Notes 1.

Rik Sarkar *Notes*

Optimizations. In class we discussed optimization problems and approximations. For a given function $f(\cdot)$, a *maximization* problem is to find the input x to f that achieves the maximum possible $f(x)$. Similarly, a *minimization* problems about finding the input to minimize f.

For the influence maximization problem, the input x too the form of subsets of V. This creates a computational challenge, since there are many possible subsets of V. We were looking for subsets of size k, but even then, there are $\binom{n}{k}$ possibilities, which is can be very large for a large k.

Q 1. *Suppose $k = n/2$. Show that the number of possible subsets of size k is at least $2^{\Omega(n)}$.*

Approximations. For a maximization problem, if f achieves its maximum value for input x^* and $f(x^*) = OPT$, then a c-approximation algorithm finds an x such that $f(x) \geq c \cdot OPT$. The value for c in this case will be a positive fraction less than 1.

For a minimization problem, if f achieves its minimum value for input x^* and $f(x^*) = OPT$, then a c-approximation algorithm finds an x such that $f(x) \leq c \cdot OPT$. The value of c will be greater than 1.

In both cases, c is called the approximation factor.

Q 2. *What is the range of possible values for the approximation factor of an approximation algorithm for the influence maximization problem?*

Q 3. *Suppose we want to find shortest paths. Is this is a maximization or minimization problem? What is the range of possible values for the approximation factor of an approximation algorithm for this problem?*

Two very useful inequalities:

$$\left(1 + \frac{1}{x}\right)^x \leq e$$

$$\left(1 - \frac{1}{x}\right)^x \leq \frac{1}{e}$$

We will make use of these many times in the course.

Q 4. *Suppose a tortoise is at distance n meters from its destination. It is getting tired with time, and in each hour, it covers $1/2$ of the remaining distance to the destination. How long does it take the tortoise to get to the destination?*

Q 5. *How long does it take it to get to within* 1 *meter of the destination?*

Q 6. *Now suppose instead the tortoise covers* $1/k$ *of the remaining distance to the destination in each hour. How long does it take it to get to within* 1 *meter?*

Q 7. *If the tortoise covers* $1/k$ *of the remaining distance to the destination in each hour, what fraction of the distance remains to be covered after* 1 *hr? What fraction remains to be covered after* k *hours?*

Q 8. *In the independent activation model, suppose node* u *has neighbors* $v_1, v_2, \ldots$ *and the corresponding edges have associated probabilities* $p_1, p_2, \ldots$. *Suppose the probability of* u *being activated is* $p(u)$. *If all neighbors* $v_1, v_2, \ldots$ *are active, can we say that* $p(u) \leq p_1 + p_2 + \ldots$*? What is the exact probability of* u *becoming active?*

Q 9. *Suppose for a node* x *in a network, an edge to each other* $n-1$ *nodes exists with probability* $\frac{\ln n}{n-1}$. *Show that the probability that* x *has no edge is* $\leq \frac{1}{n}$ *[hint: use* $(1-\frac{1}{x})^x$ *with* $x = \frac{\ln n}{n-1}$*.]*

Problem instances An *instance* of a problem is the problem asked for a particular dataset. For example, finding shortest path is an algorithmic problem, while finding the shortest path between a specific pari of nodes on a particular network is an instance of the shortest path problem.

NP hardness. (optional. not in exam). There are certain problems that are considered NP-hard, and belong to the class of problems called NP-hard. Let us refer to this set as NPH.

It can be shown that for any pair of problems $A, B \in NPH$, any given instance of A can be "reduced" to an instance of B in polynomial time. Meaning that given an instance of A, we can convert it to an instance of B in polynomial time, and then any solution of the instance of B can be converted back to a solution of the instance of A in polynomial time.

This also implies that if there is a polynomial time solution to B, then that can be used to get a polynomial time solution of A via the reduction. Thus, if any problem in NPH has a polynomial time solution, then every problem in NPH will have a polynomial time solution via the reduction. It is however generally believed that NP-hard problems cannot have polynomial time solutions. Though a proof of this fact is not known.

When we encounter a new problem C, the usual way to show that it is NP-hard is to take a known problem $B \in NPH$ and show that a polynomial time reduction exists from B to C. This implies that C is NP hard, and a polynomial time algorithm is unlikely. If a poly time algorithm is found for C, that would imply that all problems in NPH has poly time algorithms. Thus, once we have shown a problem to be NP-hard, finding ploy time algorithms for it is really unlikely.