PROBLEM SET 7 – Manav Bilakhia

CSC 250, Spring 2022
Due: Wednesday, Week 9

Aaron G. Cass
Department of Computer Science
Union College

LOWER BOUNDS

1. Establish a **tight asymptotic lower bound** for the *Sorted Search Problem*. In sorted search, you are given a sorted array A of n numbers and a single key k and you are to determine the index i such that $A[i] = k$, or determine that no such index exists. Recall that to establish a tight asymptotic lower bound, you must prove that no algorithm can do asymptotically better than the lower bound and also provide an algorithm that is as efficient as the lower bound.

Answer:

$\forall$ algorithms that solve this problem, must carry out comparisons between the key k and elements of the array A[]. For each comparison, there could be three possible outcomes:

- k $<$ A[i]
- k = A[i]
- k $>$ A[i]

This can be depicted using a binary decision tree where each node represents a comparison.

- if k $<$ A[i] $\rightarrow$ left branch
- if k = A[i]$\rightarrow$ terminate
- if k $>$ A[i]$\rightarrow$ right branch

By the end of making this decision tree we know that there are n leaves for every possible element +1 if the element is not found. Therefore there are a total of $n + 1$ leaves in this decision tree. From this we can calculate the height of the decision tree to be $log_2 n$ which also is the lower bound for this problem. Therefore we know that any algorithm that uses comparisons must have a lower bound of $log_2 n$ and no algorithm that uses comparison can outperform the lower bound of $log_2 n$. An algorithm we know that works as efficiently as the lower bound is Binary search. From Problem set 6 we know that the lower bound for binary search is $log n$.

2. Recall the Euclidean version of the traveling salesperson problem (TSP), in which you must find the shortest tour of a set of n points in the plane. While the problem is known to be NP-complete and is therefore believed to require exponential time in the worst case, this has yet to be proven by any of the many computer scientists and mathematicians working on the problem. However, some conservative lower bounds can be established.

   (a) Prove that the Euclidean TSP problem is in the $\Omega(n \log n)$ worst-case time efficiency class.

   (b) Prove again that the Euclidean TSP problem is in the $\Omega(n \log n)$ worst-case time efficiency class. **Use a different approach than you did in part 2a**.

Answer:

(a) Proof by decision tree
ETSP takes an array of of size n which contain a group of points as input and then determines the most optimal path that goes through every point. To be able to find the most optimal path, the algorithm must prepare a list of all possible permutations for the given group of points. Hence

there must be n! possible permutations. The number of all possible permutations which is also the number of all possible outcomes represents the number of leaves present in decision tree of an algorithm. If there are n! leaves in the decision tree for ETSP then its height must be $nlogn$. Therefore it belongs to $\omega(nlogn)$ worst case time efficiency class.

(b) Proof by Reduction
We know that comparison based sorting belongs to the worst case time efficiency class of $\omega(nlogn)$. To show that ETSP also belongs to the same worst case time efficiency class, we must be able to solve comparison based sorting using ETSP as a sub routine. In other words we must be able to reduce comparison based sorting to ETSP.

MYSORT($A[1....n]$)
Input: Takes an array as an input
Output: returns the sorted array

1: Construct a set of points on the x-y plane such that each element A[i] of the array is the x coordinate of a point and $(A[i]^2)$ is the y-coordinates and store these points in another array L[1....n]
2: M $\leftarrow$ ETSP(L[0....n])
3: truncate the y-coordinate of each point in array M[] such that M[] now only holds the x coordinates of these points that were returned by ETSP.
4: traverse through the list, if the the list is in descending order and there exists an M[i] such that M[i+1] $>$ M[i] then move all elements starting from M[0] to M[i] after M[n]. If the list is ascending and there exists an M[i] such that M[i+1] $<$ M[i] then move all elements starting from M[i] to M[n] before M[1]
5: **return** M[]

Since we were able to perform this reduction, we know that ETSP is in $\omega(nlogn)$ worst case time efficiency class.

HONOR CODE AFFIRMATION

I affirm that I have carried out my academic endeavors with full academic honesty
Manav Bilakhia