COMPUTER VERIFICATION OF THE DIAMETER OF THE N-DIMENSIONAL PANCAKE NETWORK

Marissa P. Justan
H.Benton Ellis Computer Center
Trinity College of Quezon City
Philippines
trinity@galileo.fapenet.org

Abstract

This paper is motivated by computer applications to mathematical research. In our research, we are interested in the *n*-dimensional pancake network (or graph), P_n, whose processors are labeled by permutations on *n* symbols. A link between two processors is obtained when the label of one is derived from the other by some prefix reversal. Finding the diameter of the pancake network is equivalent to finding the maximum number of pancake flips one would need to sort an arbitrary stack of pancakes. We report our computational experience with Gates and Papadimitriou's Algorithm on P_n for $n<14$.

1. Introduction

We introduce the **"Pancake Problem"** by the following quotation from **[1]**:

The chef in our place is sloppy, and when he prepares a stack of pancakes they come out all different sizes. Therefore, when I deliver them to a customer, on the way to the table I rearrange them (so that the smallest winds up on the top, and so on, down to the largest at the bottom) by grabbing several from the top and flipping them over, repeating this (varying the number I flip) as many times as necessary. If there are n pancakes, what is the maximum number of flips (as a function f(*n*) of *n*) that I will ever have to use to rearrange them?

The number of flips necessary to sort a stack of *n* pancakes is the diameter of the *n*-dimensional pancake network, P_n.

Here, we will view an interconnection network as an undirected graph. The vertices of the graph correspond to processors and the edges of the graph correspond to communication links between processors. A graph is ***regular*** if the degree of every vertex in the graph is the same. The degree of a regular graph is then the degree of any of its vertices. The ***diameter*** of a graph is the maximum distance between any pair of vertices in the graph.

The degree of a graph is a measure of the cost of the interconnection network and the diameter is a measure of the communication delay. Consequently, it is desirable to construct a large graph with small degree and small diameter. The simplest pancake graph is shown in Fig. 1.

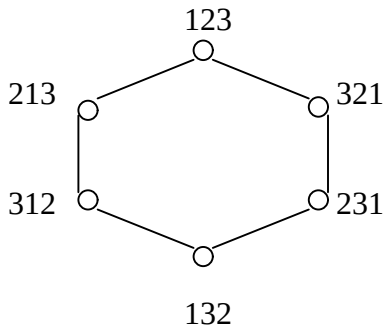

Fig. 1. 3-dimensional pancake graph, P_3.

Graph of P_4 can be viewed from **[3]**. No P_5 has been seen as of the moment.

The pancake problem is still open and Table 1 from **[3]** indicates all known exact values and lower bounds for f(*n*).

Aside from being an interesting combinatorial problem, the pancake problem is relevant to the construction of networks of parallel processors. The pancake network is an interconnection network of some interest, as its degree and diameter grow less rapidly, asymptotically, with network size than the popular hypercube.

Table 1. Known values of f(n).

n	f(n)	n	f(n)
2	1	14	≥16
3	3	15	≥17
4	4	16	≥18
5	5	17	≥19
6	7	18	≥20
7	8	19	≥22
8	9	20	≥23
9	10	21	≥24
10	11	22	≥25
11	13	23	≥26
12	14	24	≥27
13	15	25	≥28

2. Gates and Papadimitriou's Algorithm [2]

Before we present Gates and Papadimitriou's algorithm, we introduce some basic notations. We define an alphabet Σ as a finite set of symbols. The empty set, $\varnothing$, and the set consisting of the empty string $\{\in\}$ are languages. Another language is the set of all strings over a fixed alphabet denoted by Σ^*. For example, if $\Sigma=\{a\}$, then $\Sigma^* =\{\in, a, aa, aaa,\ldots\}$. If $\Sigma=\{0,1\}$, then $\Sigma^* =\{\in, 0,1, 00, 11, 01, 10, 000, \ldots\}$.

We will present permutations in the symmetric group S_n as strings in Σ_n^*, where $\Sigma_n = \{1,2,\ldots,n\}$.

We will define a binary relation $\rightarrow$ in S_n by writing $\pi \rightarrow \sigma$ whenever $\pi = xy$, and $\sigma = x^R y$, where $x, y \in \Sigma_n^*$ and x^R is the string x reversed (read backwards).

If π is a permutation, **f(π)** is the smallest k such that there exists a sequence of permutations

$$\pi = \pi_0 \rightarrow \pi_1 \rightarrow \cdots \rightarrow \pi_k \equiv e_n$$

where e_n=123 ···· n is the identity permutation. **f(n)** is the largest f(π) over all $\pi \in S_n$.

Let π be a permutation in S_n. $\pi(j)$ is the j^{th} number in π, where $1 \le j \le n$. If $|\pi(j) - \pi(j+1)| = 1$, we say that the pair $(j, j+1)$ is an ***adjacency*** in π. If $\pi = xby$, where $x, b, y \in \Sigma_n^*$ such that:

i) $(j, j+1)$ is an adjacency for $j = |x|+1, \ldots, |x|+|b|-1$, and

ii) b is a maximal with respect to this property (i.e., $|x|$, $|x|+1$) and ($|x|+|b|, |x|+|b|+1$) are not adjacencies),

then, b is called a ***block***.

If $\pi(j)$ is not a block (i.e., $(j-1, j)$ and $(j, j+1)$ are not adjacencies in π), then $\pi(j)$ is ***free***.

For the purpose of Gates and Papadimitriou's algorithm, we will consider $(j, j+1)$ to be an adjacency also if $\{\pi(j), \pi(j+1)\} = \{1, n\}$.

Gates and Papadimitriou's algorithm will sort the permutation π so as to create a total of $n-1$ adjacencies, that is, a block b of size n such as the ones shown in Fig. 2(*a*) and 2(*b*). These permutations can then be transformed to e_n via at most four flippings. (Figs. 2(*c*) and 2(*d*), respectively).

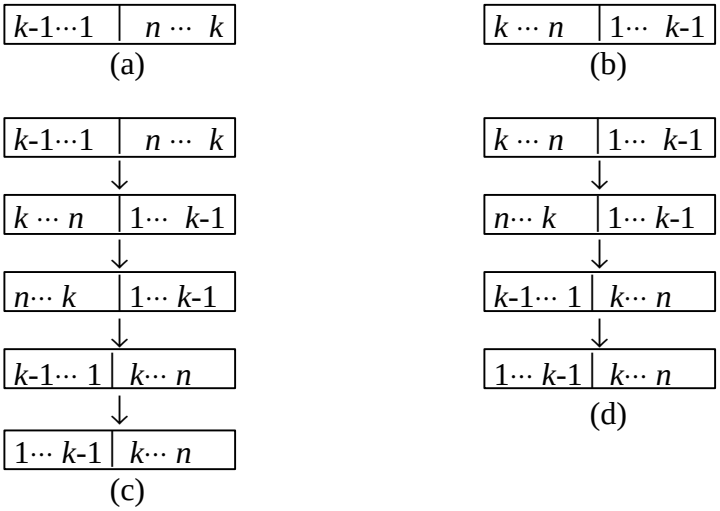

Fig. 2. (a) $\rightarrow e_n$ via the flippings in (c), (b) $\rightarrow e_n$ via the flippings in (d)

In the description of Gates and Papadimitriou's algorithm below, we use **o** to stand for one of {1, -1}. Addition is understood to be modulo *n*.

input: a permutation $\pi \in S_n$
output: a permutation σ with *n*-1 adjacencies

Set $\sigma = \pi$.
Repeat the following.

Let *t* be the first element of σ; i.e., $\sigma = t\sigma'$. (At least) one of the following eight cases applies. In each case, take the corresponding action.

1. *t* is free, and *t*+**o** is also free. Perform the flipping shown in Fig. 3(a).

2. *t* is free, and *t*+**o** is the first element of a block. Perform the flipping shown in Fig. 3(b).

3. *t* is free, but both *t*+1 and *t*-1 are the last elements of the blocks. Perform the sequence of flippings shown in Fig. 3(c).

4. *t* is in a block, and *t*+**o** is free. Perform the flipping shown in Fig. 3(d).

5. *t* is in a block, and *t*+**o** is the first element of a block. Perform the flipping shown in Fig. 3(e).

6. *t* is in a block with last element $t + k\mathbf{o}$ ($k>0$), *t*+**o** is the last element of another block and $t + (k+1)\mathbf{o}$ is free. Perform the sequence of flippings shown in Fig. 3(f) or 3(g), depending on the relative position of the two blocks and $t+(k+1)\mathbf{o}$.

7. *t* is in a block with last element $t+k\mathbf{o}$ ($k>0$), *t*+**o** is the last element of another block and $t+(k+1)\mathbf{o}$ is in a block. Perform the sequence of flippings shown in Fig. 3(h) or 3(k), depending on whether $t+(k+1)\mathbf{o}$ is at the beginning or the end of its block.

8. None of the above. σ has *n*-1 adjacencies; stop.

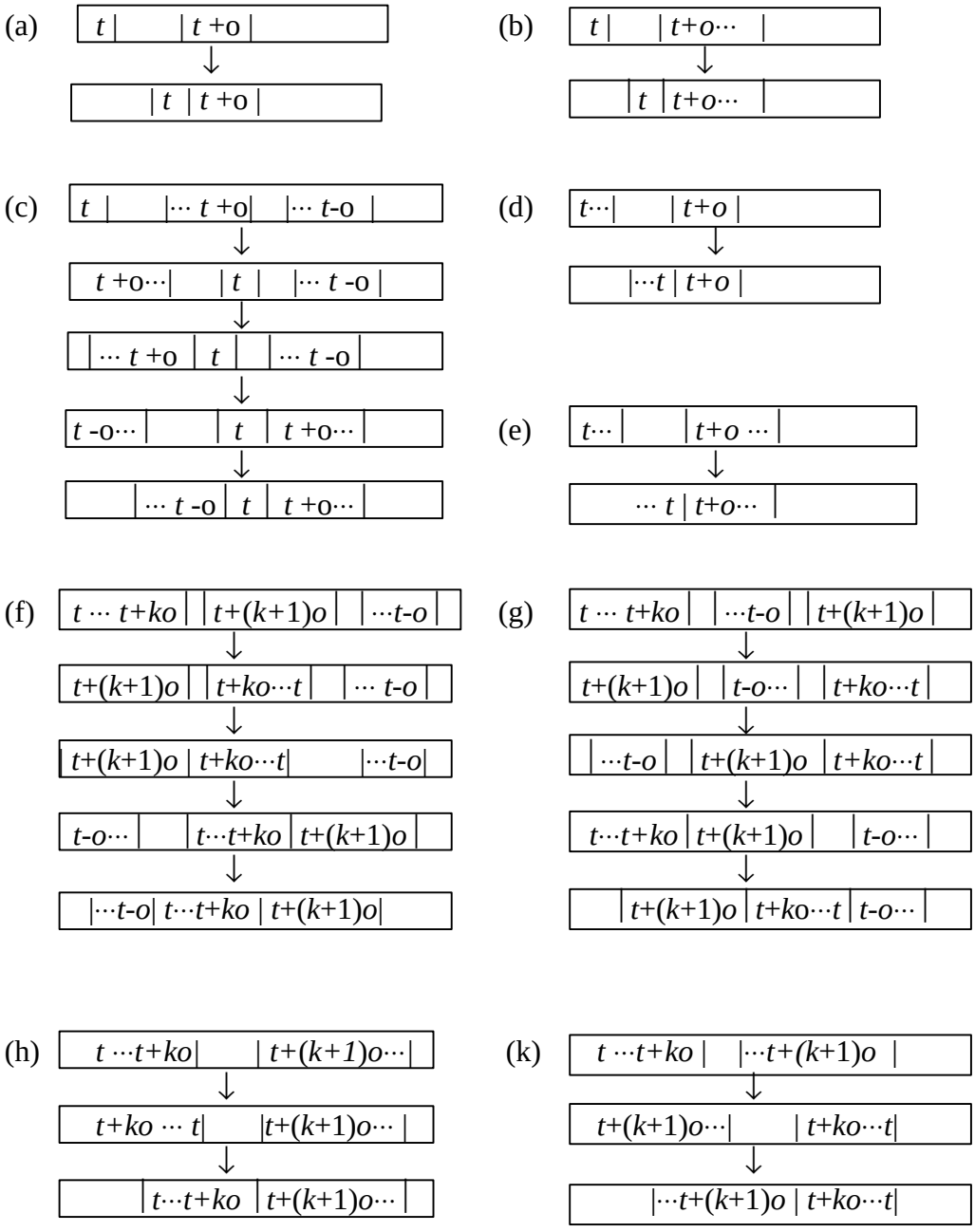

Fig. 3. Flippings for Gates and Papadimitriou's algorithm.

3. Results & Conclusions

We have written Gates and Papadimitriou's algorithm in a computer language to verify the known results of the pancake problem. However, computer verification of the results becomes unworkable for $n \geq 12$. The immediately limiting resource is actually the large space requirement of the data base file (dbf) as shown in Table 2. Here, actual computer work entailed two steps:

1. We generate all the permutations for each instance of n in dbf format.

2. We apply Gates and Papadimitriou's algorithm to each permutation. To resolve the required dbf size, say for $n=12$, we just had to partition the dbf into 11 smaller dbfs and run the algorithm on each one of these dbfs. The method is quite laborious but the algorithm becomes workable.

Table 2. Space requirement of the file for n = 11,12,13,14.

n	$n!$ (number of records in dbf)	Number of Bytes Required (size of dbf)
11	39,916,800	1,088,644,825
12	479,001,600	~15,552,068,928
13	6,227,020,800	~259,201,148,800
14*	87,178,291,200	~5,184,022,976,000

* computation in progress

Some directions which might be taken by interested reader include:

a) The design of algorithms on pancake graphs could use the algebraic properties of these graphs. Pancake graphs belong to the family of Cayley graphs.

b) Designing a branch-and-bound algorithm on the pancake graph.

c) Broadcast, message routing, binary tree simulation, and other parallel algorithms on the pancake graph.

References

[1] Harry Dweighter, American Mathematical Monthly, 82 (1) (1975) 1010.

[2] William H. Gates and Christos H. Papadimitriou, *Bounds For Sorting By Prefix Reversal*, Discrete Mathematics, 27 (1979), pp. 47-57.

[3] Mohammad H. Heydari and I. Hal Sudborough, *On the Diameter of the Pancake Network*, Journal of Algorithms, Vol. 25, No. 1, Oct 1997, pp. 67-94.