CS 577 # Assignment 13 – Randomization Spring 2023

Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document.

Name: ____________________ Wisc id: ____________________

Randomization

1. *Kleinberg, Jon. Algorithm Design (p. 782, q. 1).*

   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 $\frac{2}{3}c^*$ edges. If you want, your algorithm can be randomized; in this case, the expected number of edges it satisfies should be at least $\frac{2}{3}c^*$.

Solution:

Assign each vertex a uniformly random color. Then for each edge (u, v), this edge is only not satisfied if the two vertex colors of u and v are the same, which has a $1/3$ probability of occurring. Thus the probability that any particular edge is satisfied is $2/3$, so in expectation $(2/3)|E|$ edges are satisfied. Since $c^* \leq |E|$, this satisfies at least $(2/3)c^*$ edges in expectation.

2. *Kleinberg, Jon. Algorithm Design (p. 787, q. 7).*

   In lecture, we designed an approximation algorithm to within a factor of 7/8 for the MAX 3-SAT Problem, where we assumed that each clause has terms associated with three different variables. In this problem, we will consider the analogous MAX SAT Problem: Given a set of clauses $C_1, \cdots, C_k$ over a set of variables $X = \{x_1, \cdots, x_n\}$, find a truth assignment satisfying as many of the clauses as possible. Each clause has at least one term in it, and all the variables in a single clause are distinct, but otherwise we do not make any assumptions on the length of the clauses: There may be clauses that have a lot of variables, and others may have just a single variable.

   (a) First consider the randomized approximation algorithm we used for MAX 3-SAT, setting each variable independently to true or false with probability 1/2 each. Show that in the MAX SAT, the expected number of clauses satisfied by this random assignment is at least $k/2$, that is, at least half of the clauses are satisfied in expectation.

   Solution:

   For a MAX SAT instance of n variables $x_1, \cdots, x_n$ and k clauses $C_1, \cdots, C_k$, we set each variable independently to true or false with probability 1/2. For a clause with m variables, there is only one assignment under which it is not satisfied (all terms are false), so the probability that it is satisfied is $1 - \left(\frac{1}{2}\right)^m$. Since $m \geq 1$, $1 - \left(\frac{1}{2}\right)^m \geq 1 - \frac{1}{2} = \frac{1}{2}$, so each clause has at least a 1/2 probability of being satisfied. Thus the expected number of clauses satisfied is at least $k/2$ by linearity of expectation.

   (b) Give an example to show that there are MAX SAT instances such that no assignment satisfies more than half of the clauses.

   Solution:

   For any $n \in \mathbb{N}$, we can create an instance with n variables and $2n$ clauses, where for each variable x_i, we add both the clause $\{x_i\}$ and the clause $\{\overline{x_i}\}$. Thus the final formula looks like

$$x_1 \wedge \overline{x_1} \wedge x_2 \wedge \overline{x_2} \wedge \cdots \wedge x_n \wedge \overline{x_n}$$

   Any assignment to this formula satisfies n clauses, exactly half of all the clauses.

(c) If we have a clause that consists only of a single term (e.g., a clause consisting just of x_1, or just of $\overline{x_2}$), then there is only a single way to satisfy it: We need to set the corresponding variable in the appropriate way. If we have two clauses such that one consists of just the term x_i, and the other consists of just the negated term $\overline{x_i}$, then this is a pretty direct contradiction. Assume that our instance has no such pair of "conflicting clauses"; that is, for no variable x_i do we have both a clause $C = \{x_i\}$ and a clause $C' = \{\overline{x_i}\}$. Modify the randomized procedure above to improve the approximation factor from 1/2 to at least 0.6. That is, change the algorithm so that the expected number of clauses satisfied by the process is at least $0.6k$.

Solution:

If x_i appears in a singleton clause, then instead of assigning x_i randomly, we bias the assignment so that it has probability $p \geq 1/2$ of satisfying the singleton clause. Then, for each clause, there are two cases:

1. The clause is a singleton. In this case, the probability of satisfying the clause is p.
2. The clause contains $m \geq 2$ variables. In this case, the probability of satisfying the clause is at least $1 - p^m$. This is because the only way to not satisfy the clause is if all m variables are false, which has probability at most p^m (when there is a negating singleton for each variable). We further loose the lower bound to be $1 - p^2$, since $p \leq 1$, $p^m \leq p^2$, thus $1 - p^m \geq 1 - p^2$.

Now, we want to choose $p \in (1/2, 1]$ to maximize $\min(p, 1 - p^2)$ in order to maximize the expected number of clauses satisfied. By setting $p = 1 - p^2$ we get $p = (\sqrt{5} - 1)/2 \approx 0.618$. Thus in expectation, $0.618k > 0.6k$ clauses are satisfied.

(d) Give a randomized polynomial-time algorithm for the general MAX SAT Problem, so that the expected number of clauses satisfied by the algorithm is at least a 0.6 fraction of the maximum possible. (Note that, by the example in part (a), there are instances where one cannot satisfy more than $k/2$ clauses; the point here is that we'd still like an efficient algorithm that, in expectation, can satisfy a 0.6 fraction of the maximum that can be satisfied by an optimal assignment.)

Solution:

We can use the same algorithm as in part (c), but now we need to check for conflicting clauses. We can do this in polynomial time by searching for contradictory pair of clauses $\{x_i\}$ and $\{\overline{x_i}\}$. For each such pair, we remove one of the clauses from the formula. If we end up removing m clauses (and $k-m$ clauses remaining), note that the maximum number of clauses that can be satisfied is at most $k-m$ since we can only satisfy one of the clauses in each pair. Calling the algorithm from part (c) on the reduced formula gives an approximation we want.

3. *Kleinberg, Jon. Algorithm Design (p. 789, q. 10).*

   Consider a very simple online auction system that works as follows. There are n *bidding agents*; agent i has a bid b_i, which is a positive natural number. We will assume that all bids b_i are distinct from one another. The bidding agents appear in an order chosen uniformly at random, each proposes its bid b_i in turn, and at all times the system maintains a variable b^* equal to the highest bid seen so far. (Initially b^* is set to 0.) What is the expected number of times that b^* is updated when this process is executed, as a function of the parameters in the problem?

Solution:

Bid i only updates b^* if $b_i > b_j$ for all $j < i$. Since the ordering is chosen uniformly at random, given the set of the first i bids (the bids without the order), all orders of these i bids are equally likely, and thus the probability that bid i updates b^* is $\frac{(i-1)!}{i!} = \frac{1}{i}$ (the denominator is the number of permutations for the i bids, and the numerator is the number of permutations with the last number being the largest). Then by linearity of expectations, we sum over i from 1 to n to obtain that the expected number of updates is $\sum_{i=1}^{n} \frac{1}{i}$ (the nth Harmonic number).

4. Recall that in an undirected and unweighted graph $G = (V, E)$, a cut is a partition of the vertices $(S, V\backslash S)$ (where $S \subseteq V$). The size of a cut is the number of edges which cross the cut (the number of edges (u, v) such that $u \in S$ and $v \in V\backslash S$). In the MAXCUT problem, we try to find the cut which has the largest value. (The decision version of MAXCUT is NP-complete, but we will not prove that here.) Give a randomized algorithm to find a cut which, in expectation, has value at least 1/2 of the maximum value cut.

Solution:

Assign each vertex to a side of the cut independently and uniformly at random. Then the probability that a given edge (u, v) is cut is $1/2$, so in expectation $(1/2)|E|$ edges are cut. The optimal cut can't have value larger than $|E|$, so this is at least $(1/2)$ times the optimal cut.

5. Implement an algorithm which, given a MAX 3-SAT instance, produces an assignment which satisfies at least 7/8 of the clauses, in either C, C++, C#, Java, or Python.

   The input will start with a positive integer n giving the number of variables, then a positive integer m giving the number of clauses, and then m lines describing each clause. The description of the clause will have three integers x y z, where $|x|$ encodes the variable number appearing in the first literal in the clause, the sign of x will be negative if and only if the literal is negated, and likewise for y and z to describe the two remaining literals in the clause. For example, $3\ -1\ -4$ corresponds to the clause $x_3 \wedge \overline{x_1} \wedge \overline{x_4}$. A sample input is the following:

```
10
5
-1 -2 -5
6 9 4
-9 -7 -8
2 -7 10
-1 3 -6
```

   Your program should output an assignment which satisfies at least $\lfloor 7/8 \rfloor m$ clauses. Return n numbers in a line, using a ± 1 encoding for each variable (the ith number should be 1 if x_i is assigned TRUE, and -1 otherwise). The maximum possible number of satisfied clauses is 3, so your assignment should satisfy at least $\lfloor \frac{7}{8} \times 3 \rfloor = 2$ clauses. One possible correct output to the sample input would be:

```
-1 1 1 1 1 1 -1 1 1 1
```