Parallel Algorithm Design

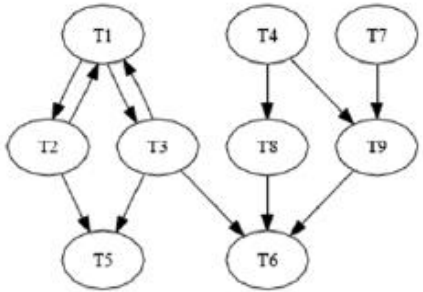
Quinn Chapter 3, GGKK Chapter 3

TASK AND CHANNEL MODEL

In this model, a parallel program is viewed as a collection of tasks that communicate by sending messages through channels.

An algorithm's data manipulation patterns can be represented as graphs: each vertex represents a data subset allocated to the same local memory, and each edge represents a computation involving two data sets.

An important goal of the parallel algorithm designer is to map the algorithm graph into the corresponding graph of the target machine's processor organization: this mapping is also called embedding.



TASKS AND CHANNELS

Task: Consists of an executable unit, together with its local memory and a collection of I/O ports.

- The local memory contains program code and private data.
- An access to a local memory is called local data access.
- The only way that a task can send copies of its local data to other tasks is through its output ports.
- Conversely, it can receive data from other tasks through its input ports.
- I/O port is an abstraction: it corresponds to some memory location that the task will use for sending or receiving data.
- Data sent or received through a channel is called non-local memory access.

A Channel is a message queue that connects one task's output port to another task's input port. A channel is reliable:

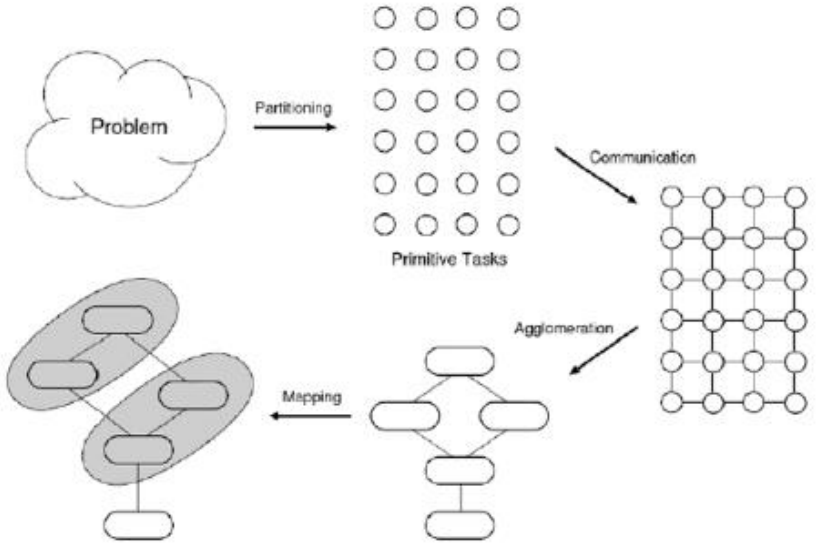
- Data values sent to the input port appear on the output port in the same order.
- No data values are lost and none are duplicated.

FOSTER'S DESIGN METHODOLOGY (1995)

4-stage design process:

- Partitioning: The process of dividing the computation and data into pieces.
- Communication: The process of determining how tasks will communicate with each other, distinguishing between local communication and global communication.
- Agglomeration: The process of grouping tasks into larger tasks to improve performance or simplify programming.
- Mapping: The process of assigning tasks to physical processors.

ILLUSTRATION



REDUCTION: A CASE STUDY

Recap: Given a set of n numbers, a1, …,an, reduction is the process of computing, op(a1,a2,…,an), where op is an associative operator.

- Many examples, like addition, multiplication, maximum, minimum, etc.

Partitioning: We have studied trivial cost-optimal solutions for the problem, by assigning one task to each number.

Note: If a cost-optimal CREW PRAM algorithm exists, and the way the PRAM processors interact through shared variables maps onto the target architecture, a PRAM algorithm is a reasonable starting point.

But now, we also need to consider the communications.

COMMUNICATION

There is no shared memory now in the computational model.

Our tasks must exchange data through messages.

To compute the sum of two numbers held by tasks T1 and T2, one must send its number to the other, which will then sum up.

When the task is finished the sum must be in a single task. This task will be called the root task.

A naïve solution would be for each task to send its value to the root task, which would then add all of them.

- Let λ denote the time for a task to send or receive a value from another task.
- Let χ denote the time for adding two numbers.

Thus, this algorithm would require time for n-1 additions in the root task, thus totaling $(n-1)\chi$. Additionally, there will be (n-1) receive operations by the root task, thus totalling $(n-1)\lambda$ for communication delay.

Total delay=$(n-1)(\lambda + \chi)$ which worse than a sequential algorithm.

A BETTER COMMUNICATION PATTERN

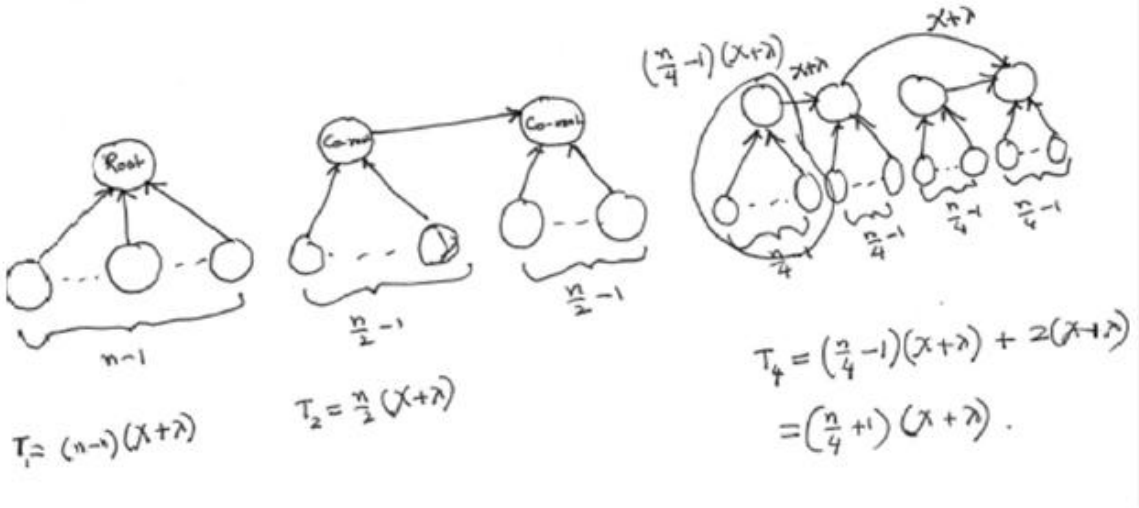
Imagine first we replace the single root task by two co-root tasks (assume n is even for simplicity).

Each co-root task will be sent (n/2-1) values will then add them up.

One of the co-roots will then communicate the result to the other, which will form the grand total.

Total time$=(\lambda+\chi)\left(\frac{n}{2}-1\right)+(\lambda+\chi)=\frac{n}{2}(\lambda+\chi)$

ILLUSTRATION OF THE PROCESS



EXTENDING THE STRATEGY

Assume $n=2^k$ for some integer k.

Let us denote the tasks as T0,T1,…,T(n-1).

The algorithm starts with the tasks $T_{n/2}$,$T_{n/2+1}$,…,T_{n-1} each sending its number to tasks T0,T1,…,$T_{n/2-1}$.

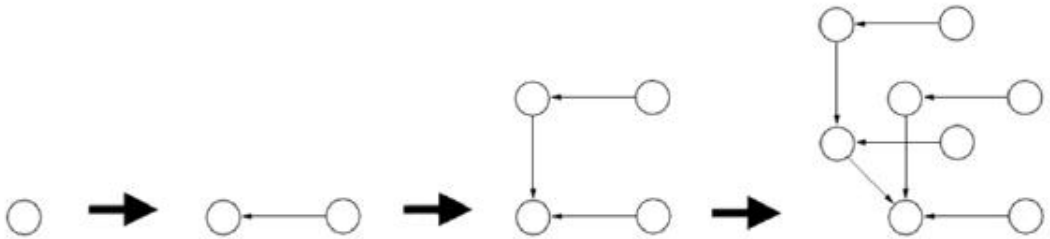
Each of them performs the sum in parallel.

Now we have exactly the same problem, but n is divided by two.

So, we repeat the logic: The upper half of the set of tasks T_0,…$T_{n/2-1}$ sends its number to the lower half, and each task in the lower half adds its pairs of numbers.

This sequence is repeated till n=1, at which point T_0 has the total.

PICTORIAL DESCRIPTION OF THE COMMUNICATION PATTERN



Such graphs are called as Binomial Trees.

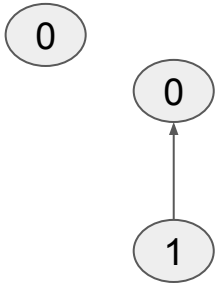
k=-1

0

BINOMIAL TREES

Recursive definition: Note that the tree of order k+1 (ie. No of nodes is 2^{k+1}) is obtained by cloning the tree of order k and labeling each node by adding 2^k to its old label.

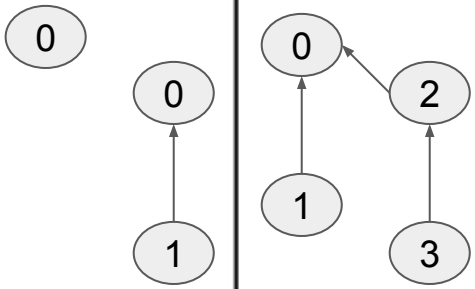
k=0



BINOMIAL TREES

Recursive definition: Note that the tree of order k+1 (ie. No of nodes is 2^{k+1}) is obtained by cloning the tree of order k and labeling each node by adding 2^k to its old label.

k=1

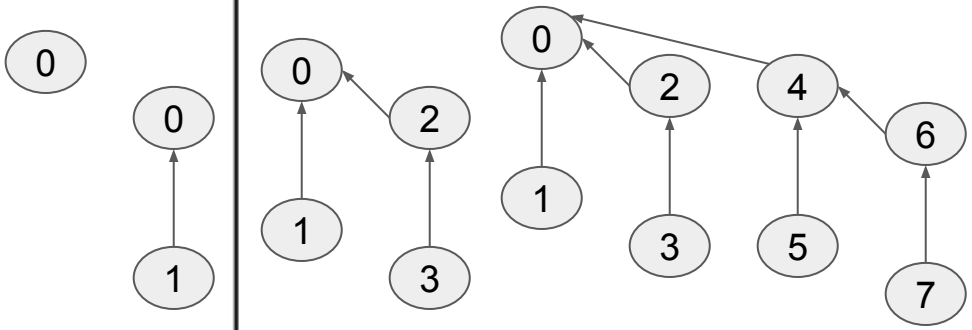


BINOMIAL TREES

Recursive definition: Note that the tree of order k+1 (ie. No of nodes is 2^{k+1}) is obtained by cloning the tree of order k and labeling each node by adding 2^k to its old label.

k=2

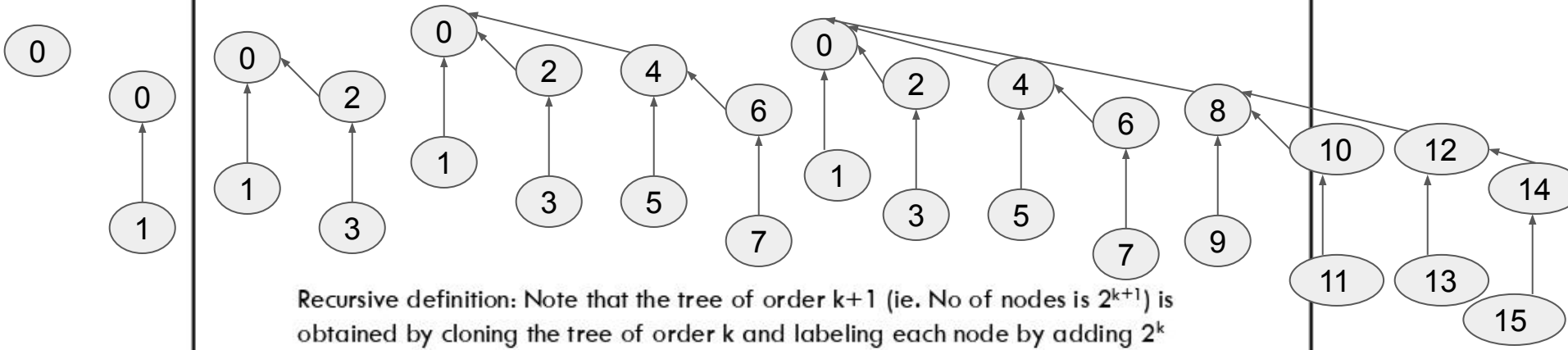
BINOMIAL TREES



Recursive definition: Note that the tree of order k+1 (ie. No of nodes is 2^{k+1}) is obtained by cloning the tree of order k and labeling each node by adding 2^k to its old label.

k=3

BINOMIAL TREES



Recursive definition: Note that the tree of order k+1 (ie. No of nodes is 2^{k+1}) is obtained by cloning the tree of order k and labeling each node by adding 2^k to its old label.

BINOMIAL TREES

A Binomial Tree B_n of order $n \geq 0$ is a rooted tree such that, if $n=0$, B_0 is a single node called the root.

If $n \geq 0$, B_n is obtained by taking two disjoint copies of B_{n-1} and joining their roots by an edge, then taking the first copy to be the root of B_n.

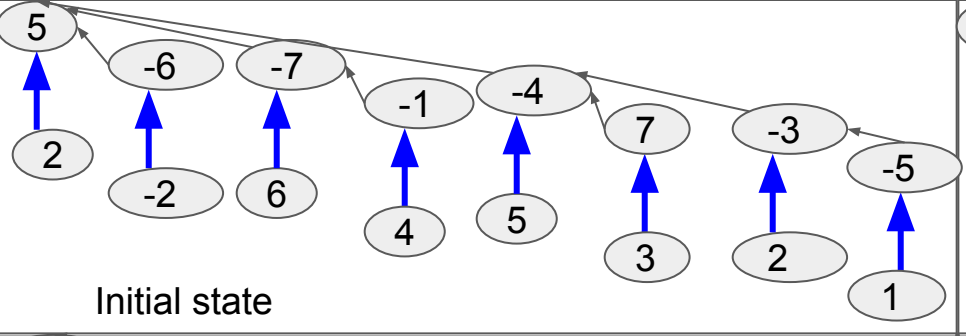
A binomial tree of order n has $N=2^n$ roots and 2_n-1 edges.

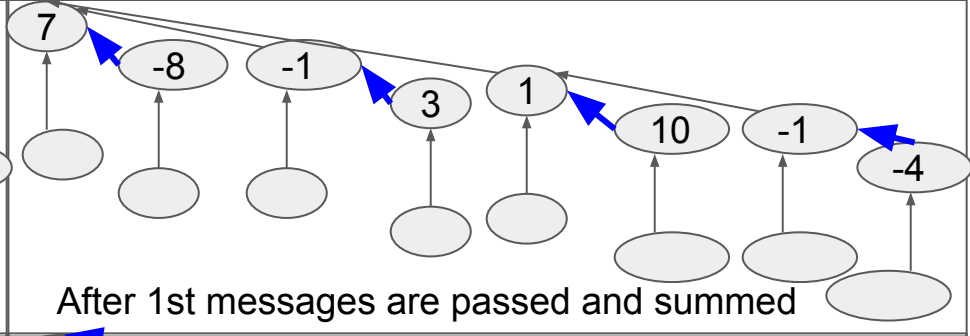
Each node (except the root) has exactly one outgoing edge.

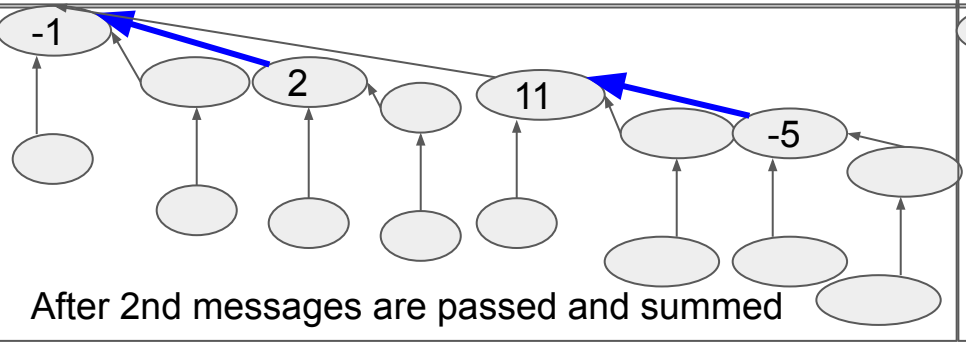
The maximum distance from any node to the root of the tree is n, ie $\log_2 N$.

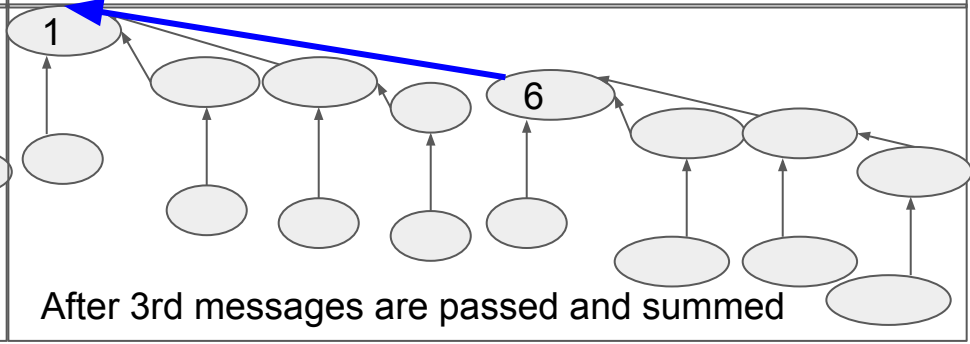
- This means a parallel reduction can always be performed with at most $\log_2 N$ communication steps.

The number of leaves is 2^{n-1}.

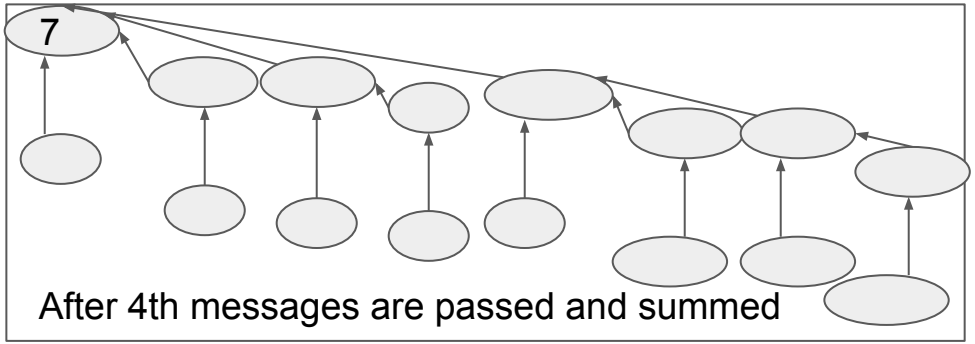








Parallel reduction of 16 numbers



AGGLOMERATION

It is likely the number of processors p will be much smaller than the number of tasks n in any realistic problem.

We agglomerate tasks also to reduce the number of communications.

We agglomerate so that the resultant graph still remains a binomial tree.

Thus this improves the efficiency of the implementation.

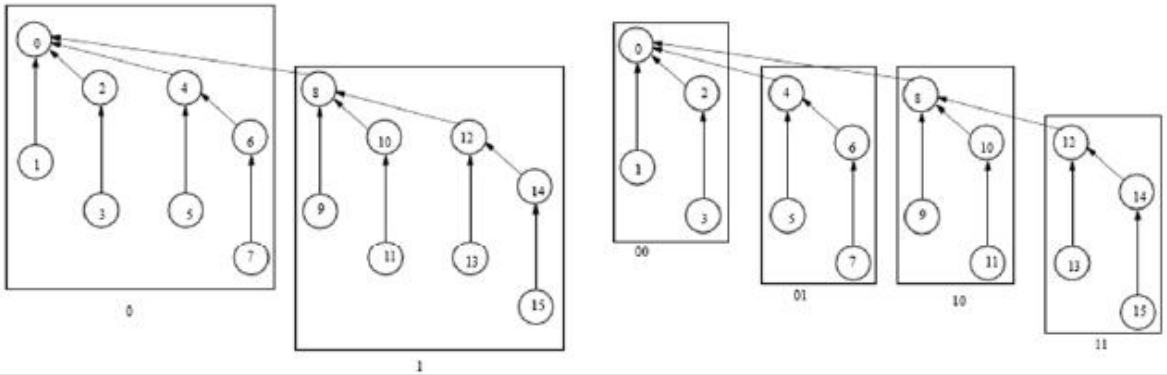
AGGLOMERATION OF THE BINOMIAL TREE

Assume $p=2^m$, $n=2^k$, $m<=k$

We number (label) the binomial tree, the node labels being k bits long, such that they can be partitioned in the following way:

All nodes whose label's upper m bits are the same will be agglomerated into a single task.

For example, if $p=2^1$, then all nodes whose upper bit is 0 are in one task, while those whose upper bit is 1 is other.

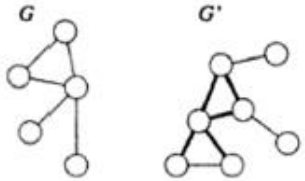


MAPPING/EMBEDDING

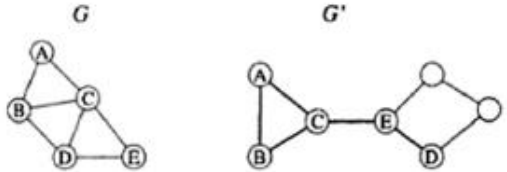
Embedding of a graph G=(V,E) into a graph G'=(V',E') is a function ϕ from V to V'.

Let ϕ be a function that embeds a graph G into a graph G'. The dilation of the embedding is defined as: $dil(\phi) = \max\{dist(\phi(u), \phi(v)) | (u, v) \in E\}$ where dist(a,b) is the distance between a and b in G'.

Dilation-1 embeddings are desirable: as communication time is roughly proportional to the length of the path between processors.



A Dilation 1 embedding



A Dilation 3 embedding

EMBEDDING BINOMIAL TREE TO HYPERCUBE

A graph G is called cubical if there is a dilation-1 embedding of G into a hypercube.

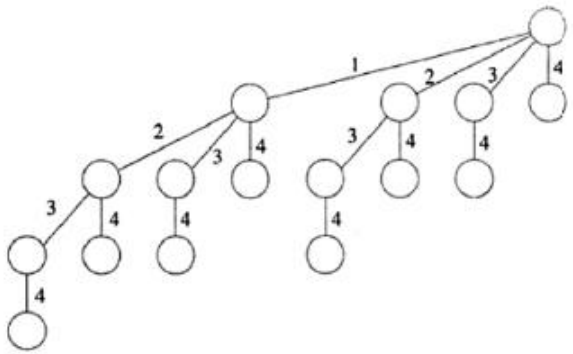
The problem of determining whether an arbitrary graph G is cubical is NP-complete.

A dilation-1 embedding of a connected graph G into a hypercube with n nodes exist iff it is possible to label the edges of G with the integers {1,2,…,n} st:

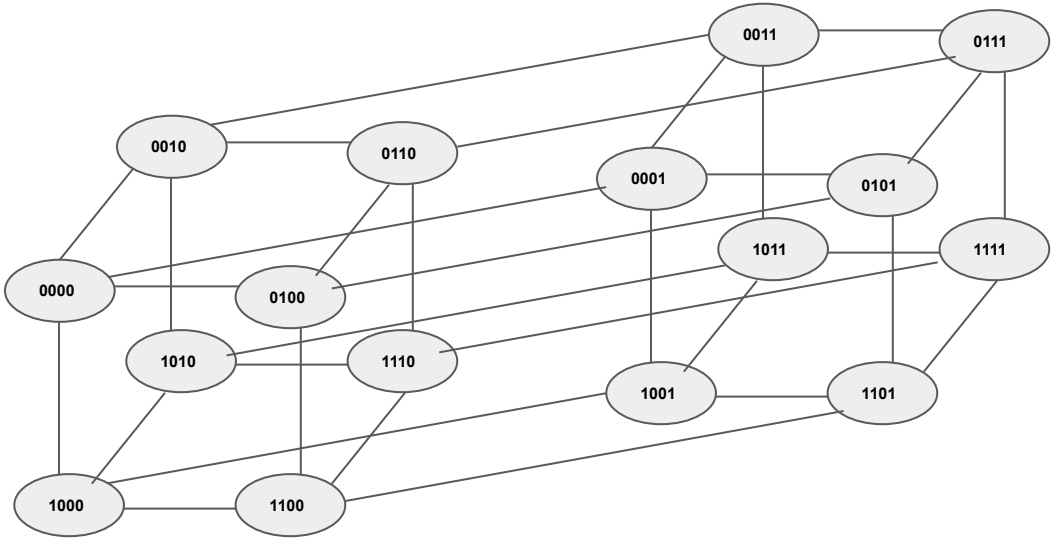
- 1. Edges incident with a common vertex have different labels
- 2. In every path of G at least one label appears an odd number of times.
- 3. In every cycle of G no label appears an odd number of times.

EMBEDDING BINOMIAL TREE TO HYPERCUBE

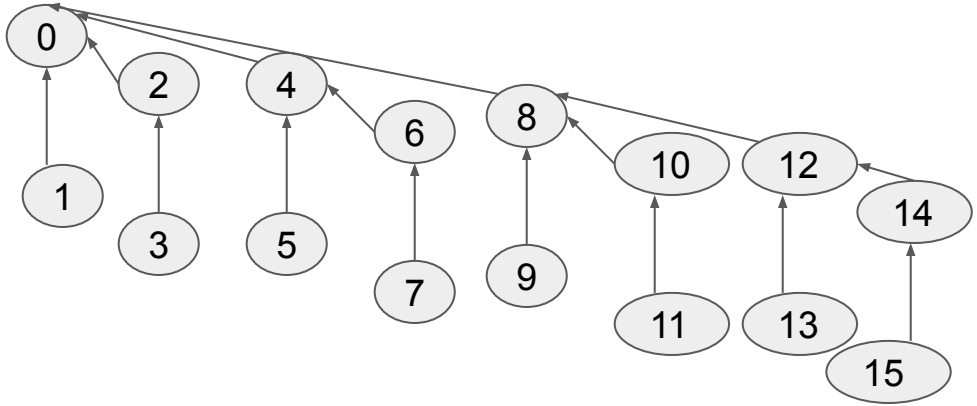
A binomial tree of height n can be embedded in a hypercube of dimension n such that the dilation is 1.

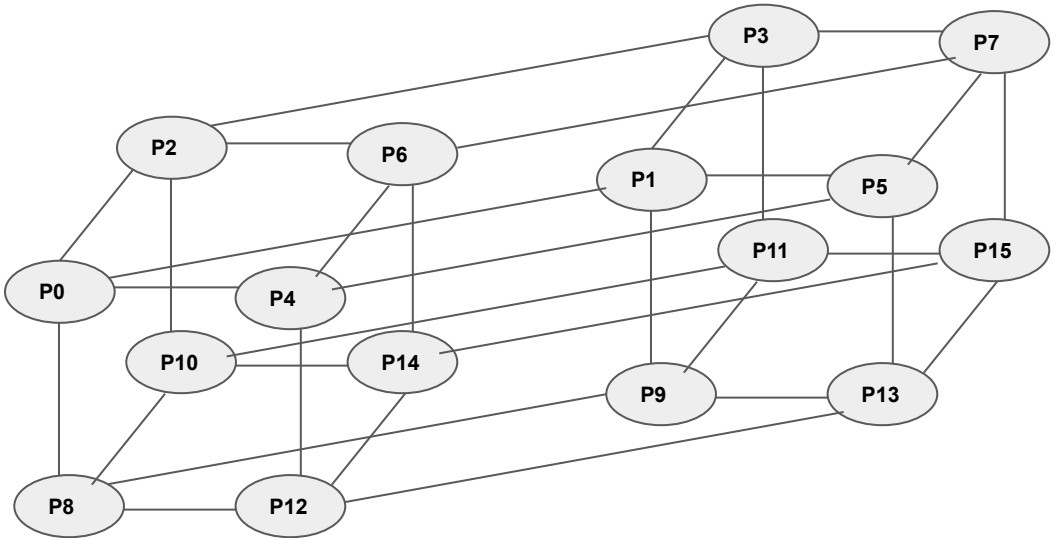


What was a hypercube?

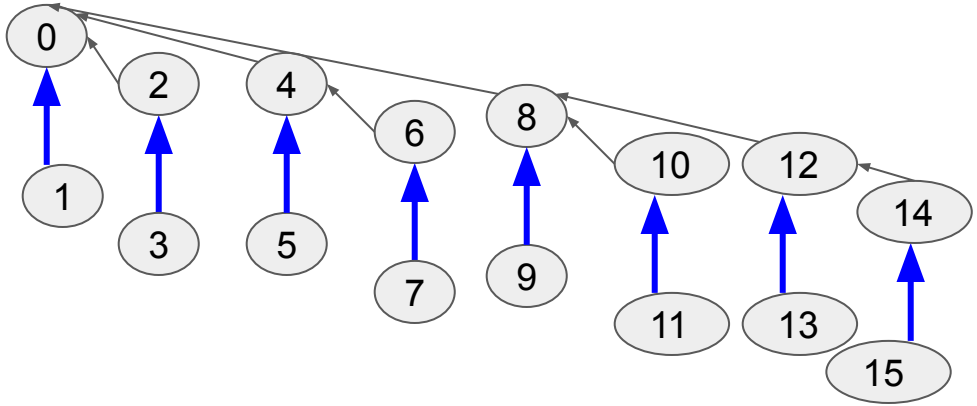


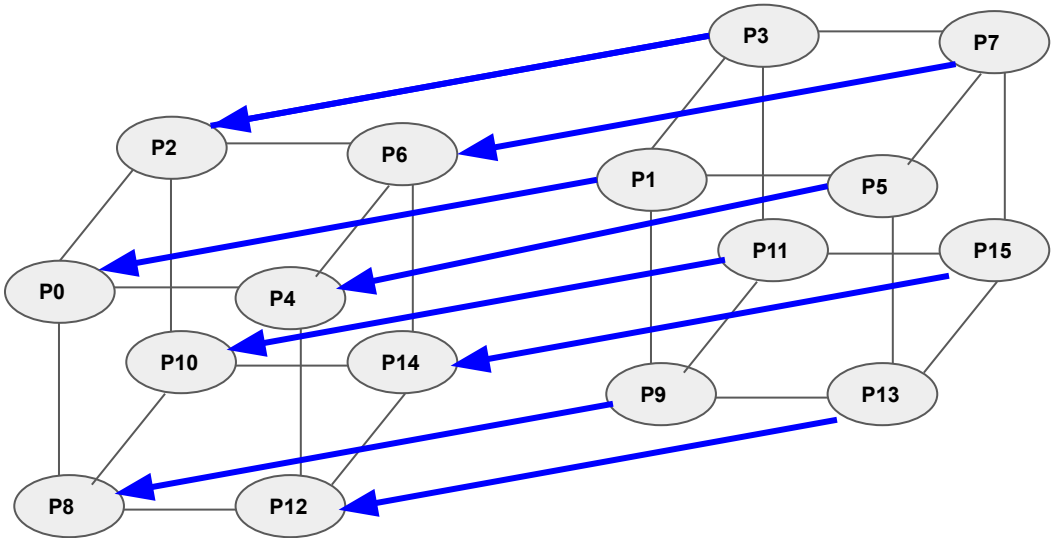
How to embed a binomial tree computation in a hypercube?



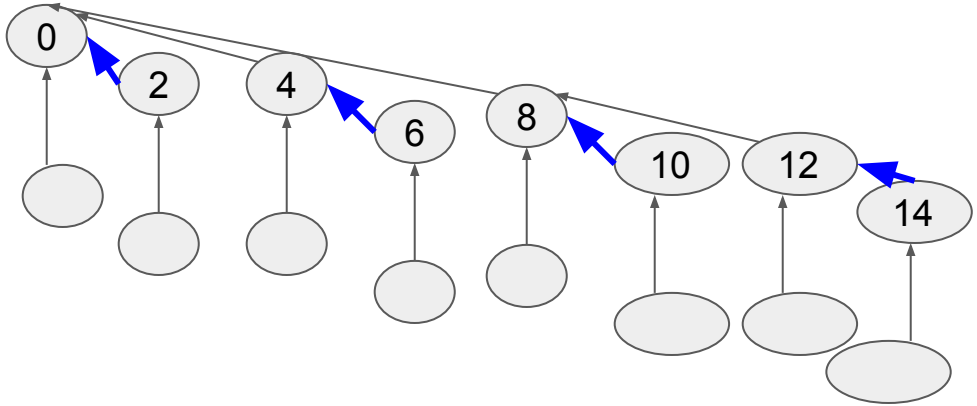


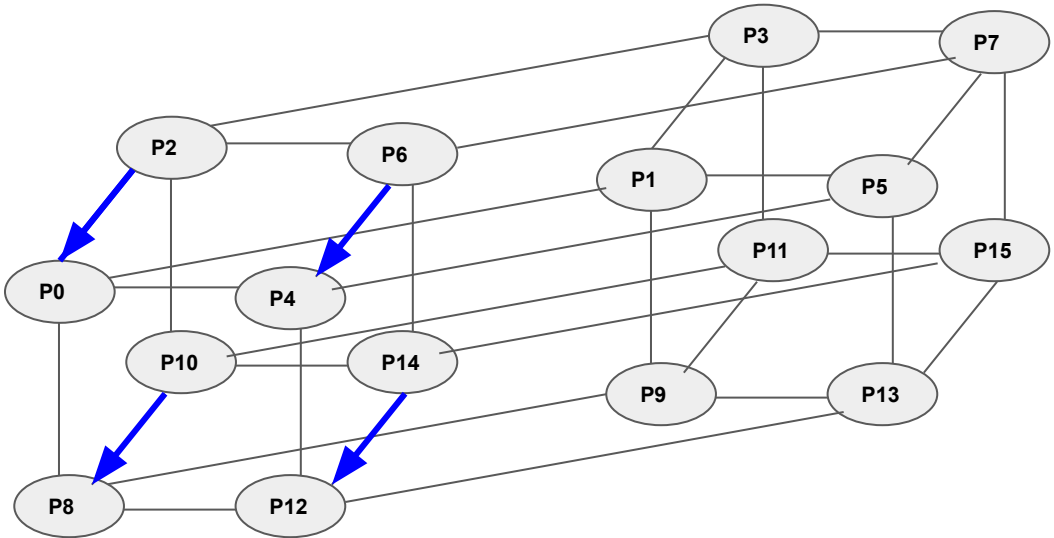
1st messages are passed and summed



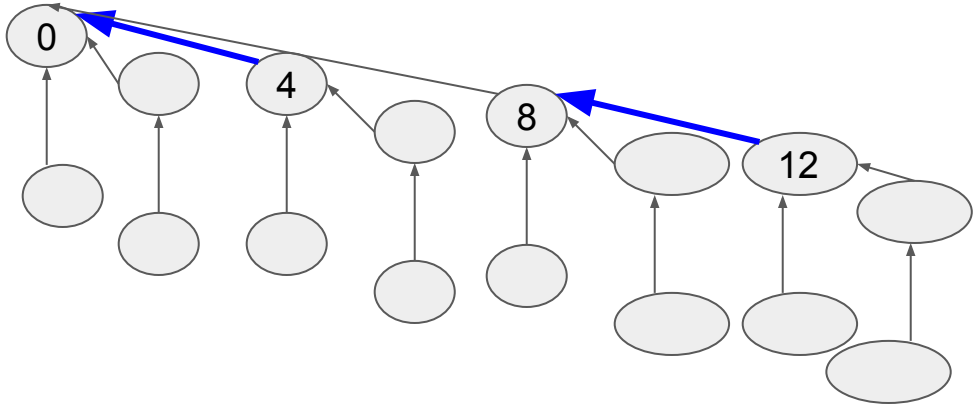


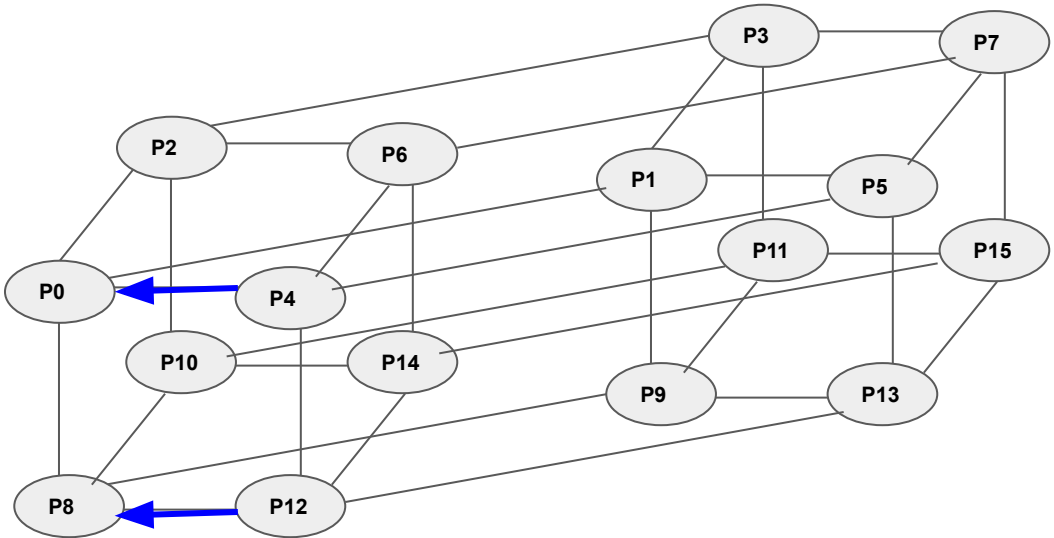
2nd messages are passed and summed



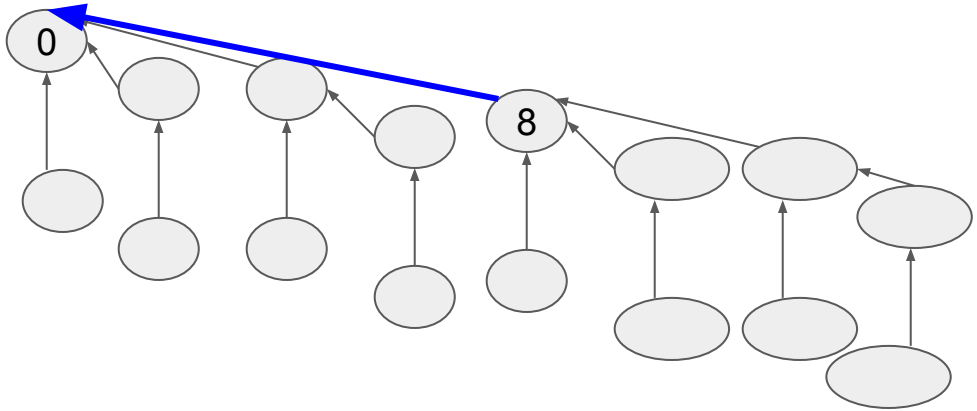


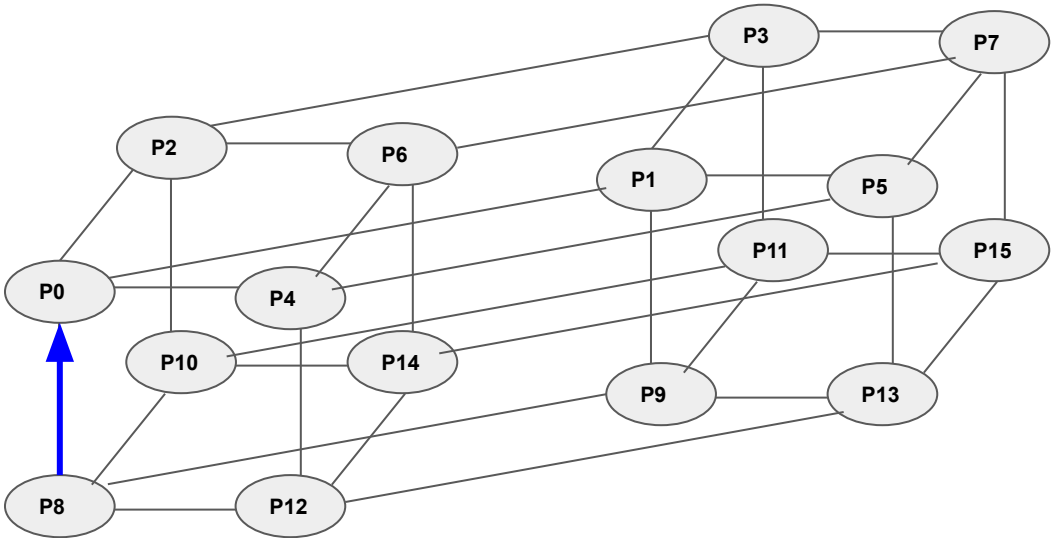
3rd messages are passed and summed





4th messages are passed and summed





ANALYSIS

Run Time depends on 2 parameters: χ and λ

Performing the sequential sum of (n/P) numbers assigned to each task=$\left(\frac{n}{p}-1\right)\chi$

The parallel reduction takes log p steps.

Each process must receive a value and then add to its partial sum.

Thus each step takes $(\lambda+\chi)$ time.

Thus total time, $T=\left(\frac{n}{p}-1\right)\chi+(\lambda+\chi)\log p$