Mandatory Part

Problem 1. Fast Approximate Near Neighbor in $\ell_1$. Construct a new data structure for the Approximate Near Neighbor problem in $\mathbb{R}^d$ under $\ell_1$ norm. Your data structure should have the following parameters:

- Approximation factor: $O(d)$
- Space: $O(dn)$
- Query time: $O(d)$

Your data structure can be randomized.

Problem 2. LSH in $\mathbb{R}^d$ under $\ell_1$. In the class, we have seen how to embed $\{0\ldots M\}^d$ equipped with $\ell_1$ norm, into the Hamming space $\{0,1\}^M$. This automatically yields a randomized data structure solving a $c$-approximate Near Neighbor with query time $O(dMn^{1/c})$, for $c = 1 + \epsilon > 1$.

Show how to extend the latter data structure so that it works for points in $\mathbb{R}^d$ (again, the distance is defined by the $\ell_1$ norm). Your data structure should support queries in time $O((d \log n/\epsilon)^{O(1)}n^{1/c})$.

Your data structure can be randomized.

Hint: Problem 1, and Problem 2, can be solved using similar methods.

Problem 3. Distortion Lower Bounds

(a) Show that there exists an $n$-point metric $M$, such that for any constant $d \geq 1$, any embedding of $M$ into $\mathbb{R}^d$ (under the $\ell_2$ norm), has distortion $\Omega(n^{1/d})$.

(b) Let $M'$ be the shortest path metric of a star with 3 leaves. Formally, $M' = (X, D)$, where the point set is $X = \{1,2,3,4\}$, and the distance between points $i \neq j$ is

$$D(i,j) = \begin{cases} 1 & \text{if } i = 1, \text{ or } j = 1 \\ 2 & \text{if } i, j \neq 1 \end{cases}$$

Show that there exists a constant $c > 1$, such that any embedding of $M'$ into $\ell_2$ (with unbounded dimension), has distortion at least $c$. 

Optional Part

Theoretical Problem. \((1, 2) - B\) metrics  In the class, we have seen how to construct an exact embedding of a given metric \(M = (X, D), |X| = n\), into \(\ell_\infty^d\). In this problem we consider embeddings of a special subclass of metrics called \((1, 2) - B\) metrics. A metric is a \((1, 2) - B\) metric if it satisfies the following two very particular conditions:

1. All non-zero distances are either 1 or 2
2. For any point \(p \in X\), the number of points \(q \in X\) such that \(D(p, q) = 1\) is at most \(B\).

Show that there is a constant \(C\) such that any metric \(M\) satisfying the above conditions can be embedded exactly into \(\ell_\infty^d\) where \(d = CB \log n\).

Hint: Use probabilistic method, similar to the proof of Matousek’s theorem.

Note: You might wonder: why anyone would be interested in \((1, 2) - B\) metrics? It turns out that it is possible to show that, for a certain constant \(A > 1\), it is NP-hard to find an \(A\)-approximate solution the Traveling Salesman Problem for such metrics (this is a much stronger fact than the NP-hardness of the exact TSP showed in the Intro to Algorithms class). This remains true even if \(B\) is constant.

The embedding implies that the problem is equally hard even if the metric is induced by \(n\) points living in \(l_\infty\) with dimension \(d = O(\log n)\). So, any \(A\)-approximation algorithm for this problem is unlikely to run in time \(2^{2^{2d}}\). Otherwise, we would have an algorithm solving an NP-hard problem in time \(2^{2^{2d}} = 2^{2^{O(\log n)}} = 2^{2^\Theta(1)}\), i.e., in sub-exponential time, which is conjectured to be impossible.

So, the problem of approximately solving TSP in \(d\)-dimensional \(l_\infty\) norm suffers from doubly exponential dependence on \(d\). This is a "super-curse of dimensionality"!

Programming Problem. Construct a Java Applet that solves the following problem. Given a set \(P\) of random points on the plane, compute a TSP tour for \(P\). Each coordinate of a point in \(P\) is distributed uniformly at random in the unit interval (therefore each point is distributed uniformly in the unit square).

You can use any heuristic approach you prefer. Your algorithm does not have to compute an exact solution, or even a provably good approximation. However, to get the full score, the expected cost of the solution should grow as \(O(\sqrt{n})\) (if in doubt, ask Tasos, or Piotr).

The user should be able to specify the number of points \(n\), and the Applet should visualize the TSP tour.