Graph Isomorphism

We began our discussion of the graph isomorphism algorithm. We noted that one way to solve this problem was to obtain a canonical isomorph=canonical ordering of an input graph. To determine if two graphs are isomorphic, we could find the canonical isomorph of both and see if they are equal. We can also use canonical isomorphs to create a dictionary of ,e.g. known chemical compounds. Then to test if an input compound is one of the known ones, we simply create its canonical isomorph and see if it is in the dictionary.

We noted that one canonical way to build an isomorph is to use $C_{\leq}(G)$ defined as follows. For any labelling of V(G) using $v_1,\ldots,v_n$. we let $x_{i,j}$ be 1 if v_iv_j is an edge and 0 otherwise. Then, $C_{\leq}(G)$ is the labelling which lexicographically maximizes $(x_{1,2},x_{1,3},\ldots,x_{1,n},x_{2,3},\ldots,x_{2,n},\ldots,x_{n-1,n})$. However, this algorithm can take quite a while. One must look at essentially all orderings- which would take $\Omega(n!n^2)$ time, if we kept track of the best ordering found so far.

We observe that if S is any set of orderings on V(G) defined independently of the vertex labels and we take the labelling in S which lexicographically maximizes $(x_{1,2},x_{1,3},\ldots,x_{1,n},x_{2,3},\ldots,x_{2,n},\ldots,x_{n-1,n})$ then this is a canonical labelling.

In order to apply this observation in order to find a canonical labelling more efficiently, we consider using the Partition Refinement procedure. Here we start with some ordered partition and repeatedly apply the following procedure:

If the current ordered partition is $P_1,\ldots,P_l$ then for each vertex x, we determine a degree sequence $(d_1(x),\ldots,d_l(x))$ where $d_i(x)$ is the number of neighbours of x in P_i We order the vertices within each part by non-increasing lexicographic order on these sequences and refine the partition so that two elements of P_i are in the same new partition element precisely if they have the same degree sequence.

We continue to refine until the refinement did not split any partition element into smaller partition elements.

We perform at most n iterations, since each creates a new partition element. In each iteration we can sort on the ith element of the degree sequence using a bucket sort in O(n) steps. So the sorting takes $O(n^2)$ steps. In refining we just need to check which pairs of adjacent elements in our sort of P_i have different degree sequences which can also be done in $O(n^2)$ time, O(n) per pair.

The most simplistic approach is to apply this to an initial partition which has only one element V. If the refinement of this partition our refinement procedure returns has only singleton elements, we are done. Of course, in certain symmetric graphs, this will not happen, indeed in a regular graph, our final partition has only one element which is all of V. In such cases, more sophisticated algorithms resort to

group theoretic approaches which break symmetry to obtain a somewhat efficient algorithm. However, we saw that for almost every graph, the refinement procedure will work without any group-theoretic addition, and furthermore that the probability that it fails is so small that applying the refinement procedure in a slightly more sophisticated way if it fails, does yield to a canonical labelling algorithm whose running time, averaged over all inputs (where each graph on n vertices is equally likely) is $O(n^3)$.

To wit, if running the refinement procedure starting with a one element partition fails then we taking the lexicographically largest ordering respecting the refinement provided by running our algorithm on some partition with $n^{1/10}$ singletons and all the remaining vertices in a single partition element. This yields an algorithm whose expected running time on a uniformly chosen input was $O(n^3)$.

We sketched a proof (for which you are not responsible) that the probability that our refinement algorithm did not return a partition consisting only of singletons was at most o(2 to the $-n^{1/4}$). So the probability we run the second phase is o(2 to the $-n^{1/4}$). If we let b be the maximum number of vertices in the non-singleton classes over all the (n to the $n^{1/10}$) partitions constructed in the second phase, we see that the second phase takes at most O((n choose $n^{1/10}$)(n^3+b^{b+2})) time. If b is at most $n^{1/10}$ then this is O(2 to the $n^{1/4}$). So the expected time taken in the second phase for inputs where b is at most $n^{1/10}$ is o(1). We also saw that the probability that b exceeds $n^{1/10}$ is o((n to the $n^{1/10}$)(n^3+n^{n+2})). So the expected time taken in the second phase for inputs where b exceeds $n^{1/10}$ is o(1). Thus the second phase takes expected time o(1) and we are done.

It remains to show:

Lemma 4: The probability that there is an (ordered) set S of at least $n^{1/10}$ vertices such that the following fails is o(1/(n to the $n^{1/10}$)(n^3+n^{n+2})):

(*) refining starting from the partition where the partition elements are the vertices of S(in order) followed by V-S, does not lead to a partition with at most $n^{1/10}$ vertices in non-singleton partition elements.

Proof: We need only show that for a particular S, the probability that (*) fails is o(1/(n to the $n^{2/10}$)(n^3+n^{n+2})) which is $o(n^{2n})$, ie. $o(2^{2n\log n})$.

We do this in two parts, we first show that the probability that after one refinement we have at least n/2 singletons is $o(2^{2n\log n})$, we then show that the probability that after one refinement we have at least n/2 singletons but after two refinements we have fewer than $n-n^{1/10}$ singletons is $o(2^{2n\log n})$.

To prove the first part, we let F be the vertices in non-singleton partition elements in the refinement returned after the first iteration which appear first in their partition and B be the remaining vertices in non-singelton partition elements of this

partition. Obviously neither B nor F intersects S, and |B| is at least |F| and hence at least n/4. We note further that the every vertex of B must have the same neighbourhood on S as vertex of F on S. Now, there are at most 4^n choices for a partition of V into four sets S,A,B,V-A-S. For each such choice, there are at most $|F|^{|B|}<n^{|B|}$ ways of pairing each vertex of B with a vertex of F which has the same neighbourhood on S. The probability that the vertices of B do have the same neighbour as the specified vertices of F is $2^{-|S||B|}$, so summing over all choices of S,S,A,B, and the pairing we see that the probability that an input graph has such an F,B is at most $4^n n^{|B|} 2^{-|S||B|}=2^{2n+(\log n-|S|)|B|}$. Since $|B|>n/4$ and $|S|>n^{1/10}$, this is $o(2^{-n\log n})$.

To prove the second part, we let S' be the set of singleton partition elements in the refinement obtained after the first iteration and let F be the vertices in non-singleton partition elements of the refinement returned after the second iteration which appear first in their partition and B be the remaining vertices in non-singelton partition elements of this partition. . Obviously neither B nor F intersects S'. By hypothesis, $|S'| >n/2$ and |B| is at least |F| and hence at least $n^{1/10}/2$. We note further that the every vertex of B must have the same neighbourhood on S' as vertex of F on S'. Now, there are at most 4^n choices for a partition of V into four sets S',A,B,V-A-S'. For each such choice, there are at most $|F|^{|B|}<n^{|B|}$ ways of pairing each vertex of B with a vertex of F which has the same neighbourhood on S'. The probability that the vertices of B do have the same neighbour as the specified vertices of F is $2^{-|S'||B|}$, so summing over all choices of S',A,B,V-A-B-S' and the pairing we see that the probability that an input graph has such an F,B is at most $4^n n^{|B|} 2^{-|S'||B|}=2^{2n+(\log n-|S'|)|B|}$. Since $|S'|>n/4$ and $|B|>n^{1/10}/2$, this is $o(2^{-n\log n})$.