Advanced Algorithms: Homework 6

Ivani Patel

December 9, 2022

1. Google uses a Markov chain to mimic a person browsing webpages. Actually, markov chains have many other applications as well. However, a difficult part is, at first place, a markov chain is hard to obtain for practical application. Suppose that I treat a coffee maker as a markov chain (to simulate how one would use the machine). Notice that the modern coffee maker is quite complex and full of buttons to control almost everything. Suppose now that I think the coffee maker as a program and I want to test it. Can you give me a "random" test sequence (of buttons)? This is a difficult program. Open your mind.

   (a) After testing coffee maker, we have finite IO sequences.

   We define a finite graph G has a set of nodes Q and a set of directed edges E. Without loss of generally, we assume that every node is reachable from the initial node q_1. One IO sequence can be considered as a finite sequence of nodes in G.

   In this question, each output is produced by an input preceding it, so we can use Markov chain to represent the process of IO sequence. A finite state Markov chain χ is a discrete stochastic process $X_1, X_2, ...X_n, ...$ where the sample space for each random variable X_n is Q, and the conditional probability of χ need to satisfy:

   Prob$(X_n = x_n|X_{n-1} = x_{n-1}, ...X_1 = x_1)$ = Prob$(X_n = x_n|X_{n-1} = x_{n-1})$ for all $x_1, x_2, ..., x_n \in Q$. Together with the initial distribution Prob$(X_1 = x_1) = 1$, the probability of a particular sequence $\pi = x_1, x_2, ..., x_n$ for some n $\geq$ 1 is:

   $$Prob(\pi) = Prob(X_1 = x_1) \cdot Prob(X_2 = x_2|X_1 = x_1) \cdots Prob(X_n = x_n|X_{n-1} = x_{n-1})$$

   Hence, the Markov chain can also be represented in the form of probability transition matrix T = $[T_{i,j}]$ where each $T_{i,j}$ indicates the transition probability Prob$(X_{n+1} = q_j|X_n = q_i)$ from the node q_i to q_j. The Markov chain χ is called a represented Markov chain if, for each $T_{i,j}, T_{i,j} = 0$ where there is no edge from node q_i to q_j in G.

   In information theory entropy rate indicates the growth rate of the entropy of a stochastic process.The entropy rate of χ is defined as:

   $$\lambda_x = \limsup_{n\to\infty} \frac{1}{n} H(X_1, ..., X_n)$$

   where $H(X_1, ..., X_n)$ is the joint entropy of $X_1, ..., X_n$, defined as, according to Shannon,

   $$\sum_{x_1,...,x_n} p(x_1, ..., x_n) log \frac{1}{p(x_1, ..., x_n)}$$

   Notice that the upper limit always exists when the Maarkov chain χ is G represented. In this case, the maximal rate $\lambda*$ among all possible G-represented χ is shown that:

$$\lim_{n\to\infty} \frac{log(S_n)}{n} = \lambda*$$

where S_n is the number of paths in G with length n, and the rate $\lambda*$ is achievable by a G-represented χ. $\lambda*$ refers to the entropy rate of a most "random" Markov walk on the graph.

Let a positive small number ϵ ¿ 0. For a chosen test sequence and its Markov walk χ, if

$$|\lambda_x - \lambda *| < \epsilon$$

which means this test sequence is "similar" to the most "random" Markov walk on the graph. We call this test sequence is random.

2. Show that #3SAT is #P-complete.

   (a) We need to show that #3SAT is #P-complete, So we can apply Cook-Levin reduction on #P-problem. And for that we study some definitions and theorem.

   Definition 1: R is an NP-relation if there is a polynomial time algorithm A such that (x,y) $\in R \Leftrightarrow A(x,y) = 1$ and there is polynomial p such that (x,y) $\in R \Rightarrow |y| \le p(|x|)$.
   #R is the problem that given x, asks how many y satisfy (x,y) $\in R$.

   Definition 2: #P is the class of all problems of the form #R, where R is an NP-relation.

   Definition 3: We say there is a parsimonious reduction from #A to #B (written #A $\le_{par}$ #B) if there is a polynomial time transformation f such that for all x,

   $$|\{y, (x,y) \in A\}| = |\{z : (f(x), z) \in B\}|.$$

   Often this definition is a little too restrictive and we use the following definition instead.

   Definition 4: #A $\le$ #B if there is a polynomial time algorithm for #A given an oracle that solves #B.

   Then taking #CIRCUITSAT, it is the problem where given a circuit, we want to count the number of inputs that make the circuit output 1.

   Theorem: #CIRCUITSAT is #P-complete under parsimonious reductions.

   Proof: Let #R be in #P and A and p be as in the definition. Given x we want to construct a circuit C such that | { z : C(z) } | = | { y : |y| ≤ p(|z|),A(x,y) =1 } | . We construct $\hat{C_n}$ that on input x,y simulates A(x,y). From earlier arguments we know that this can be done with a circuit with size about the square of the running time of A. Thus, $\hat{C_n}$ will have size polynomial in the running time of A and so polynomial in x. Then let C(y) = $\hat{C_n}$(x,y).

   Finally, there is a theorem that shows #3SAT is #P-complete.

   Proof: We show that there is a parsimonious reduction from #CIRCUITSAT to #3SAT. That is given a circuit C, we construct a Boolean formula ϕ such that the number of satisfying assignments for ϕ is equal to the number of inputs for which C outputs 1. Suppose C has inputs $x_1, ..., x_n$ and gates $1, ..., m$ and ϕ has inputs $x_1, ..., x_n, g_1, ..., g_m$, where the g_i represent the output of gate i. Now each gate has two input variables and one output variable. Thus, a gate can be complete described by mimicking the output for each of the 4 possible inputs. Thus, each gate can be simulated using at most 4 clauses, In this way we have reduced C to a formula ϕ with n + m variables and 4m clauses. So, there is a parsimonious reduction from #CIRCUIT to #3SAT.

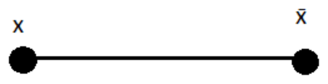
3. Consider a 3SAT Boolean formula F which has variables $x_1, \cdots, x_n$. Recall that the F is a conjunction of clause is a disjunctions of three literals. A Boolean variable a shows up in a clause if either a or $\bar{a}$ is in the clause. A set C of variables is a cover of F if for each clause in F there is a variable in C that shows up in the clause. Let k be the size of a cover C of a given F. Prove that the problem of 3SAT is fixed parameter tractable wrt k.

   (a) 3SAT Boolean Formula F with variables $x_1, x_2 \cdots, x_n$ is given to us. F is a conjunction of clauses ans each clause is a disjunction of three literals. For example it could be $(x_1 \vee \overline{x_3} \vee x_6) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_6})$

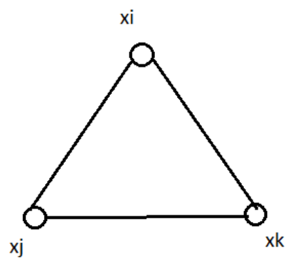
   Let U be the set of all variables in F such that = $\{x_1, x_2, ..x_n\}$
   We have to show that the 3SAT problem in this case which is whether there are values of $x_1, x_2, ..x_n$ that cause the given 3SAT Boolean formula F to be true.

   The 3SAT problem can be reduced to the Vertex Cover problem. That is, the 3SAT problem is equivalent to the problem that given an arbitrary graph G, we can find a subset(cover) of k nodes in the graph such that every edge in the graph touches one of the nodes in this subset.

   The reduction can be done in the following way:
   1. Every variable x in F can be replaced by 2 nodes in the graph.



   2. Each clause in F can be replaced by a clique of 3 nodes in the form of a triangle.

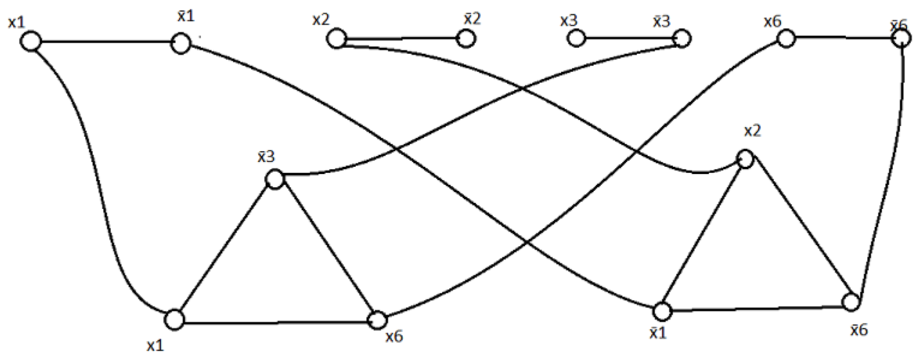


   The above clique would represent $x_i \vee x_j \vee x_k$
   For example for $F_1 = (x_1 \vee \overline{x_3} \vee x_6) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_6})$ we can reduce it to a graph G as shown in figure:

   If a Boolean formula F has n variables and c clauses, the graph G would have 2n + 3c vertices.

   Such a graph reduced from a Boolean formula F would have a vertex cover of size n + 2c.

   1. We saw from inspection that every satisfying True assignment gives a vertex cover and every vertex cover gives a satisfying True assignment.
   2. We see that vertex cover (the subset of k nodes in the graph such that every edge in the graph touches one of the nodes in the subset) is equivalent as the definition of the cover in the problem (for each clause there is a variable belonging to the cover(of size k) that shows up in clause).
   3. Thus we have shown 3SAT problem in this case can be reduced to a Vertex Cover problem with the same vertex cover size k.



4. We know that the vertex cover problem wrt fixed parameter k is tractable as shown in the lecture.
5. Thus the 3SAT is fixed parameter tractable wrt k since 3SAT ≤ vertex cover.

References:

1. ”Random Words in a (Weighted) Regular Language: A Free Energy Approach” by Cewei Cui and Zhe Dang, 2017.

2. ”Typical Paths of a Graph” by Cewei Cui, Zhe Dang and Thomas R. Fischer