Performance Comparison: Fast Copy vs. Normal Copy

1. Overview

In performance-critical or simulated execution environments, the cost of repeated loop control checks (branching) can significantly impact execution time. The objective of this study is to compare two data-copying strategies:

- **Normal Copy** – a straightforward, readable approach using a standard loop
- **Fast Copy** – a Duff's Device–style loop-unrolled approach that minimizes control checks by copying data in fixed-width groups of eight elements

2. Implementation Codes

2.1 Normal Copy (General / Pythonic Approach)

This method uses a standard counting loop. It is simple and readable but performs a loop condition check for every element transferred.

```
def gen_copy(src, cnt):
    dest = [None] * cnt
    for i in range(cnt):
        dest[i] = src[i]
    return dest
```

2.2 Fast Copy (Loop Unrolling – Duff's Device Style)

This method reduces loop overhead by copying elements in **groups of exactly eight**. Any remaining elements are handled first using structured control flow rather than a compact loop.

```
def fast_copy(source, count):
    dest = [None] * count

    n = count // 8
    r = count % 8

    i = 0

    if r == 7:
        dest[i] = source[i]; i += 1
    if r >= 6:
        dest[i] = source[i]; i += 1
    if r >= 5:
        dest[i] = source[i]; i += 1
    if r >= 4:
        dest[i] = source[i]; i += 1
    if r >= 3:
        dest[i] = source[i]; i += 1
    if r >= 2:
        dest[i] = source[i]; i += 1
    if r >= 1:
        dest[i] = source[i]; i += 1

    while n > 0:
        dest[i]     = source[i]
        dest[i + 1] = source[i + 1]
        dest[i + 2] = source[i + 2]
        dest[i + 3] = source[i + 3]
        dest[i + 4] = source[i + 4]
        dest[i + 5] = source[i + 5]
        dest[i + 6] = source[i + 6]
        dest[i + 7] = source[i + 7]

        i += 8
        n -= 1

    return dest
```

2.3 Sample Execution

```
source = list(range(1, 21))
output = fast_copy(source, 20)
print(output)
```

Output

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

3. Comparison of Execution Logic

Feature	Normal Copy	Fast Copy
Logic type	Compact iteration	Loop unrolling
Control checks	1 per element	1 per 8 elements
Remainder handling	Implicit	Explicit if-sequence
Assignments	One at a time	Eight per loop
Branching	Minimal	Multiple conditional checks

4. Comparative Table

Aspect	normal_copy	fast_copy
Loop iterations	n	n / 8
Assignments per loop	1	8
Branching	No	Yes (if checks)
Code simplicity	Very simple	Complex
Readability	High	Low
Exam friendliness	⭐⭐⭐⭐⭐	⭐⭐