# Partition

- Partition is another simple technique.
- It consists of breaking the problem into independent subproblems of almost equal sizes and then solving these subproblems concurrently.
- Consider the problem of merging two sorted sets.
- Let

$$\begin{aligned} \mathtt{X} &= \{-5, 14, 34, 89\} \\ \mathtt{Y} &= \{-1, 24, 77, 82\} \end{aligned}$$

- Let $\mathtt{rank}(a\colon \mathtt{X})$ be rank of element $a$ in $\mathtt{X}$.
- Define $\mathtt{rank}(a\colon \mathtt{Y})$ like-wise.
- Then $\mathtt{rank}(a\colon \mathtt{X}\cup\mathtt{Y})$ gives the position of $a$ in the merged set.



# Partition

- In the example, the cross ranks and full ranks are:

| Element e | rank(e: X) | rank(e: Y) | rank(e: X∪Y) |
|---|---|---|---|
| -5 | 0 | 0 | 0 |
| 14 | 1 | 1 | 2 |
| 34 | 2 | 2 | 4 |
| 89 | 3 | 4 | 7 |
| -1 | 1 | 0 | 1 |
| 24 | 2 | 1 | 3 |
| 77 | 3 | 2 | 5 |
| 82 | 3 | 3 | 6 |

- So the merged list is:

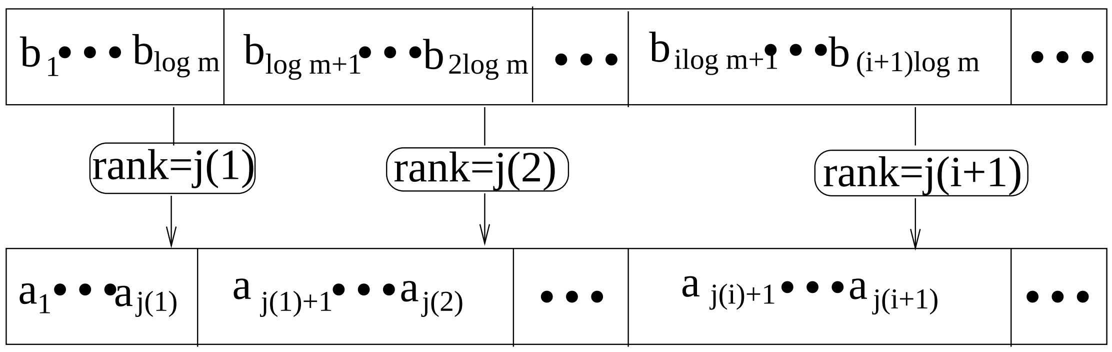
$$\{-5, -1, 14, 24, 34, 77, 82, 89\}.$$

# Optimal Merging

- Optimal merging can be realised as follows:
  - Choose $n/\log n$ elements of each of $A$ and $B$ that partitions these sets into blocks of equal length each.
  - Apply binary search to rank each of the chosen element in the other sequence.
- By this partitioning we get two subsequences of length O($\log n$) each.
- The time to merge the subsequences is O($\log n$) in one processor.
- So, in all O($n/\log n$) processors can perform the merging in O($\log n$) parallel time.
- Since total of each of the two stages of merging algorithm is O($\log n$) with processor requirement of $n/\log n$, the method is optimal.

# Optimal Merging

- Let the rank of element $b_{i \log m}$ in the array $A$ be $j(i)$ for $1 \leq i \leq m/\log m$.
- It means all elements of $A$ after position $j(i)+1$ are larger than $B[(i-1)\log m + 1], \ldots, B[i \log m]$.
- So merging $A[j(i-1)+1] \ldots A[j(i)]$ and $B[(i-1)\log m+1] \ldots B[i \log m]$ can be done independently.
- These results of independently merged subsequences can be concatenated to produce the final merged result.





# Optimal Merging

```
partitioning() {
    shared var k_m, j[.], A[.];
    shared var B[.], A_i[.], B_i[.];
    j[0] = 0;
    j[k_m] = n;
    for all (P_i: 1 ≤ i ≤ k_m-1) {
        j[i] = rank(B[i log m]: A);
        B_i = {B[i log m + 1]...B[(i + 1) log m]}
        if (j[i]≠ j[i+1])
            A_i = {A[j[i]+1]...A[j[i+1]]}
        else
            A_i = Φ;
    }
}
```

# Performance Measure

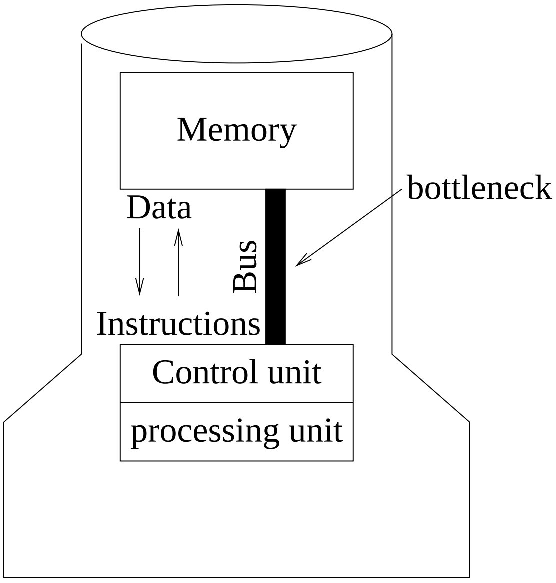
- Herb Grosch's law states that to sell a computer for twice as much the speed must be 4 times as fast.
- The reasons are:
  - Rapid advancement of technology
  - Reluctance of buyer to go for new h/w
- It forces both buyer and seller to use the performance and the price ratio as suggested by Grosch as thumb rule.
- But according to the mainframe computer economics: twice as much h/w will actually cost twice as much.
- So h/w designer need to come up with entirely new design to extract four-fold performance with only 2 times of cost on existing computers.



# Performance Measures

- The problem of performance is the following bottleneck.




- Only way to extract performance is to remove the bottleneck.

# Amdhal's Law

- Why it took so much time to build a parallel computer?
- Gene Amdhal an IBM designer came up with arguments that the cooperation needed to combine solutions of subproblems that is inherent in design of parallel programs is inherently sequential.
- The overhead places an upper limit on throughput.
- So there is no way to overcome the sequential bottleneck.
- Amdhal's law can be stated in terms of a simple formula as follows.



# Amdhal's Law

- Suppose a parallel computer has $N$ interconnected processors.
- Let the sequential part of computation in a parallel program be $\beta$ fraction of the whole.
- Speedup $S$ is achieved by running the program on parallel computer is

$$S = T(1)/T(N)$$

  where $T(1)$ is the running time on sequential computer.
- Note that parallel computer will require time of $T(1)\beta$ to compute the serial part.
- But it would requires $T(1)(1-\beta)/N$ time for the parallel part.
- That is:

$$T(N) = T(1)\beta + \frac{T(1)(1-\beta)}{N}$$

# Example: Amdhal's Law

- So
$$S = \frac{1}{\beta + \frac{(1-\beta)}{N}} = \frac{N}{\beta N + (1-\beta)}$$
- Let $\beta = 0.67$ in a parallel program.
- Then on a parallel computer using 10 processor the speedup is:
$$S = \frac{1}{[0.67 + \frac{.33}{10}]} = 1.42$$
- John Gustafson and Ed Barsis used 1024 processors in nCUBE/10 machine and were successful in achieving a speedup of 1000 on a problem that had $\beta$ ranging between 0.004 to 0.008.
- According to Amdhal's law the speedup is predicted to be around 125 to 150.
- Gustafson and Barsis explained that the implicit assumption in Amdhal's law that (1-$\beta$) independent of $N$ is wrong.

# Gustason Barsis Law

- The idea behind this law is parallelism can be used to increase parallel size of the problem.
- If one processor is used then it would compute both parallel and serial part.

$$T(1) = \beta + (1 - \beta)N$$

- But if $N$ parallel processor is used then the problem can be scaled up so all $N$ processors compute both parallel and the serial part, one followed by the other.
- Adding the two part

$$T(N) = \beta + (1 - \beta) = 1$$

- Thus the speedup is

$$S = N - (N - 1)\beta$$

- For the earlier example the speedup is 3.97 according to this law.

# Distributed Memory Machines

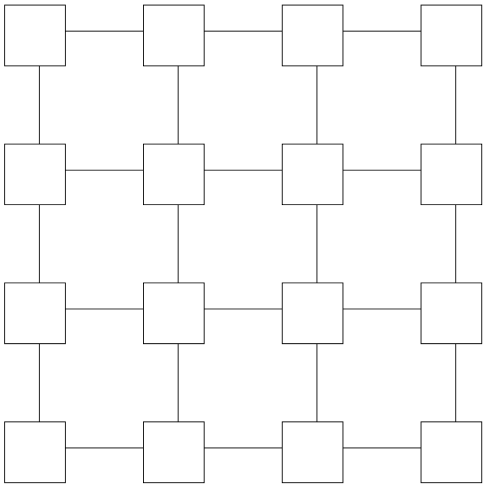
- We now consider some of the practical models for parallel computing.
- The communication links and the processors together can be conveniently represented by a graph.
- There are 4 important parameters of the representative graph:
  - Diameter – must be low.
  - Bisection width – must be large.
  - Degree or links of node – must be constant.
  - Maximum link length – must be constant.
- These parameter are useful in designing algorithms so that communication overheads are minimized.



# Mesh Networks

- Arranged in a $q$-D lattice.
- Each interior processor can communicate with $2q$ other processor.
- A 2-D mesh is shown below.



# Mesh Network

- Assuming each dimension is of size $k$ such a $q$-D mesh has $k^q$ processors and the mesh does not have wrap around connection.
- Diameter of such a mesh is equal to number of links from one corner to the diagonally opposite corner.
- This would require traversal of $q$ sides of the mesh.
- As there are $k$ processor in each side, the number of links to be traversed is $k - 1$ on each.
- So diameter is $q(k - 1)$.
- In order to break the mesh into 2 meshes of same size, $k$ links should be removed on each the $q-1$ planes.
- So the bisection width is $k^{q-1}$.
- Each interior processor is linked to $2q$ other neighbouring processors as $A[i_1, i_2, \ldots, i_q]$ should be connected to $A[\ldots, i_j \pm 1, \ldots]$

