The Greedy Tourist

David Pritchard[1]

[1]Department of Combinatorics and Optimization
University of Waterloo

Graduate Student Seminar, 2006

Outline

Tourist Season

- We have just landed at the airport of a foreign island, and wish to visit every village. The catch is that we didn't buy a map.
- From each village there are a number of outgoing expressways, and there are signs that indicate the destination of each one.
- We maintain our own map of the villages we've been to and the destinations of all outgoing expressways. How can we visit all of the cities efficiently?

A Greedy Approach

- Here is a greedy strategy for exploring the island.
  1. Initially our map contains the airport's village and its outgoing expressways.
  2. Whenever we visit a previously unvisited village, add its outgoing expressways to the map.
  3. As long as there is some unvisited village, use the map to determine the closest unvisited village, and go there by traversing a series of known expressways.
- Note: we might have to travel through villages that have already been visited in order to get to an unvisited village.
- We call a "step" the process of moving from one village, along an outgoing expressway, to an adjacent village.
- How many steps will the tourist take, in the worst case?

Modeling the Greedy Approach

- Model the island by an *unweighted* graph $\mathcal{G} = (V(\mathcal{G}), E(\mathcal{G}))$.
- villages $\leftrightarrow V$, expressways $\leftrightarrow E$, city with airport $\leftrightarrow v_1 \in V$.
- In the tourist story, nodes become *known* when we go to one of their neighbours for the first time; the tourist greedily picks the closest known unvisited vertex, repeatedly.
- Claim: even if the whole graph had been known to the tourist in the first place, the tourist would still produce the exact same behaviour. Follows from a lemma on the next slide:

Lemma

At any point during the tourist's trip, all nearest unvisited vertices are known.

Proof.

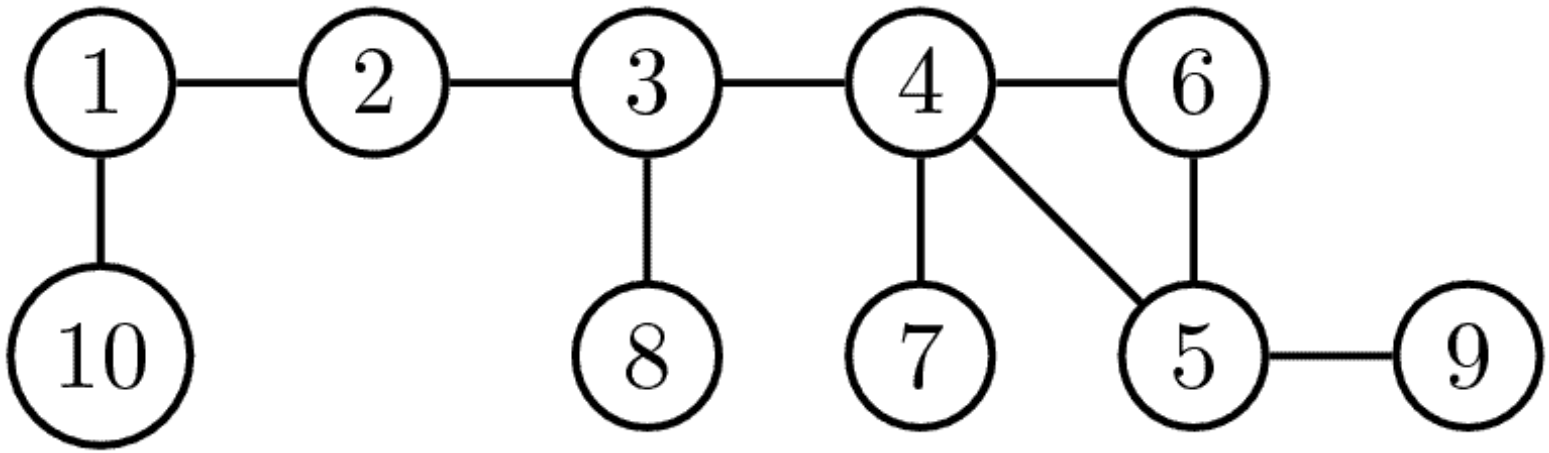
Suppose some nearest unvisited vertex v is unknown. Consider any shortest path from the tourist's current position to v. Since v is unknown, the vertex w preceding v on that path is unvisited. But then w is unvisited and strictly closer to the tourist's current position than v. □

- The greedy tourist algorithm may thus be described as follows.

1: Let $pos := v_1; visited := \{v_1\}$
2: **while** $visited \neq V(\mathcal{G})$ **do**
3: Select $dest \in (V(\mathcal{G}) - visited)$ such that $d_{\mathcal{G}}(pos, dest)$ is minimum, breaking ties arbitrarily ▷ (nondeterministic step)
4: $pos := dest; visited := visited \cup \{dest\}$
5: **end while**

An Example on Ten Nodes

- Below is an undirected, unweighted graph on 10 vertices.
- The nodes are labeled in the order that they are visited. For example the airport where we start is labeled “1.”



- In total the tourist takes

$$1 + 1 + 1 + 1 + 1 + 2 + 3 + 4 + 6 = 20$$

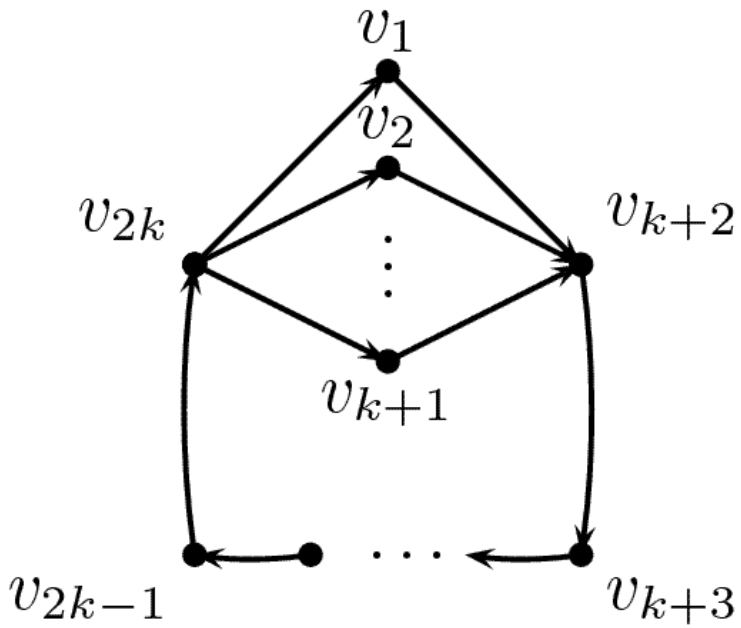
steps.

Analysis of the Greedy Approach

- Hereafter let n denote the total number of vertices.
- We should never get stuck; thus we assume that the graph is strongly connected.
- To reiterate, the main question is: **In the worst case, how many steps might the tourist take, as a function of n?**
- Here is a simple upper bound.
- We need to visit n cities in total.
- For any vertices v, v' we must have $d_{\mathcal{G}}(v, v') \leq n - 1$.
- Thus the total number of steps is less than n^2.

Bad Performance in Directed Graphs

- It turns out that the n^2 bound is essentially as good as can be proved if we allow the graph to have directed edges.
- Consider the following graph on $n = 2k$ vertices. There are $k + 1$ vertices $v_1, \ldots, v_{k+1}$ in the center region. The other vertices $v_{k+2}, \ldots, v_{2k}$ form a path of length $k - 2$.
- Starting from v_1, one must make at least k trips around the loop in order to visit all of the vertices in the center.
- Thus the tourist will take at least $k(k - 2 + 1 + 1) = k^2 = \Omega(n^2)$ steps.
- Therefore in directed graphs the $O(n^2)$ upper bound is tight.



The Undirected Case: Connection to TSP

[Rosenkrantz, Stearns, and Lewis, 1977]

- When the graph is undirected, the performance is much better than n^2 steps, namely $O(n \log n)$.
- By using [RSL 77] this bound is easily proved.
- That paper is concerned with approximate solutions to the Traveling Salesman Problem on complete undirected graphs with edge weights that satisfy the triangle inequality.
- They show that the length L of the tour taken by the tourist satisfies

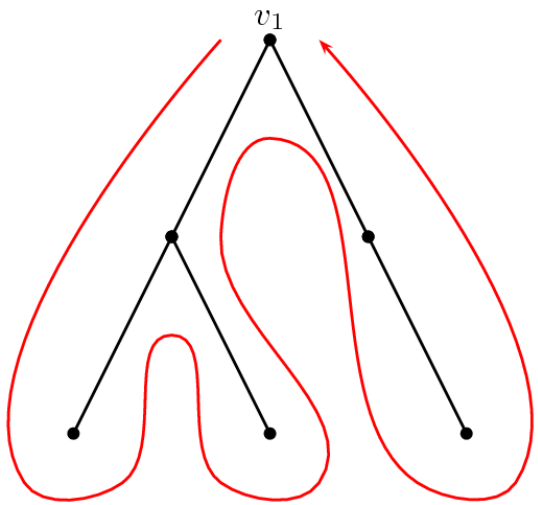
$$L \leq \left(\frac{1}{2} \lceil \log_2 n \rceil + \frac{1}{2} \right) \text{MinTSP},$$

where MinTSP is the minimum possible cost of a Hamilton circuit in $\mathcal{G}$.

The Undirected Case: Connection to TSP, II

[Rosenkrantz, Stearns, and Lewis, 1977]

- In our case the weight of the edge between u and v is just $d_{\mathcal{G}}(u, v)$.
- Note that $d_{\mathcal{G}}$ satisfies the triangle inequality.
- Let $\mathcal{T}$ be any spanning tree of $\mathcal{G}$.
- Walking around the outside of $\mathcal{T}$ takes $2(n-1)$ steps.
- By the triangle inequality, visiting nodes in preorder gives a Hamilton circuit of cost $\leq 2(n-1)$ in $\mathcal{G}$.
- Thus MinTSP $\leq 2(n-1)$ in our case, and consequently $L = O(n \log n)$ by [RSL 77].



What's To Come

- We will see a simpler proof of the upper bound using vertex partitions.
- The authors of [RSL 77] give an explicit construction showing that the greedy tourist may walk a total distance of

$$\Omega(\log n \cdot \text{MINTSP}).$$

- However, their construction uses weights and cannot be modified to work for the unweighted case.
- We will see an unweighted construction in which the tourist takes $\Omega(n \log n)$ steps, proving that our upper bound is tight.

Outline

Overview: Proof of the $O(n \log n)$ Upper Bound

- A node is *explored* when it is visited for the first time.
- Henceforth, consider a specific execution of the greedy algorithm. Label the nodes, in order of exploration, $(v_1, v_2, \ldots, v_n)$.
- For a given traversal, call the steps from v_i to v_{i+1} the ith *leg* of the traversal, and let $\ell_i = d_{\mathcal{G}}(v_i, v_{i+1})$ be the length of leg i.
- So we want to bound $L = \sum_{i=1}^{n-1} \ell_i$.
- First, we rephrase the desired bound in terms of counting long legs.
- Next, we observe that, if the vertices can be partitioned into few graphs of small diameter, then there aren't many long legs.
- Finally, we show how every graph has a vertex partition of this form.

Rephrasing the Problem

- The total number of steps which the tourist takes is $L = \sum_{i=1}^{n-1} \ell_i$.
- Use an idea from partition theory: define a *conjugate* sequence λ_j such that $\sum \ell_i = \sum \lambda_j$.
- Specifically, we have that

$$\begin{aligned} \sum_{i=1}^{n-1} \ell_i &= \sum_{i=1}^{n-1} \sum_{j=1}^{\ell_i} 1 \\ &= \sum_{j=1} \sum_{i:\ell_i \geq j} 1 \\ &= \sum_{j=1} \#\{i \mid \ell_i \geq j\}. \end{aligned}$$

- Thus we define $\lambda_j = \#\{i \mid \ell_i \geq j\}$ and will bound the λ_j's instead.

Bounding the λ_j with Vertex Partitions of Small Diameter

Lemma

Let $\{P_j\}_{j=1}^{k}$ be a partition of V, and $D = \max_j \mathsf{Diam}(G[P_j])$. Then $\lambda_{D+1} \leq k$.

Proof.

Let v_i be a node from the walker's tour, and P_j be the component containing v_i. Suppose that v_i is not the last node of P_j to be explored, say $i' > i$ satisfies $v_{i'} \in P_j$. Then after exploring v_i, since the tourist is greedy and $v_{i'}$ is not yet visited, he will choose to explore v_{i+1} such that

$$\ell_i = d_{\mathcal{G}}(v_i, v_{i+1}) \leq d_{\mathcal{G}}(v_i, v_{i'}) \leq \mathsf{Diam}(G[P_j]) \leq D.$$

Thus, for each part P_i, $1 \leq i \leq k$, at most one leg from a vertex of that part (namely, the last leg) has length $D+1$ or more. Thus at most k legs are of length $D+1$ or more. The result follows since $\lambda_{D+1} = \#\{i \mid \ell_i \geq D+1\}$. □

The Existence of Good Partitions

Lemma

Let t be a non-negative integer. There exists a partition of $V(\mathcal{G})$ into at most $\lceil n/t \rceil$ parts, such that $\mathsf{Diam}(\mathcal{G}[P]) \leq 2t - 2$ for each part P.

Proof. (Sketch).

Let $\mathcal{T}$ be any rooted spanning tree of $\mathcal{G}$.
As long as $\mathcal{T}$ has height t or more, we cut off a subtree of height $t - 1$ and make its vertices into a part. Eventually the tree has height $< t$ and we put all remaining vertices into a part. A (sub)tree's diameter is at most twice its height, so each part induces a subgraph of diameter at most $2(t - 1)$ in $\mathcal{T}$. Since each part, except possibly the last one, contains at least t vertices, we have no more than $\lceil n/t \rceil$ parts in total.
But as replacing the nontree edges cannot increase the distance between two nodes, $\mathsf{Diam}(\mathcal{G}[P]) \leq \mathsf{Diam}(\mathcal{T}[P]) \leq 2(t - 1)$ for each part P. □

Conclusion of the Upper Bound

- From the second lemma, for each t, there is a partition of the vertices into $\lceil n/t \rceil$ parts with diameter at most $2t - 2$.
- Thus, by the first lemma, $\lambda_{2t-1} \leq \lceil n/t \rceil$.
- Note that λ_j is nonincreasing, so $\lambda_{2t} \leq \lceil n/t \rceil$.
- Also, $\lambda_j = 0$ for $j > \mathsf{Diam}(\mathcal{G})$.
- It follows that the total number of steps is

$$\begin{aligned}
\sum_{i=1}^{n-1} \ell_i &= \sum_{t=1}^{\lceil \mathsf{Diam}(\mathcal{G})/2 \rceil} \lambda_{2t-1} + \lambda_{2t} \\
&\leq \sum_{t=1}^{\lceil \mathsf{Diam}(\mathcal{G})/2 \rceil} 2\lceil n/t \rceil \\
&= 2n \ln(\mathsf{Diam}(\mathcal{G})) + O(n).
\end{aligned}$$

- (Or if $\mathcal{G}$ is weighted, modify 2nd lemma to get same bound as [RSL 77].)

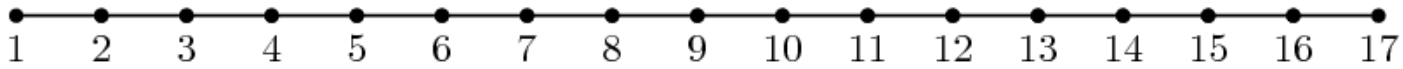
Outline

Tightness of the $O(n \log n)$ Upper Bound

- Now let's actually find a family of graphs that can take $\Omega(n \log n)$ steps.
- What about very simple graphs?
- The densest graph K_n is always traversed in $n - 1$ steps.
- The sparsest graphs — trees — take at most $2n - 3$ steps. (The greedy tourist performs a depth-first search.)
- From the upper bound of $O(n \log \mathsf{Diam}(\mathcal{G}))$, the graphs cannot have very small diameter.

The Layered Ring Construction, Phase 1

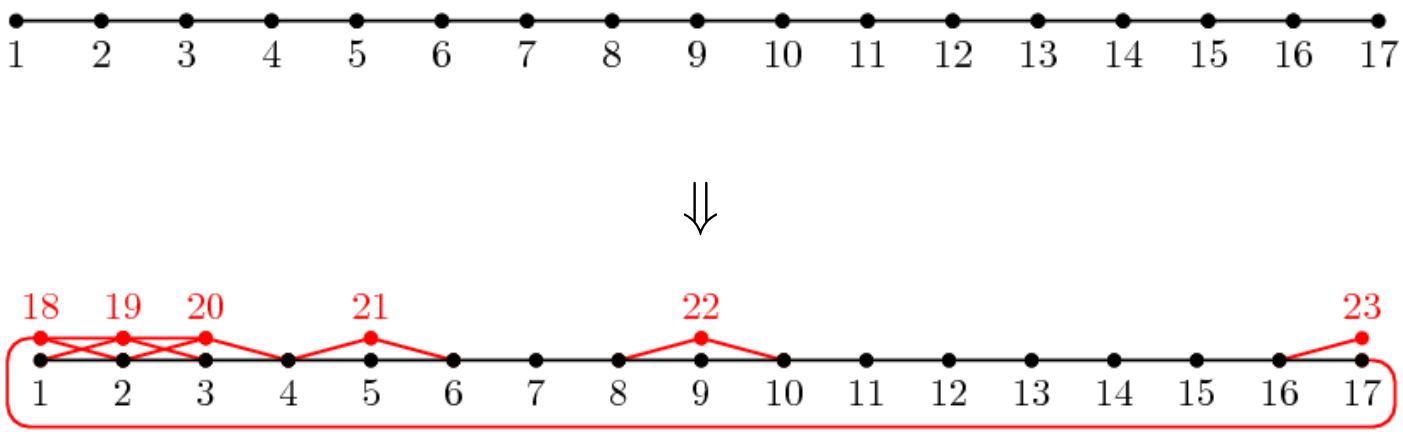
- We show the *layered ring* construction for the lower bound.
- The basic idea is to start with a long path, and then augment it using very few nodes so that the whole length of the path can be traversed many times.
- We picture these graphs with their nodes labeled according to the order of exploration.
- The graph is built in phases, starting with a path of $2^m + 1$ nodes. Here is the first phase for $m = 4$:



- A sequence of 2^m steps is called a *lap*. Each phase will add one more lap.

The Layered Ring Construction, Phase 2

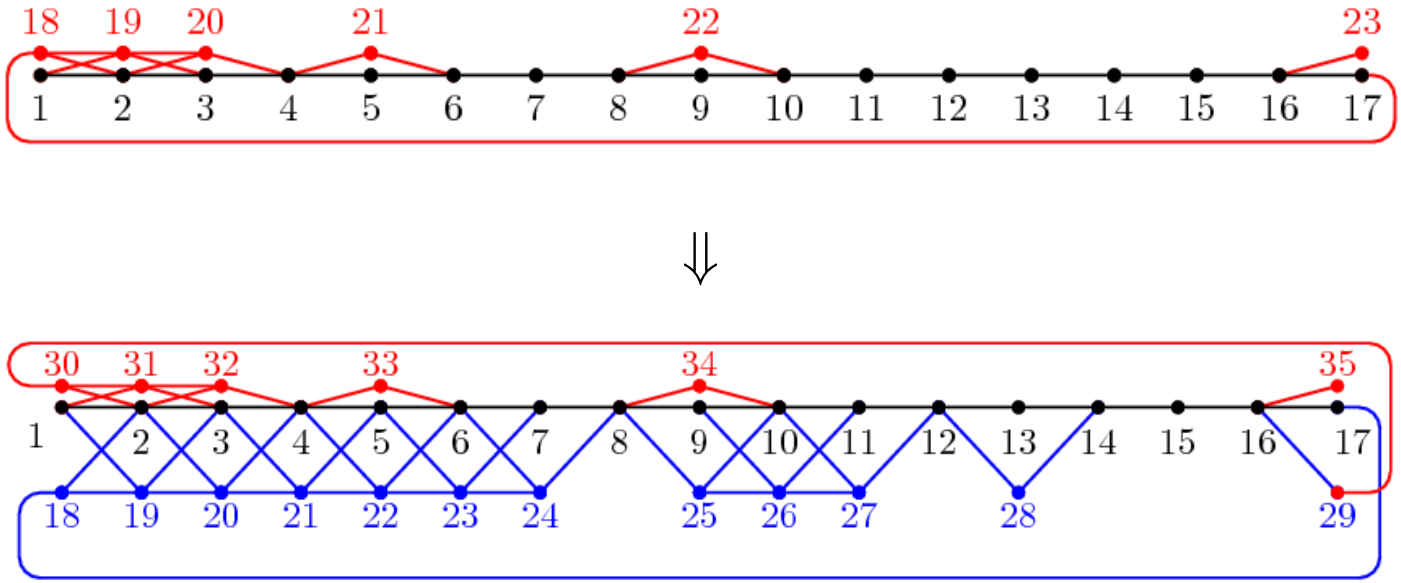
- In the second phase, add a "layer" of $m + 2$ nodes that permit the tourist to take a second lap:



- The nodes are spaced out in a binary geometric progression.

The Layered Ring Construction, Phase 3

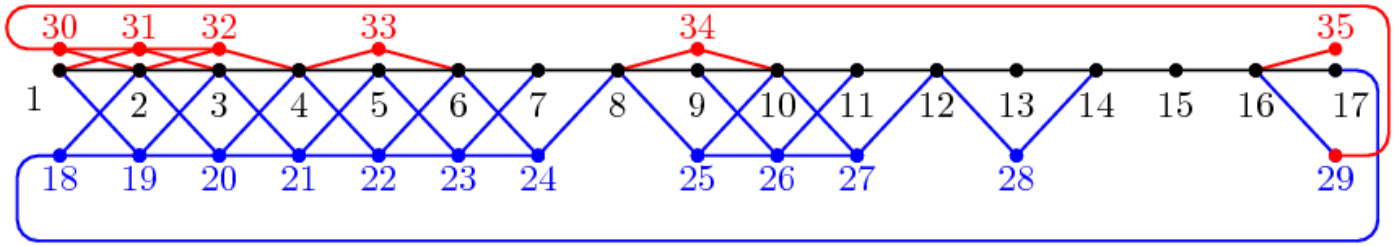
- In the third phase, we "splice" in another layer, giving three laps.



- For each leg of length 2^k added in the previous phase, we add $k + 1$ legs of respective lengths $1, 1, 2, 4, \ldots, 2^{k-1}$.
- The tourist traverses the original (black) nodes 1–17 first, then the new (blue) nodes 18–29, then finally we traverse the (red) nodes that have been renamed 30–35.

The Layered Ring Construction, Phases 3, 4, . . .

- Since the nodes are spaced in a binary geometric progression, the tourist has enough "momentum" to pick up the layers one at a time.
- For example, here is the third phase of the layered ring:



- After exploring node 27, the closest unvisited nodes are at distance 2, so the tourist can go to node 28; then the closest unvisited nodes are at distance 4, so the tourist can go to node 29.
- The fourth and later phases are just like the third. Splice in a new layer. Each leg of length 2^k from the previous phase gives rise to $k + 1$ legs of length $1, 1, 2, 4, \ldots, 2^{k-1}$ in the new phase. This adds one more lap.
- First the original $2^m + 1$ nodes are explored, then the new layer, then the second-newest, ... The sparsest layer, of $m + 2$ nodes, is explored last.

Analysis of the Construction

- The fact that a leg of length 2^k decomposes into legs of length $1, 1, 2, 4, \ldots, 2^{k-1}$ permits a simple recurrence relation for the number of nodes added in each layer. The details are omitted here.
- After $\ell = m/3$ phases, we can show that the total number of nodes is

$$n = 2^m(1 + o(1)),$$

  which also implies $m = \Theta(\log n)$.
- Thus, taking $\ell = m/3$, the total number of steps is ℓ laps, or

$$\ell \cdot 2^m = \frac{m}{3}2^m = \Theta(n \log n).$$

Outline

Application to Distributed Networks

- Suppose that we have a network of computers, modeled by a graph.
- It may be desirable to have an *agent* visit every node of the network. This can be used to conduct a census, compute aggregates in the network, assign unique IDs to each node, etc.
- If the network is reliable then DFS is essentially optimal, taking less than $2n$ steps.
- If the edges of the network are not 100% reliable, then DFS may fail (for example, if the agent can't backtrack from v to $parent(v)$). We may instead use the greedy tourist strategy to traverse the entire network.
- In order to distributively implement the shortest-path algorithm, each node v records the distance $d(v)$ to the nearest unvisited node.
- At each time step each node recomputes

$$d(v) = \begin{cases} 0, & \text{if } v \text{ has not been visited;} \\ \min_{w \in \Gamma(v)} 1 + d(w), & \text{otherwise.} \end{cases}$$

Randomized Version of the Greedy Tourist

- We have only shown that on layered rings, for a very particular exploration sequence, the greedy tourist can take $\Omega(n \log n)$ steps.
- It is possible that a particular tie-breaking rule (to determine which of the closest unvisited nodes should be explored next) could give better performance.
- What if ties are broken randomly? Two reasonable versions of this rule.
- Centralized version: after visiting a new node, of the closest unvisited nodes, pick one uniformly at random and move to it.
- Distributed version: at each step, of all outgoing edges from your current position which lie on the shortest paths to the closest unvisited nodes, pick one uniformly at random and move along it.
- What is the *expected* number of steps taken?
- Hopefully, if the $\Omega(n \log n)$ lower bound holds, some family of random graphs will exhibit this performance, simplifying analysis.

Questions?

- Thank you for listening!