**ArSeqProduct — Arithmetic Sequence Product (Draft)**

**Status:** Draft. For internal review only (do not publish yet).

Compute the exact product of a finite arithmetic progression (AP)

P(a,d,n)=∏k=0n(a+k d),P(a,d,n) = \prod\_{k=0}^{n} (a + k\,d),

using **pairing by difference-of-squares** and **balanced product trees**, with optional **stride‑based parallelism**. Factorial is the special case a=1, d=1, n=999 for 1000!.

* **a** — begin (first term)
* **d** — difference (step)
* **n** — number of differences (there are n+1 terms)
* **p** — workers to use (≤ logical processors)
* **t** — log interval in **minutes** (0 means "no periodic log")

All parameter names are **one letter**, **case‑insensitive** (e.g., A=... is the same as a=...). Values are **positive numbers**. Whitespace is allowed.

Multiple runs can be issued in one command by separating parameter groups with a semicolon ;. **Parameters persist** across runs: a parameter keeps its previous value if not supplied again.

**Algorithm (one paragraph)**

Each worker handles a residue class modulo the **stride** s = d \* p, multiplying an **odd block** of L = 2h+1 terms centered at m = a + d\*t + h\*s. Pairs (m−k s)(m+k s) collapse to m² − (k s)², cutting the **number of big multiplications roughly in half** compared to naïve multiply‑left‑to‑right. Partial results are combined with a **balanced product tree** to keep operand sizes similar.

**Command‑line usage**

ArSeqProduct [parameter-list]

Where **parameter-list** is a sequence of name=value pairs separated by spaces or commas, and **runs** are separated by semicolons ;.

**Recognized parameters:** a=, d=, n=, p=, t=

**Initial defaults (when first omitted):**

* a=1, d=1, n=0 (i.e., product has one term a)
* p = min(Environment.ProcessorCount, max supported), chosen by the app
* t=0 (no periodic log)

**Persistence across runs:** after a semicolon, any missing parameter reuses the previous run’s value.

**Shell notes**

* **Windows CMD:** semicolon is safe; quoting is optional.
* **PowerShell / Bash / Zsh:** **quote the whole parameter string** so ; is not treated as a command separator.

**Examples**

**1) Factorial comparisons (easy sanity checks)**

* **PowerShell / Bash / Zsh**
* ./ArSeqProduct "a=1 d=1 n=999 p=8 t=1; n=9\_999; n=99\_999; n=999\_999"
* **Windows CMD**
* ArSeqProduct a=1 d=1 n=999 p=8 t=1; n=9\_999; n=99\_999; n=999\_999

**2) Large step (significant differences)**

* **10K terms spaced by 1000** starting at 1
* ./ArSeqProduct "a=1 d=1000 n=9\_999 p=8 t=2"
* **Start far from zero with a million‑scale step**
* ./ArSeqProduct "a=1\_000\_000\_000\_000 d=1\_000\_000 n=9\_999 p=16 t=2"

**3) Batch benchmark in one line (parameter persistence demo)**

* Starts with factorial‑like run, then only changes what’s needed each segment
* ./ArSeqProduct "a=1 d=1 n=99\_999 p=12 t=1; n=199\_999; d=2 n=99\_999; a=3 d=10 n=200\_000; p=8; d=1\_000 n=50\_000"

Explanation:

* + Run 1: a=1 d=1 n=99\_999 p=12 t=1
  + Run 2: only n changes → factorial with bigger n
  + Run 3: change d and n; a/p/t persist
  + Run 4: change a,d,n together
  + Run 5: change p only (same a,d,n as run 4)
  + Run 6: change d,n only

**4) Stress mixes (quick sweep)**

./ArSeqProduct "a=1 d=1 n=9\_999 p=8; d=2 n=4\_999; d=10 n=999; d=100 n=999; a=10\_000 d=10\_000 n=2\_000 p=16"

**Output**

* Prints a short header with resolved parameters per run.
* Shows timing for **compute parts** and **combine** (balanced tree), and total.
* Periodic log records are emitted every t minutes when t > 0.

**Notes & edge cases**

* If any AP term is 0, the product is 0 (fast check available).
* Negative terms are supported (sign is handled by BigInteger).
* If n is too small relative to p, the app may reduce workers or switch to a simpler path.

**Building**

Standard .NET build (e.g., dotnet build -c Release). The executable is typically in bin/Release/....

**License**

TBD (draft phase).