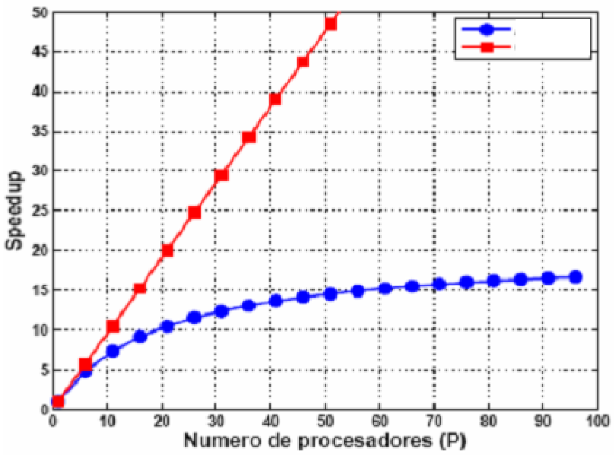
Exercise 1.- The figure shows the speedup achieved with P processors when executing a weather forecasting calculation. In one case the execution time is reduced while maintaining the accuracy and in the other case the execution time is maintained while improving the accuracy.



- Explain under which conditions each of the Speedup behaviors represented in the graph occurs.

  Squares: fixed time speed-up, workload increases with the number of processors (Gustafson's Law)
  Circles: Speed-up of fixed workload, Execution time decreases with the number of processors but is limited by the serial work (Amdahl's Law).

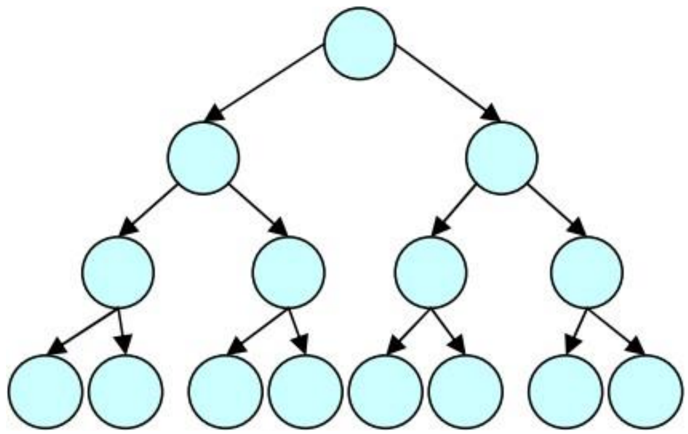
- Formulate the value of Speedup that corresponds to Amdahl's law and Gustafson's law. AmdahlS = p/(1+fs*(p-1)) if p very large S = 1/fs
  Gustafson S= p + (1-p)*fs
- Identify in the graph which is each case and detail what each parameter represents in both formulations.
  S = Speed-up or acceleration = serial time/parallel time. Number of processors is p
  The fraction of work done in series is fs
  In Amdahl, the parallel working time on one processor is (1-fs) and if it is done with p processors it is (1-fs)/p
  In Gustafson the parallel time with p processors is (1-fs) and if done on one processor it is p*(1-fs).

From the approximate SpeedUp values obtained from the graph calculate serial fraction, fs, of the corresponding problem for P = 50 in each case and extrapolate the performance for P = 200. Calculate the behavior of the serial fraction and the acceleration obtained when P is very large for each of the cases.

Plot circles(amdahl) S with large p is 18 S = 1/fs , fs = 1/18 = 0.056 fs = 5.6% for all P For p = 200; S=200/(1+199*0.056) = 200/12.1= 16.5
From the square plot S=49 with p =50
49 = 50 + (1-50)*fs; 1=49fs , fs = 1/49 = 0.02 fs=2% for P=50
The behavior of the serial fraction(fs),

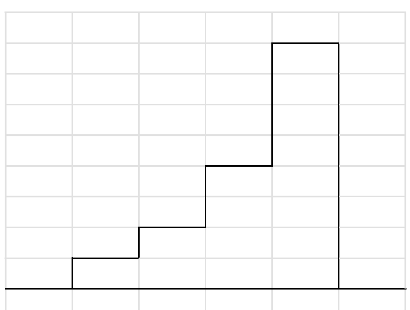
- In fixed workload problems (Amdahl), increasing the number of processors fs remains constant.
- for fixed time problems (Gustafson) by increasing the number of processors fs => 0.

The figure represents the dependency network of an application to be executed in a multiprocessor system with p processors. We want to generalize by calling N the number of nodes in the network and n a power of 2, so that there is N = summation from 1 to n of 2^(n-1) nodes. In the figure it is represented for n=4.



It is assumed that there is no penalty for communications and that each node represents a task that takes one unit of time and cannot be divided into subtasks of lower granularity.

a. Plot the parallelism profile during the execution of the application for the number of nodes represented (n=4).



b. The maximum degree of parallelism and the work performed with maximum degree of parallelism.

DOP max =8W8 = 8

c. Calculate the speedup (S) and efficiency (E) as a function of the number of processors S(p) and compare for this case the values S(), S(p=4) and S(p=8).

S(oo) = S (8) = (1+2+4+8)/ 1+1+1+1+1= 15/4 = 4.5 | E(8) =15/4*8=0.46 S(4) = (1+2+4+8)/ 1+1+1+2= 15/5 = 3 |E(4) =3/4 = 0.75

d. Express the maximum degree of parallelism and the maximum Speedup as a function of n.

DOP max = 2^(n-1) |S (oo) =ceil((n+1)/2)

Given the following loop, indicate how to execute it in parallel in the most efficient way possible. Write the resulting code.

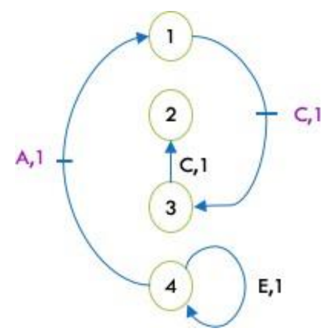
```
do i = 2, 1001
     (1) A(i) = C(i) - 10
     (2) B(i+1) = C(i-2) + alpha
     (3) C(i-1) = D(i+1) * 3
     (4) E(i) = E(i-1) + A(i+1)
enddo
```

Note: alpha is a constant value

a. We are asked to analyze the dependencies between the instructions and generate the dependency graph and the iteration space.

```
A(2) = C(2) - 10
B(3) = C(0) +
alpha C(1) = D(3)
* 3 E(2) = E(1) +
A(3)


A(3) = C(3) - 10
B(4) = C(1) +
alpha C(2) = D(4)
* 3 E(3) = E(2) +
A(4)
```



b. Write the code (pseudocode) resulting from the most efficient parallelization possible.

```
do i = 2, 1001
      (4) E(i) = E(i-1) + A(i+1)
enddo

C(1) = D(3) * 3
B(3) = C(0) + alpha
B(4) = C(1) + alpha

do i = 2, 999
     (1) A(i) = C(i) - 10
     (3) C(i) = D(i+2) * 3
     (2) B(i+3) = C(i) + alpha
enddo

A(1000) = C(1000) - 10
A(1001) = C(1001) - 10
C(1001) = D(1002) * 3
```

c. If the execution time of each instruction is T, make an estimate of the speedup factor and efficiency to be achieved by using 100 processors.

Time series. 1000 iterations of 4 insructions Ts=
4000 T

Parallel time, sum: 1000 iterations of 1 instruction in series, 1000T
6 T of prologue and epilogue + 998 iterations of 3 instructions

Tp = 1000T + 6T + 998x3T/100 =1006T+2994T/100=1036T

S=4000T/1036T =3.86

d. What is the maximum achievable acceleration with as many processors as needed? Justify the results.

As fs=0.25=1/4S (infinity) = 1/fs =4it is also worth 4000T/1009T= 3.96