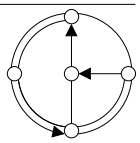
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



FS 2009 Prof. R. Wattenhofer / C. Lenzen / T. Locher

Principles of Distributed Computing
Solution 12

1 Pancake Networks

Generally, observe that $N = |V(P_n)| = n! \in O(n^n) \Rightarrow n \in O(\frac{\log N}{\log \log N})$.

a) See Figure 1. For drawing P_n, first draw n copies of P_{n-1}, each of which will have some $j \in \{1, \ldots, n\}$ fixed as the last vertex. Then there are $(n-2)!$ nodes of such a P_{n-1} connected to the same $(n-1)$-dimensional pancake. To see this, fix v_1 and v_n, the remaining node combinations in the middle will be the link between pancake $P_{n-1}|v_n$ and $P_{n-1}|v_1$. There are $n-1$ such sets in $P_{n-1}|v_n$, each connecting with another $(n-1)$-dimensional pancake.

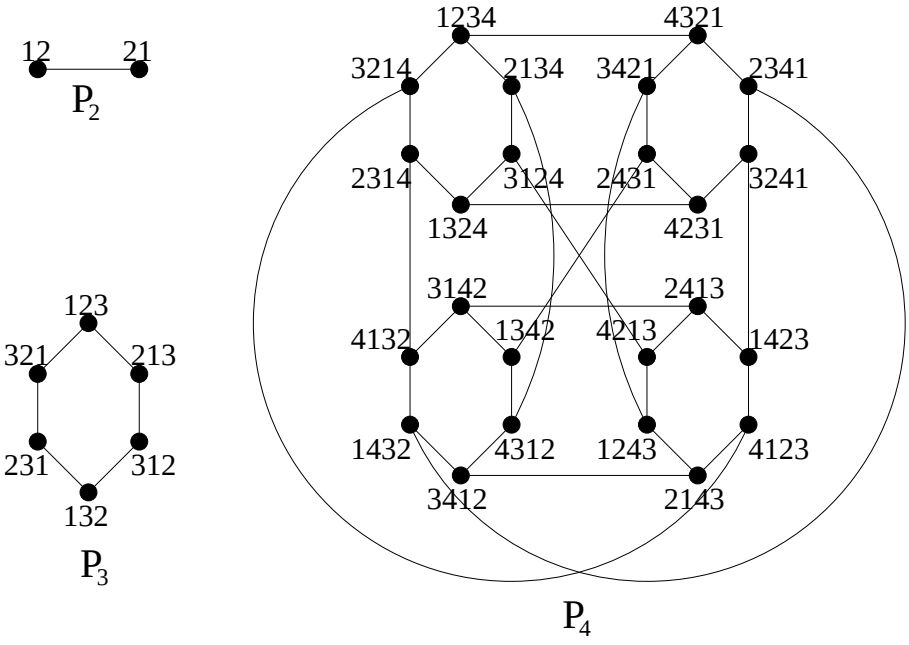


Figure 1: Pancake graphs for $n = 2, 3, 4$.

b) Let us look at the second, more intuitive definition (Eq. (1)). Basically, it states that for every node, there exists exactly one edge for every distinct prefix reversal. So the node degree of P_n can be stated as follows: how many non-trivial prefix reversals are there for a sequence of n nodes? Answer: $n-1$ with edges $e_2, \ldots, e_n$. Succinctly,

$$\deg(v) = n - 1 \qquad \forall v \in V(P_n).$$

Thus the degree of an N-node pancake graph is in $O(\log N / \log \log N)$.

c) To give an upper bound on the diameter, we need to determine in how many steps, at most, we can go from one node to any other node. Say we want to get from node $v = (v_1, \ldots, v_n)$ to node $w = (w_1, \ldots, w_n)$. As with all hypercube-like graphs, we will proceed by correcting one "coordinate" at a time. In this case, we start at the back. Since the nodes are all permutations, there will exist a v_j such that $v_j = w_n$. We bring v_j to the front by taking v's edge on level j and then to the back by taking the level n edge of the corresponding node, leading to node $v^1 = (v_n, \ldots, v_{j+1}, v_1, \ldots, v_{j-1}, v_j) = (v_n, \ldots, v_{j+1}, v_1, \ldots, v_{j-1}, w_n)$. We proceed inductively by applying the same procedure to the prefix of length $n-1$ of v^1, resulting in the node v^2 where also $v^2_{n-1} = w_{n-1}$, then the prefix of length $n-2$ of v^2, and so on. Thus, after traversing $2(n-1)$ edges (because the remaining index is automatically correct) we reach w. Therefore,

$$D(P_n) \leq 2(n-1)$$

that is, the diameter of P_n is in $O(\log N / \log \log N)$.

Gates and Papadimitriou [1] have also shown that this is asymptotically optimal, that is,

$$D(P_n) \geq n.$$

d) To show that P_n is Hamiltonian, we proceed by induction on n. We will actually show the following stronger claim: In P_n, there exists a Hamiltonian path from $(1, \ldots, n)$ to $(n, \ldots, 1)$ (the cycle then is completed by the level n edge of these nodes). Observe that since in P_n the graph looks the same from every vertex, this also holds for any given vertex v.

We have seen in **a)** that P_3 is a cycle, i.e., we have the path $(1,2,3) \to (2,1,3) \to (3,1,2) \to (1,3,2) \to (2,3,1) \to (3,2,1)$ and the final edge $(3,2,1) \to (1,2,3)$.

Assume that P_{n-1} has such a Hamiltonian path H_{n-1} from any node $(v_1, \ldots, v_{n-1})$ to $(v_{n-1}, \ldots, v_1)$. Then we can construct a Hamiltonian path in P_n by concatenating the Hamiltonian paths of the n P_{n-1} subgraphs as follows. Taking all indices mod n, define for $i \in \{0, \ldots, n\}$ that $a_i := (1-i, 2-i, \ldots, n-i)$ and $b_i := (n-i, n-1-i, \ldots, (1-i)$. We can see that b_{i+1} can be obtained from a_i by reversing its prefix of length $n-1$. Therefore, the following Hamiltonian path exists between $a_0 = (1, \ldots, n)$ and $b_n = (n, \ldots, 1)$:

$$a_0\ H_{n-1}\ b_1 \to a_1\ H_{n-1}\ b_2 \to \ldots \to a_i\ H_{n-1}\ b_{i+1} \ldots \to a_{n-1}\ H_{n-1}\ b_n,$$

where the paths H_{n-1} from a_i to b_{i+1} through P_{n-1} exist by the induction hypothesis. We complete the cycle with the edge $b_n \to a_0$.

e) In distributed hash tables, data items are hashed and the first d bits of their hash codes determine which peers are responsible for them. For example, if the node set is $V = [2]^d$, the first d bits of the hash code can be interpreted as the ID of the node where it is stored. If only a subset of all (2^d) possible IDs is used, we can use a virtual ring to determine where data is stored: Each node v has a link to the node w with the next larger ID in the network and the node with the largest ID is connected to the node with the smallest ID.[1] If the first d bits of the hash of a data item are in the range $[v, w)$, then v is responsible for this data item.[2]

If we want to store data on the pancake graph P_n, we can use the fact that P_n is Hamiltonian! Thus, if we use, e.g., the Hamiltonian path from $(1, \ldots, n)$ to $(n, \ldots, 1)$ constructed in **d)** as the ring, a unique peer can be determined just like for hypercubic networks. A file is then looked up by simply routing on the pancake to the responsible node.

References

[1] W. H. Gates, C. H. Papadimitriou, Bounds for sorting by prefix reversal, Discrete Math. 27, (1979), 47–57.

[1] If these edges are not part of the edge set E, we could simply add these edges to E, which would only increase the degree of each peer by at most 2.

[2] For example, the hash code 1010 is in the range $[1001, 1110)$. Thus, the node with the ID 1001 is responsible for the corresponding data item.