Part A5 – Push/Pop Operations

Both push\_front and pop\_front operations run in O(1) time for singly linked lists and circular singly linked lists because they only modify a few pointers. The measured times remain nearly constant as ( n ) increases, confirming constant-time behavior. Small differences appear due to memory allocation or CPU timing noise.

Part A6 – Memory Audit and Free

The audit (counting nodes) requires traversing all ( n ) elements, producing an O(n) time pattern — the measured time grows roughly linearly with list size. The free operation also touches and deletes each node, showing similar ( O(n) ) scaling. Circular and non-circular lists have nearly equal times because both traverse every node once; the difference comes mainly from the extra check to detect the end in the circular version.

## Part A5 — Push/Pop Ends: SLL vs DLL, Head vs Head+Tail

### 📌 Goal:

Implement:

\* SLL head only

\* SLL head+tail

\* DLL head+tail

Test:

\* Push front

\* Push back

\* Pop front

\* Pop back

Under mix like 70% push\_back, 30% pop\_front.

### ✅ Prediction:

\* Head-only SLL → slow push\_back (O(n))

\* Head+tail SLL → fast push\_back (O(1))

\* DLL → fast both ends (O(1) for all)

### 🧪 Measurements:

| Structure | Time (ms) |

| --------------- | --------- |

| SLL (head only) | |

| SLL (head+tail) | |

| DLL (head+tail) | |

### 🧠 Explanation:

Tail pointer is essential for fast push\_back. DLL also allows efficient pop\_back, which SLL can't do without O(n) search.

---

## Part A6 — Memory Overhead Audit

### 📌 Goal:

Compare memory per node and allocation time:

\* SLL: 1 pointer

\* CSLL: 1 pointer

\* DLL: 2 pointers

### ✅ Prediction:

\* DLL has highest memory per node.

\* Allocation time differences may appear at large n.

### 🧪 Measurements:

| Type | Pointers/Node | Bytes/Node | Total Bytes (n=1e5) | Allocation Time (ms) |

| ---- | ------------- | ---------- | ------------------- | -------------------- |

| SLL | 1 | ~8 | | |

| CSLL | 1 | ~8 | | |

| DLL | 2 | ~16 | | |

### 🧠 Explanation:

Time–space tradeoff: DLL costs more memory but enables efficient operations (like erase and pop\_back). For frequent mid-list deletions, DLL is often preferred despite its space cost.\* Push front

\* Push back

\* Pop front

\* Pop back

Under mix like 70% push\_back, 30% pop\_front.

### ✅ Prediction:

\* Head-only SLL → slow push\_back (O(n))

\* Head+tail SLL → fast push\_back (O(1))

\* DLL → fast both ends (O(1) for all)

### 🧪 Measurements:

| Structure | Time (ms) |

| --------------- | --------- |

| SLL (head only) | |

| SLL (head+tail) | |

| DLL (head+tail) | |

### 🧠 Explanation:

Tail pointer is essential for fast push\_back. DLL also allows efficient pop\_back, which SLL can't do without O(n) search.

---

## Part A6 — Memory Overhead Audit

### 📌 Goal:

Compare memory per node and allocation time:

\* SLL: 1 pointer

\* CSLL: 1 pointer

\* DLL: 2 pointers

### ✅ Prediction:

\* DLL has highest memory per node.

\* Allocation time differences may appear at large n.

### 🧪 Measurements:

| Type | Pointers/Node | Bytes/Node | Total Bytes (n=1e5) | Allocation Time (ms) |

| ---- | ------------- | ---------- | ------------------- | -------------------- |

| SLL | 1 | ~8 | | |

| CSLL | 1 | ~8 | | |

| DLL | 2 | ~16 | | |

### 🧠 Explanation:

Time–space tradeoff: DLL costs more memory but enables efficient operations (like erase and pop\_back). For frequent mid-list deletions, DLL is often preferred despite its space cost.