Arrow and Ivy on Grids

Consider a grid graph G of n nodes, where each side has $\sqrt{n}$ nodes. Based on this grid, we define a complete weighted graph G' with the same node set V, and an edge of weight $d(u,v)$ between every pair of nodes $u, v \in V$, where $d(u,v)$ is the shortest distance between u and v in G.

In this exercise, we want to analyze the performance of Arrow and Ivy on the weighted graph G'. That is, whenever Arrow or Ivy includes an edge of G' in the spanning tree, we assume that this edge automatically has a weight of $d(u,v)$. We illustrate an example 3×3 grid G and a weighted spanning tree of G' on Figure 1. Observe that the edges of a spanning tree of G' are not necessarily edges of the original grid.

We assume that the requests to the token are sequential, i.e., the next node only requests the token after the previous request is satisfied. We measure the performance of a protocol as the overhead ratio with respect to the shortest path cost.

More formally, assume that the token is initially placed at the root r_0 of an initial tree T_0, and we are given a sequence $\sigma = r_1, r_2, \ldots, r_k$ of requests. For a protocol P, we define the cost $cost_P(\sigma, T_0)$ as $\Sigma_{i=1}^{k} d_{T_{i-1}}(r_{i-1}, r_i)$, where T_{i-1} is the weighted tree obtained after the request r_{i-1} is satisfied by P, and $d_{T_{i-1}}(r_{i-1}, r_i)$ is the distance of r_i and r_{i-1} in T_{i-1}. Similarly, we define the shortest path cost $cost(\sigma)$ as $\Sigma_{i=1}^{k} d(r_{i-1}, r_i)$, where $d(r_{i-1}, r_i)$ is the distance between the nodes r_{i-1} and r_i in the grid G. Finally, the overhead ratio of a protocol P is defined as

$$c_P(T_0, \sigma) := \frac{cost_P(\sigma, T_0)}{cost(\sigma)},$$

for a specific initial tree T_0 and a specific request sequence σ.

The next two questions analyze the Arrow algorithm on the grid graph:

Used notation (for notationally accurate reader): something like $f(n) = O(...)$ here means in rigorous notation $f(n) \in O(...)$). When something is deduced from $O(...)$ or $\Omega(...)$ in formulas, it means it works for any function set from this sets (e.g. $\frac{n^2}{n} = \frac{O(n^2)}{\Omega(n)} = O(n)$ means rigorously $\frac{n^2}{n} \in S, S = \{\text{functions f s. t. } f = h/g, h \in O(n^2), g \in \Omega(n)\})$, and $S = O(n)$ as sets.

1. **Show that there exists a tree T_0 so that $c_{Arrow}(T_0, \sigma)$ on G' is $O(\sqrt{n})$ for every request sequence σ. [3]**

   If we have T_0 as in fig. 1, then any distance in the initial graph D_0 can be estimated with the upper bound of distance in tree T_0: we need to go the same amount of hops in vertical direction (let's say $\Delta y \geq 0$), but in horizontal we will go no more than $2\sqrt{n}$ hops (from rightmost position to the rightmost). Let's denote as $\Delta x \geq 0$ horizontal shift in the initial graph. Therefore, as any round costs at least one hop (and minimal cost is obviously (as Manhattan distance) $\Delta x + \Delta y \geq 1$), at any round its cost would be $\frac{\Delta y + 2\sqrt{n}}{\Delta x + \Delta y} \leq 1 + \frac{2\sqrt{n}}{\Delta x + \Delta y} \leq 1 + 2\sqrt{n} = O(\sqrt{n})$. And if every round is such, then the final costs will give the same (as $\frac{cost_P(\sigma, T_0)}{cost(\sigma)} = \frac{cost_P(r_1, T_0) + \ldots + cost_P(r_k, T_0)}{cost_P(r_1) + \ldots + cost_P(r_k)} = \frac{O(\sqrt{n}) cost_P(r_1) + \ldots + O(\sqrt{n}) cost_P(r_k)}{cost_P(r_1) + \ldots + cost_P(r_k)} = O(\sqrt{n}) \frac{cost_P(r_1) + \ldots + cost_P(r_k)}{cost_P(r_1) + \ldots + cost_P(r_k)} = O(\sqrt{n})$.

2. **Show that there exists a tree T_0 and a sequence of requests σ_0 where $c_{Arrow}(T_0, \sigma_0)$ on G' is $\Omega(n\sqrt{n})$. [5]**

   Let's introduce coordinate system with $(0,0)$ in the left upper corner and x–axis horizontal to the right, y–axis vertical downwards. Then let our sequence of requests be just requests from the node $(0,0)$ and then from $(0,1)$ (w.l.o.g token is in $(0,1)$ at the start (the first round doesn't input much relative to infinity).

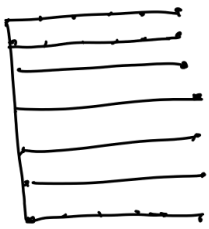

Figure 1:

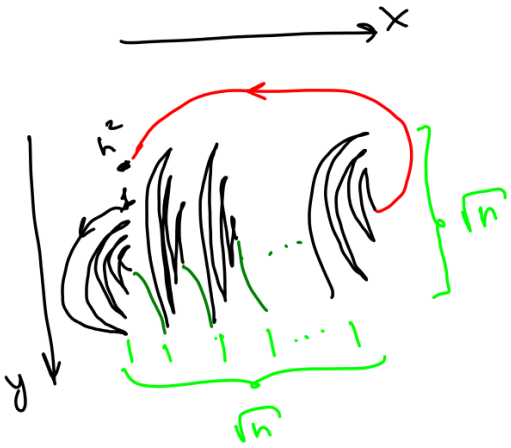

Figure 2:

Our tree T_0 will look like this (fig. 2): it's a linear tree, one end in $(0,0)$, another in $(0,1)$. It goes from $(1,0)$ to $(\sqrt{n},0)$, then $(2,0)$, $(\sqrt{n}-1,0)$ and so on, covering first one column and jumping at every step as much as possible. After there is no available nodes in column ($(0,0)$ is excluded initially), it goes to the lowest node in the next column, jumps there until all nodes in column are covered, and goes to the next. When covered the last column, it jumps to the node $(0,0)$.

Obviously, we covered all nodes and it's tree (linear). After each phase, token is in one end, and request is from another, so the cost will be the sum of all the weights of edges of the tree.

Let's estimate it.
Jumping in columns from the one (except the first) to $\sqrt{n}$ in each of them we have the length $(\sqrt{n}-1)+(\sqrt{n}-3)+\ldots$ until we reach 2 or 1 (depending on oddity). The sum of this arithmetic progression is (average sum of the first and the last) $\times$ (number of summands) $=$ $\begin{cases} \frac{\sqrt{n}+1}{2}\frac{\sqrt{n}-1}{2}, \iff \sqrt{n}-1 \mod 2 = 0 \\ \frac{\sqrt{n}}{2}\frac{\sqrt{n}+1}{2}, \iff \sqrt{n}-1 \mod 2 = 1 \end{cases}$. It's of the order $\Theta(n)$. We have $\sqrt{n}-1=\Theta(\sqrt{n})$ such columns (and also some additional length from the first, in connections of columns and in connection with $(0,0)$), but the overall order of length is then nevertheless not less than $\Theta(n)\Theta(\sqrt{n})=\Theta(n\sqrt{n})$, therefore it's $\Omega(n\sqrt{n})$.

At the same moment, usual length between $(0,0)$ and $(1,0)$ is 1, so the cost in our tree would be $\Omega(n\sqrt{n})$ times more at each round, and then overall too (proof is the same as in end of subtask 1,

changing $O(\sqrt{n})$ to $\Omega(n\sqrt{n})$). And that's what needed.

In the following, we will consider the Ivy algorithm from the lecture. Let $\sqrt{n}$ be odd, and let us choose T_0 as a shortest path tree of G with the root at the center $v_{(\sqrt{n}/2+1)(\sqrt{n}/2+1)}$ of the grid G, i.e. the path from the center node to any other node in T_0 is also a shortest path in G. Note that T_0 is a spanning tree of G'.

3. **Show that $c_{Ivy}(T_0, \sigma)$ on G' is $O(\log n\sqrt{n})$ for every request sequence σ of length at least n. [7]**

Using Theorem 7.12 from Chapter 7 of lectures [[1]] we have all proven there claims in this task too, except the one which uses that the initial graph was a star (from which comes inequality $\sum_{i=1}^{m} a_i \geq \sum_{i=1}^{m} k_i$ which we can't use here). Also, it was assumed in the theorem that every hop costs 1, but we know that any hop in our grid costs no more than diagonal, which is $\sqrt{2}(\sqrt{n}-1) = O(\sqrt{n})$, so we'll have that

$$cost_{Ivy}(T_0, \sigma) = O(\sqrt{n}\sum_{i=1}^{m} k_i). \tag{1}$$

We also have $m \geq n$ from the task description.

From this theorem we have using potential function $\Phi(T) = \sum_{u \in V} \frac{\log(s(u))}{2}$ that

$$\sum_{i=1}^{m} a_i = \sum_{i=1}^{m} k_i - \Phi(T_0) + \Phi(T_m) \tag{2}$$

and

$$a_i \leq \log n \ \forall i \Rightarrow \sum_{i=1}^{m} a_i \leq m \log n. \tag{3}$$

From (3) and (2) we then have

$$\sum_{i=1}^{m} k_i \leq m \log n + \Phi(T_0) - \Phi(T_m) \tag{4}$$

Obviously, after each step we will have rooted tree, because there can't appear loops (because Ivy–algorithm doesn't create new paths between nodes, it only deforms old ones (so if there was one between any two nodes initially, as in our shortest–path, then at each step it will be only one. And for loop we need 2 distinct paths)).

As it's noted in the aforementioned theorem,

$$\Phi(T) \geq \frac{\log n}{2} = O(n \log n) \tag{5}$$

(just because subtree of the root contains n nodes and log of natural numbers is non–negative). Also, because $s(u) \leq n$, we have

$$\Phi(T) \leq \frac{n \log n}{2} = O(n \log n) \tag{6}$$

So, then from (5) and (6) we have $\Phi(T_0) - \Phi(T_m) = O(n \log n) - O(n \log n) = O(n \log n)$ and from (4) and using $m \geq n$ we have

$$\sum_{i=1}^{m} k_i \leq m \log n + O(n \log n) = O(m \log n). \tag{7}$$

Now, from this, (1) and the obvious fact that $cost(\sigma) \geq m \Rightarrow cost(\sigma) = \Omega(m)$ (as each round we need to change root, distance between nodes is at least 1), we have:

$$c_{Ivy}(T_0, \sigma) = \frac{cost_{Ivy}(T_0, \sigma)}{cost(\sigma)} = \frac{O(\sqrt{n}\sum_{i=1}^{m} k_i)}{\Omega(m)} = \frac{O(\sqrt{n}m \log n)}{\Omega(m)} = O(\sqrt{n} \log n), \tag{8}$$

what needed.

4. **Show that there exists a shortest path tree T_0 and a sequence of requests σ_0 where $c_{Ivy}(T_0, \sigma_0)$ on G' is $\Omega(\sqrt{n})$. [10]**

   See the tree on the fig 4. It's christmas tree–like structure in every quarter of the grid, rotated when going to the next quarter (as the base we take upper–left). Obviously, it's short–paths tree (distance to any node from the central root is equal to Manhattan distance).

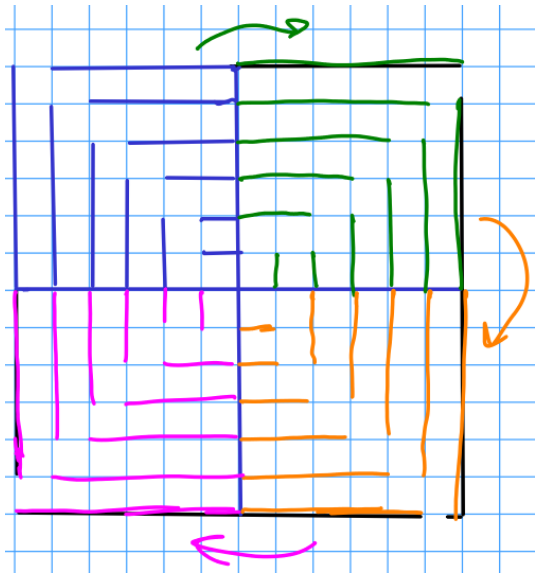

Figure 3: Tree T_0 in 4 subtask

We consider only this quarter and requests will be there, other quarters won't matter as no paths will go through them (and therefore no changes will be made there).

Consider sequence as shown in the figure 4. We consequently start requests along the diagonal of the quarter till the last node in the corner. Colors at every step: root, request node, path of request message, changed edges, edges from initial tree, path of message in the initial tree T_0 and its length (need later for counting cost), current step.

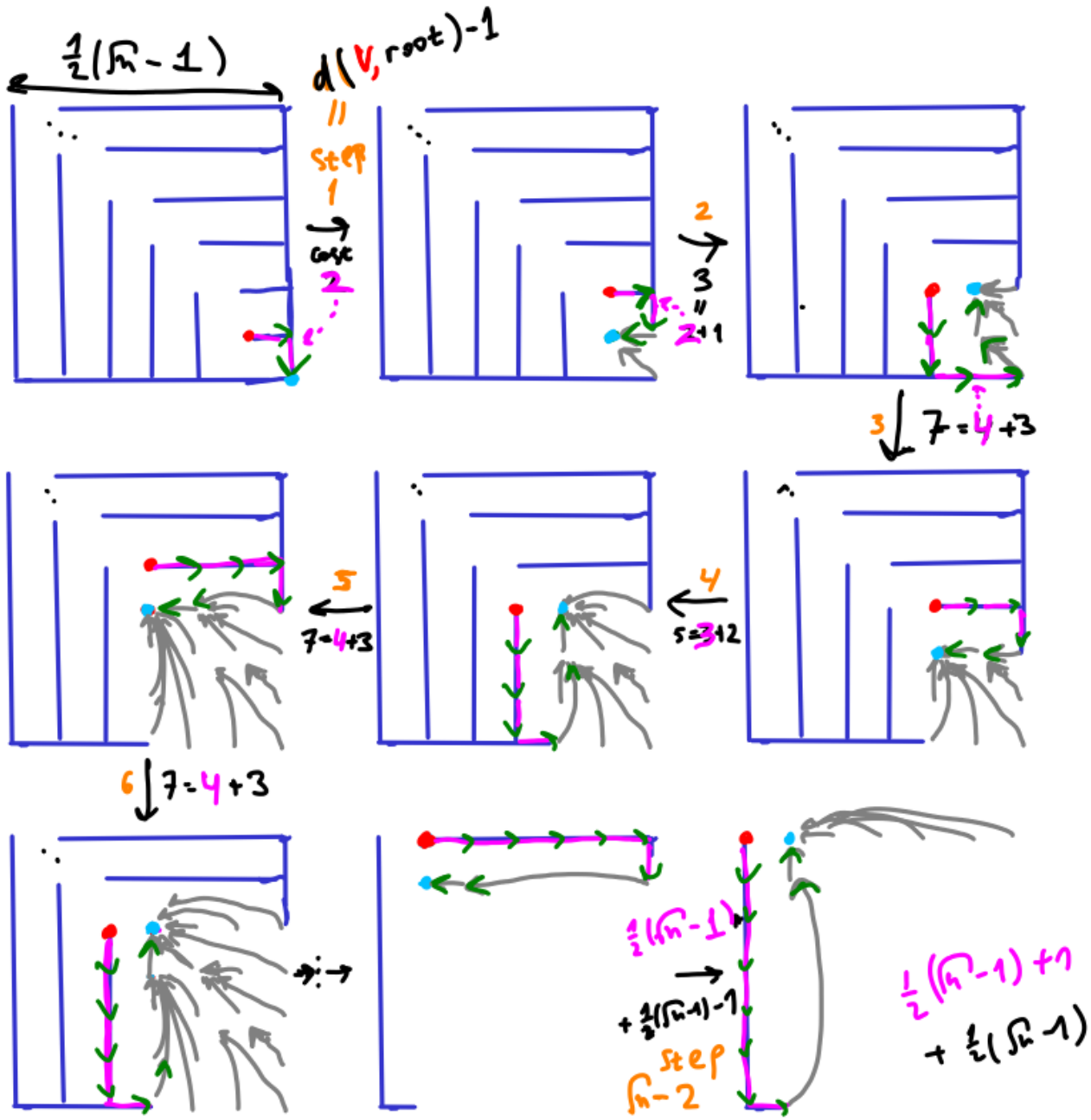

Figure 4: Tree T_0 in 4 subtask

We see, that the request node is one hop in the initial grid per step farther from initial central

root. Therefore, there will be $\sqrt{n}-1$ steps in sequence, and because root and request node are at a distance 1 in initial tree (expect distance 2 at the first step), we'll have

$$cost(\sigma) = \sqrt{n} = O(\sqrt{n}).$$

To estimate the order of $cost_{Ivy}(\sigma, T_0)$ we'll only count edges passed in the initial tree (pink at the pictures). As it could be seen, the whole subtree would be traversed, therefore length of pink edges (which is less than the whole cost), is

$$\underbrace{(\sqrt{n}-1)}_{\text{2 base lines, vert. and horiz.}} + \underbrace{\left(1+1+2+3+\ldots+(\frac{\sqrt{n}-1}{2}-1)\right)}_{\text{horiz. lines to nodes}} + \underbrace{\left(2+3+\ldots+(\frac{\sqrt{n}-1}{2})\right)}_{\text{vertical lines to nodes}}.$$

Each of the second and third summands is of the order of $\Omega(n)$ obviously (as a up to constant sum of arithmetic sequence from 1 to $\sqrt{n}/2$ with the common difference 1).

Therefore, $c_{Ivy}(T_0, \sigma) = \frac{cost_{Ivy}(\sigma, T_0)}{cost(\sigma)} = \frac{\Omega(n)}{O(\sqrt{n})} = \Omega(\sqrt{n})$, what needed.

References

[1] R. Wattenhofer, "Chapter 7." `https://disco.ethz.ch/courses/podc/lecturenotes/chapter7.pdf`.