1 a) There are 4 terms, C22 = sum of:

* C1 =
* C2 =
* C3 =
* C4 =

Since there are 4 terms, there are 4!/2 = 24/2 ways to sum up the term sequentially. E.g((C1+C2)+C3)+C4 divided by 2 since ((C2+C1)+C3)+C4 , is the same due to commutative law

For summing two terms up before combining them, we have (4Combination2)/2=3, there are 6 combination for summing up two terms twice, e.g (C1+C2)+(C3+C4), and since it is commutative, e.g has the same result as (C3+C4)+(C1+C2), the result is divided by 2.

Total number of combinations: 12+3=15, ignoring repeated pattern due to commutative law

1b)

|  |  |
| --- | --- |
| **Sequence** | **Result** |
| (((C1+C2)+C3)+C4) | 0.0 |
| (((C1+C2)+C4)+C3) | -1.0 |
| (((C1+C3)+C2)+C4) | 0.0 |
| (((C1+C3)+C4)+C2) | 2.0 |
| (((C1+C4)+C2)+C3) | 1.0 |
| (((C1+C4)+C3)+C2) | 1.0 |
| (((C2+C3)+C1)+C4) | 0.0 |
| (((C2+C3)+C4)+C1) | 0.0 |
| (((C2+C4)+C1)+C3) | 1.0 |
| (((C2+C4)+C3)+C1) | 0.0 |
| (((C3+C4)+C1)+C2) | 2.0 |
| (((C3+C4)+C2)+C1) | 2.0 |
| ((C1+C2)+(C3+C4)) | 0.0 |
| ((C1+C3)+(C2+C4)) | 2.0 |
| ((C1+C4)+(C2+C3)) | 1.0 |

1c)

1d)

For this particular problem, we can:

* Compute R1 = C1+ C4 on one process R2 = C2 + C3 on another process
* Compute R1 = C4 + C1 on one process R2 = C2 + C3 on another process
* Compute R1 = C1 + C4 on one process R2 = C3 + C2 on another process
* Compute R1 = C4 + C1 on one process R2 = C3 + C2 on another process

Before combining the result of R1 and R2.

The four ways that yielded the correct result for this operations is:

* (((C1+C4)+C2)+C3)
* (((C1+C4)+C3)+C2)
* (((C2+C4)+C1)+C3)
* ((C1+C4)+(C2+C3))

In general, this problem can be minimized by

* using integer scaling and count square, upperbound/lowerbound.
* Or simply using integer.
* Or possibly rounding up until Most Few Significant bits (would result in loss of precision for large number)

2a)

j = start(i) = floor(N\*i/P)

k = length(i) = floor(N \* (i+1) /P) – floor(N \* i /P)

where arraysize N = 30 and number of processor P = 7 for processor P*i*

2b)

|  |  |  |
| --- | --- | --- |
| **i** | **j** | **k** |
| 0 | 0 | 4 |
| 1 | 4 | 4 |
| 2 | 8 | 4 |
| 3 | 12 | 5 |
| 4 | 17 | 4 |
| 5 | 21 | 4 |
| 6 | 25 | 5 |

2c)

Time taken processing 1 number = T

Time taken 1 Processor only = 30T

Time taken 7 Processor = MAX(k) \* T = 5T

Speed Up = 30T/5T = 6

3)

*t*(*N*, *1*) = 0.1 + 10–7 (*N*/*P*) lg(*N*/*P*) seconds + 5×10-7*N* lg(*P*) = 0.1 + 10–7 (*N*) lg(*N*)

Now, solve for N when:

t(N,1) > t(N,2) = 0.1 + 10^-7 (N/2)lg(N/2) + 5\*10^-7Nlg(2)

We get:

2N\*lgN > 9N + Nlg(N)

2lgN > 9 + lgN

lgN > 9