Student ID: 113153 Student Name: 蘇苡瑄

Course: Data Structures (CSE CS203A)

Assignment III: Linked List Selection Sort

140

Due date: 2025.10.20 10:00:00

Important Notice: You must print this assignment (Student Worksheet Companion) and **write your answers by hand with a pen**.

Goal

Practice representing data using a **linked list**, apply the **selection sort** algorithm step by step, and analyze the **differences between array and linked list** implementations in terms of traversal, exchange cost, and computational overhead.

Given Data

Input integers: 60, 24, 15, 42, 20, 11, 90, 8

Background

Linked List: A linked list is a sequence of nodes stored in **non-contiguously memory**. Each node contains a **value field** and a **pointer field** that links to the next node. Traversal is sequential O(n), but **insertion** or **deletion** (given a node) is O(1).

Selection Sort Algorithm: Selection sort is an **in-place comparison-based** algorithm that divides the structure into **sorted** and **unsorted** portions. For each position i, it finds the minimum element in the remaining unsorted portion and swaps it with the element at position i.

Selection Sort Algorithm

Basic Idea: For each position i from 0 to n-2, find the minimum element in the unsorted suffix (from position i+1 to n-1) and swap it into position i.

Pseudocode:

```
SELECTION-SORT-LIST(head):
    i = head
    while i != NULL and i->next != NULL:
        min_node = i
        j = i->next
        while j != NILL:
            if j->val < min_node->val:
```

Student ID: 113157 3 Student Name: 蘇苑瑄

```
            min_node = j
        j = j->next
    if min_node != i:
        swap i->val, min_node->val
    i = i->next
```

Tasks

Approach

Represent the integers in a singly linked list data structure and apply the selection sort algorithm step by step. In this pseudocode, only swap the values not modify the next pointer to complete the sorting algorithm

Analysis: Compare Array (Assignment II) vs Linked List in terms of exchanges, traversal, overhead, and visualization.

Required Answer Parts

A1. Draw the visual representations of the node in the singly linked list structure

A2. Populate the linked list with the given integers and add appropriate annotations (head pointer, next pointers and values)

A3. The first step is the demo step for you to complete the following three steps of the selection sort algorithm with detailed traces

A4. Discussion: Analyze and compare Array vs Linked List.

Deliverables

Print the **Student Worksheet Companion and submit** handwritten worksheet containing:

- Linked list structure drawings (A1)
- Populated linked list (A2)
- Step-by-step traces of the first 3 steps (A3)
- Comparative discussion (A4)

Evaluation (100 pts)

- Representations (A1): 5 pts
- Populated linked list (A2): 32 pts
- Linked list selection sort steps (A3): 25 pts
- Discussion quality (A4): 40 pts

Student ID: 113123 Student Name: 蘇芃瑄

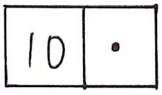
Course: Data Structures (CSE CS203A)

Assignment III: Linked List Selection Sort

Student Worksheet Companion

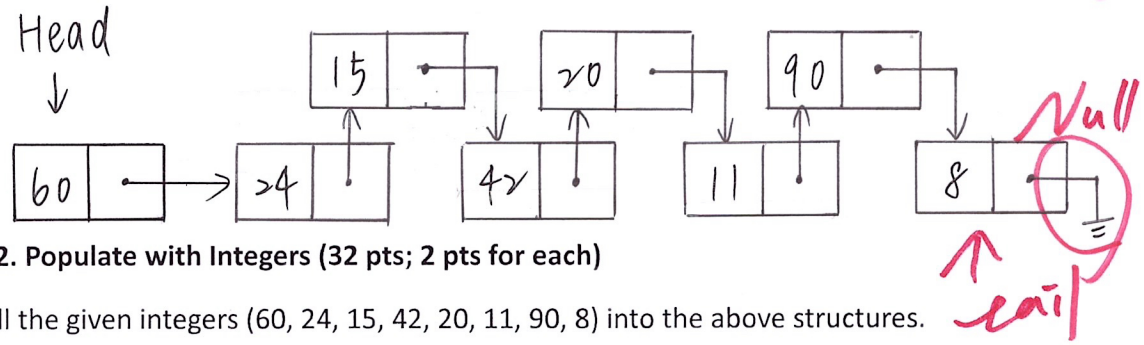
A1. Linked List Representation Drawing (5 pts)

a. (2 pts) Instructions: Draw a visual representation of a single node with next pointer that contains the initialized integer 10



b. (3 pts) Linked list representation with the given integers (Hint: For safety and clarity, include identifiable head and tail nodes)

Example: the input integers are (10, 20) and linked list representation will be [10 | •] → [20 | •] →



A2. Populate with Integers (32 pts; 2 pts for each)

Fill the given integers (60, 24, 15, 42, 20, 11, 90, 8) into the above structures.

Annotate:

Node #	Value	Next Pointer
1	[60]	→ Node [2]
2	[24]	→ Node [3]
3	[15]	→ Node [4]
4	[42]	→ Node [5]
5	[20]	→ Node [6]
6	[11]	→ Node [7]
7	[90]	→ Node [8]

Student ID: 113153	Student Name: 蘇莌這

8	[8]	→ [NULL]

A3. Selection Sort – First Three Steps (45 pts; 15 pts for each step)

Step Trace Table (Linked list):

Step 1 is the example to help you to complete step 2 to 4.

Step 1 (i = head = 60): Traverse list to find minimum value 8 → call swap function Yes; swap (60, 8).

head → [8|•] → [24|•] → [15|•] → [42|•] → [20|•] → [11|•] → [90|•] → [60|NULL]

Step 2 (i = 24): Minimum value [11] → call swap function (Yes) / No; swap ([24], [11]).

head → [8|•] → [11 |•] → [15 |•] → [42 |•] → [20 |•] → [24 |•] → [90 |•] → [60 |NULL]

Step 3 (i = 15): Minimum value [15] → call swap function Yes / (No); swap ([15], [15]).

head → [8|•] → [11 |•] → [15 |•] → [42 |•] → [20 |•] → [24 |•] → [90 |•] → [60 |NULL]

Step 4 (i = 42): Minimum value [20] → call swap function (Yes) / No; swap ([42], [20]).

head → [8|•] → [11 |•] → [15 |•] → [20 |•] → [42 |•] → [24 |•] → [90 |•] → [60 |NULL]

Student ID: 113153123 Student Name: 蘇芃誼

A4. Discussion (68 pts)

Guiding Questions:

- How many swaps/exchanges are performed?
- How expensive is traversal for arrays vs. linked lists?
- What memory/overhead differences do you see?
- Which representation is easier to visualize?
- Which would you choose for implementing selection sort and why?

Time complexity comparison (14 pts, 1pt for each)

Aspect / Operation	Array	Linked List	Explanation
Access Element 存取元素	(1)	(2)	Array allows direct indexing; linked list needs traversal. 直接索引 v.s. 遍歷
Find Minimum 找min值	(3)	(4)	Both must scan all remaining elements/nodes.
Swap Operation swap操作	(5)	(6)	In array, swap by indices; in linked list, swap node values. 索引swap v.s. 節點swap
Traversal Between Elements 元素間遍歷	(7)	(8)	Linked list traversal requires pointer navigation.
Overall Time Complexity (Selection Sort) 時間複雜度	(9)	(10)	Both involve nested traversal to find minima; linked list adds traversal overhead.
Space Complexity 空間複雜度	(11)	(12)	Both sorts are in-place if swapping values, not nodes.
Implementation Overhead 實作 overhead	(13) Low or Moderate	(14) Low or Moderate	Linked list needs pointer operations and careful null checks.

Student ID: 113153 Student Name: 蘇苑瑄

(1)	O(1)	(2)	O(n)
(3)	O(n)	(4)	O(n)
(5)	O(1)	(6)	swap value: O(1) swap nodes: O(n)
(7)	O(n)	(8)	O(n)
(9)	O(n²)	(10)	O(n²)
(11)	O(1)	(12)	O(1)
(13)	Low	(14)	Moderate.

-2

Student ID: 113153 Student Name: 蘇芃瑄

Characteristics (54 pts, 3 pts for each)

Aspect	Array	Linked List
Storage 存儲	(1)	(2)
Access 存取	(3)	(4)
Extra Variables 額外變數	(5)	(6)
Traversal 遍历	(7)	(8)
Overhead	(9)	(10)
Visualization 可視化	(11)	(12)
Swaps	(13)	(14)
Flexibility 彈性	(15)	(16)
Overall	(17)	(18)

(1)

Contiguous memory

(2)

Non-contiguous memory

(3)

Random access (O(1))

Student ID: 1131523 Student Name: 蘇芃道

(4) sequential access (O(n))

(5)

Only array index needed

(6)

Requires pointers (head, next)

(7)

Faster (index increment)

(8)

sequential (pointer following)

(9)

Minimal space overhead

(10)

Extra space for pointers

Student ID: 113 1523 Student Name: 蘇芃瑄

(11)

Easier, single contiguous block

(12)

Harder, seperate nodes and arrows

(13)

O(1), value assignment

(14)

O(1) value exchange / O(n) node exchange

(15)

Fixed size (traditional arrays) / Dynamic size (dynamic arrays)

(16)

Dynamic size, efficient insertions or deletions

Student ID: 1131523 Student Name: 蘇芃萱

(17)

Array is better for stable data and fast access.

2

(18)

Linked List is better for frequent structural changes.

2

-6

教授(助教)抱歉，因為先寫在平板上再謄寫至紙本因此漏抄一頁，造成批改不便真的非常抱歉！

沒事！