

$$\begin{aligned} \sum_{i=1}^{c} \mathbb{P}[E_i] &\leq 1 \\ \Rightarrow \frac{2c}{n(n-1)} &\leq 1 \\ \Rightarrow c &\leq \frac{n(n-1)}{2} \end{aligned}$$

Problem 2. The goal of this exercise is to compute uniformly random sequences of length k of the integers $\{1, \ldots, n\}$ *without replacement*. If $k = n$, the sample is a permutation of $1, \ldots, n$.

(a) Assuming the existence of a constant-time generator that generates uniformly random *real* numbers in the interval $[0, 1]$, derive a $O(n \log(n))$-time algorithm for computing a random permutation of $1, \ldots, n$. (Hint: Think sorting.)

Solution: augment the integers $1, \ldots, n$ with random keys, forming the pairs $(r_1, 1), \ldots, (r_n, n)$ where each r_i is a uniform random number from $[0, 1]$. Sort by these keys using an $O(n \log(n))$ algorithm (for example, heapsort or mergesort). The result will be some ordering of the pairs $(r_1, i_1), \ldots, (r_n, i_n)$ – discard the keys (first coordinate of each pair) to get the desired permutation.

Since the r_is are independent and identically distributed from a continuous distribution (0 probability of collisions), by symmetry any ordering is equally likely. So the resulting permutation is distributed uniformly over all permutations with probability $1/n!$.

(b) Assuming the existence of a constant-time generator that generates uniformly random *integers* in the range $\{1, \ldots, m\}$ for any m, derive an $O(n)$-time algorithm that generates a random permutation of $1, \ldots, n$. (Hint: Start with a sorting algorithm that makes $O(n)$ data moves.)

Solution: use selection sort, but instead of scanning for the largest remaining element, pick one randomly. So starting with an array x containing $1, \ldots, n$ in order:

```
for i in {1, ..., n-1}
  pick j uniformly from {1, ..., n-i+1}
```

```
  j = j + i - 1 // now uniform from {i, ..., n}
  swap x[i] with x[j]
output x
```

It is easy to see that the above algorithm can produce any permutation. To produce the permutation $1 \to i_1, 2 \to i_2, \ldots, n \to i_n$ simply select element i_j at step j. Also, the above algorithm has at most $n\dot{(}n1) \cdot \ldots \cdot 2 \cdot 1 = n!$ different outputs that each occur with equal probability because in the i^{th} iteration we chose from n i + 1 numbers uniformly at random. Therefore, the above algorithm must output each permutation uniformly at random.

Also, the algorithm runs in time $O(n)$ because to produce the initial state takes time $O(n)$, and there are $n1$ iterations, and reading off the answer takes time $O(n)$. Finally, each iteration can be run in constant time because an array permits random access in constant time and with random access swapping can easily be done with 3 operations.

(c) *(Optional, not graded)* Assuming the same random number generator as in (b) above, derive an $O(k)$ expected-time algorithm that generates a random sequence of length k from $1, \ldots, n$ without replacement.

Solution: if $k > n/2$, then $O(k) = O(n)$ and we can use the solution above, but stop running after k elements have been selected.

If $k \leq n/2$, then create an empty hashtable set S and an empty output list L.

```
for i in {1, ..., k}
  while true
    pick j randomly from {1, ..., n}
    if j not in S
      add j to S
      append j to L
      break
output L
```

Since each random j is already in the hashtable with probability at most $1/2$, so the expected number of tries to find an unused j is ≤ 2, so with k steps the expected running time is $O(k)$.

Problem 3. *(Exercise 2.2 from MU)* A monkey types on a 26-letter keyboard that has lowercase letters only. Each letter is chosen independently and uniformly at random from the alphabet. If the monkey types 1,000,000 letters, what is the expected number of times the sequence "proof" appears?

Solution: Let X_i be 1 if the i^{th} – $(i+4)^{th}$ letters spell out "proof" and 0 otherwise. $X_i = 1$ occurs with probability $1/26^5$. So the total expected occurences of "proof" are

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^{10^6-4} X_i\right] &= \sum_{i=1}^{10^6-4} \mathbb{E}\left[X_i\right] \\
&= \sum_{i=1}^{10^6-4} P[X_i = 1] \\
&= \frac{10^6 - 4}{26^5} \\
&= \frac{999996}{11881376} \approx 0.084165
\end{aligned}$$

Problem 4. *(Exercise 2.18 from MU)* The following approach is often called *reservoir sampling*. Suppose we have a sequence of items passing by one at a time. We want to maintain a sample of one item with the property that it is uniformly distributed over all the items that we have seen at each step. Moreover, we want to accomplish this without knowing the total number of items in advance or storing all of the items that we see.

Consider the following algorithm, which stores just one item in memory at all times. When the first item appears, it is stored in the memory. When the k^{th} item appears, it replaces the item in memory with probability $1/k$. Explain why this algorithm solves the problem.

Solution: we will prove this by induction. Let $b_1, b_2, ...b_n$ be the first n items observed. Let M_n be a random variable that takes the value of the item in memory after the n^{th} observation. We need to show that $\Pr[M_n = b_i] = 1/n$ for all $1 \leq i \leq n$.

The base case is when $n = 1$, which is trivially true since $M_n = b_1$ with probability 1. Assume that at after n observations, $\Pr[M_n = b_i] = 1/n$ for all $1 \leq i \leq n$. Now we prove that this property holds for time $n + 1$. After $n + 1$ observations, we set $M_{n+1} = b_{n+1}$ with probability $1/(n + 1)$. Therefore, $\Pr[M_{n+1} = b_{n+1}] = 1/(n + 1)$. For $1 \leq i \leq n$,

$$\begin{aligned}
\Pr[M_{n+1} = b_i] &= \Pr[\text{no swap after } n \text{ observations and } M_n = b_i] \\
&= \Pr[\text{no swap after } n \text{ observations}] \Pr[M_n = b_i] \\
&= \frac{n}{n+1}\frac{1}{n} \\
&= \frac{1}{n+1}
\end{aligned}$$

Problem 5. *(Exercise 2.22 from MU)* Let $a_1, a_2, \ldots, a_n$ be a list of n distinct numbers. We say that a_i and a_j are *inverted* if $i < j$ but $a_i > a_j$. The *Bubblesort* sorting algorithms swaps pairwise adjacent inverted numbers in the list until there are no more inversions, so the list is in sorted order. Suppose that the input to Bubblesort is a random permutation, equally likely to be any of the $n!$ permutations of n distinct numbers. Determine the expected number of inversions that need to be corrected by Bubblesort.

Solution: let $X_{ij} = \mathbb{I}(a_i > a_j)$ and consider the number

$$S = \sum_{i<j} X_{ij}$$

since any correctional inversion decreases S by exactly 1, we know that S inversions are required in total and their order doesn't matter. So we need only find the expected value of S.

$$\begin{aligned}
\mathbb{E}[S] &= \mathbb{E}\left[\sum_{i<j} X_{ij}\right] \\
&= \sum_{i<j} \mathbb{E}[X_{ij}] \\
&= \sum_{i<j} \mathbb{P}[a_i > a_j] \\
&= \sum_{i<j} \frac{1}{2} \quad \text{by symmetry of random permutations} \\
&= \frac{1}{2}\sum_{i=1}^{n}(n-i) \\
&= \frac{1}{2}\sum_{i=0}^{n-1} i \\
&= \frac{(n-1)n}{4}
\end{aligned}$$

Problem 6. *(Exercise 2.32 from MU)* You need a new staff assistant, and you have n people to interview. You want to hire the best candidate for the position. When you interview a candidate, you can give them a score, with the highest score being the best and no ties being possible. You interview the candidates one by one. Because of your company's hiring practices, after you interview the k^{th} candidate, you either offer the candidate the job before the next interview or you forever lose the chance to hire that candidate. We suppose the candidates are interviewed in a random order, chosen uniformly at random from all $n!$ possible orderings.

We consider the following strategy. First, interview m candidates but reject them all; these candidates give you an idea of how strong the field is. After the m^{th} candidate, hire the first candidate you interview who is better than all of the previous candidates you have interviewed.

(a) Let E be the event that we hire the best assistant, and let E_i be the event that i^{th} candidate is the best and we hire him. Determine $\Pr(E_i)$, and show that

$$\Pr(E) = \frac{m}{n}\sum_{j=m+1}^{n}\frac{1}{j-1}.$$

Solution: Notice that the E_i are disjoint events, therefore $\Pr[E] = \sum_{i=1}^{n}\Pr[E_i]$. For $i \leq m$, $\Pr[E_i] = 0$, since none of the first m candidates are selected. Now, we see that for $i > m$ two independents events make up E_i.

$$\begin{aligned}
\Pr[E_i] &= \Pr[i\text{th candidate is the best}] \cdot \Pr[\text{the } i\text{th candidate is chosen}] \\
&= \frac{1}{n} \cdot \Pr[\text{best of the } (i-1) \text{ candidates is in the first } m \text{ candidates}] \\
&= \frac{1}{n} \cdot \frac{m}{i-1}
\end{aligned}$$

Now, putting this all together, we get

$$\Pr[E] = \sum_{j=m+1}^{n} \Pr[E_i] = \frac{m}{n} \sum_{j=m+1}^{n} \frac{1}{j-1}$$

(b) Bound $\sum_{j=m+1}^{n}(1/(j-1))$ to obtain

$$\frac{m}{n}(\ln n - \ln m) \leq \Pr(E) \leq \frac{m}{n}(\ln(n-1) - \ln(m-1)).$$

Solution: Using Lemma 2.10 from the book, we get the solution

$$\Pr[E] \geq \frac{m}{n} \int_{m+1}^{n+1} \frac{1}{x-1} dx = \ln(x-1)|_{m+1}^{n+1} = \frac{m}{n} \left(\ln(n) - \ln(m)\right)$$

and

$$\Pr[E] \leq \frac{m}{n} \int_{m}^{n} \frac{1}{x-1} dx = \ln(x-1)|_{m}^{n} = \frac{m}{n} \left(\ln(n-1) - \ln(m-1)\right)$$

(c) Show that $m(\ln n - \ln m)/n$ is maximized when $m = n/e$, and explain why this means that $\Pr(E) \geq 1/e$ for this choice of m.

Solution: How should we find the best m? Since the bound from above is concave, we can take the derivative, set it equal to zero, and solve for m. This yields the m that maximizes $Pr[E]$. We have

$$\frac{d}{dm}\frac{m}{n}\left(\ln(n) - \ln(m)\right) = \frac{\ln(n)}{n} - \frac{\ln(m)}{n} + \frac{1}{n} = 0.$$

Then we get $\ln(m) = \ln(n) - 1$, which is

$$m = e^{\ln(n)-1} = e^{\ln(n)}e^{-1} = ne^{-1} = \frac{n}{e}.$$