DISCLAIMER: These notes are not necessarily an accurate representation of what I said during the class. They are mostly what I intend to say, and have not been carefully edited.

Last week we formalized ways of showing that a problem is hard, specifically the definition of $NP$, $NP$-completeness, and $NP$-hardness.

We now discuss ways of saying something useful about these hard problems. Specifically, we show that the answer produced by our algorithm is within a certain fraction of the optimum.

For a maximization problem, suppose now that we have an algorithm $A$ for our problem which, given an instance $I$, returns a solution with value $A(I)$. The approximation ratio of algorithm $A$ is defined to be

$$\max_I \frac{OPT(I)}{A(I)} .$$

1 2-Approximation for Maximum Cut

We start by giving a 2-approximation to $\text{MaxCut}$, which asks to find a cut that involve the most number of edges.

Here a simple greedy algorithm works: label the vertices $v_1 \ldots v_n$.

For each vertex $v_i$, put $v_i$ on a side different than the majority of its neighbors among $v_1 \ldots v_{i-1}$.

This ensures that we get at least half of $v_i$'s neighbors to smaller labeled vertices. Since each edge $uv$ has a larger labelled end point, we get that we get at least half the edges, aka

$$A(I) \geq \frac{m(I)}{2}$$

where $m(I)$ is the number of edges in $I$.

On the other hand, we can at most cut all the edges, so we get

$$OPT(I) \leq m(I) .$$

So putting these together gives for any input $I$

$$\frac{OPT(I)}{A(I)} \leq \frac{m}{m/2} \leq 2,$$

so this is a 2-approximation.
2 Approximate Matching and Vertex Cover

This is not restricted to NP-complete problems: we can apply this to matching to get a 2-approximation.

**Lemma 2.1.** The maximal matching has size at least $1/2$ of the optimum.

**Proof.** Let the maximum matching be $M^*$. Consider the process of taking a maximal matching: an edge in $M^*$ can’t be chosen only after one (or both) of its endpoints are picked.

Each edge we add to the maximal matching only has 2 endpoints, so it takes at least $\frac{M^*}{2}$ edges to make all edges in $M^*$ ineligible. □

This ratio is indeed close to tight: consider a graph that has many length 3 paths, and the maximal matching keeps on taking the middle edges.

We now apply this to the VERTEXCOVER problem. This problem’s decision version is whether there exists a vertex cover of size $\leq k$. It is in NP because a verifier can just check the vertex cover. We now show that it’s NP-complete by reducing from CLIQUE. Recall the following fact about the close connection between CLIQUE and VERTEXCOVER.

**Lemma 2.2.** Let $G$ be the complement of $G$: $uv$ is connected in $G$ if and only if $u$ and $v$ are not connected in $G$.

$G$ has a clique of size $k$ if and only if $\overline{G}$ has a vertex cover of size $|V| - k$.

This implies

$$\text{CLIQUE} \rightarrow \text{VERTEXCOVER}$$

as well as

$$\text{VERTEXCOVER} \rightarrow \text{CLIQUE},$$

which gives an alternate proof that CLIQUE $\in$ NP. (Recall that $A \rightarrow B$ means we can convert an input of $A$ to an input of $B$, or if we can solve $B$, we can use it to solve $A$.) However, because the proof of a problem is in NP can usually be done in directly (as in this case), we usually only focus on ONE side of the reduction when using such structural results.

We now give an 2-approximate algorithm for VERTEXCOVER. This problem, unlike matching, is a minimization problem. So we will define the approximation ratio accordingly as:

$$\max_i \frac{A(I)}{OPT(I)}.$$

The algorithm is quite simple:
1. Compute a maximal matching $M$.
2. Return all $2|M|$ endpoints of edges in $M$.

This solution is a vertex cover because if there is an uncovered edge, it can be added to $M$ to form a larger matching.

Furthermore, we still have

$$|M| \leq OPT$$

since each edge must have at least one end point chosen, and the edges in $M$ don’t share endpoints.

This means the size of our solution, $A(I) = 2|M|$, is at most $2OPT$.

3. **VertexCover $\rightarrow$ MaxCut**

In the last bit we show that MaxCut is NP-hard by reducing VertexCover to it. Combining this with checking that it’s in NP gives that MaxCut is also NP-complete.

The construction is as follows: given a graph $G$ with vertices $v_1 \ldots v_n$, with degrees $\text{deg}(v_1) \ldots \text{deg}(v_n)$, we create a new graph $H$ by

1. adding a new vertex $w$, and
2. connect $w$ with each $v_i$ by $\text{deg}(v_i) - 1$ parallel edges.

We claim $H$ has a cut of size

$$2m - k$$

if and only if $G$ has an independent set of size $k$.

Consider a cut in $H$. Let $S$ be the half that does not contain $w$. The number of edges cut is

$$\sum_{v \in S} (\text{deg}(v) - 1) + |uv : u \in S, v \notin S| = \sum_{v \in S} \text{deg}(v) + |uv : u \in S, v \notin S| - |S|.$$

Each edge $uv$ with both end points in $S$ gets counted twice: once per degree term. Each edge $uv$ with one end point in $S$ also gets counted twice: once in the degree term, and once among the number of edges cut. So the above expression equals to

$$2 : \text{number of edges incident to } S - |S|.$$

Note that if we have an edge with both endpoints outside of $S$, we can always increase the above objective by adding one of its endpoints to $S$. Therefore the only $S$ we need to consider are independent sets. And for those, the objective becomes

$$2m - |S|,$$

which is at least $2m - k$ if and only if $|S| \leq k$. 

3