Student ID: 1121411 Student Name: 曾詠琪

Course: Data Structures (CSE CS203A)

Assignment III: Linked List Selection Sort

Student Worksheet Companion

144

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

[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 | •] →

Head → [60|·] → [24|·] → [15|·] → [42|·] → [20|·] → [11|·] → [90|·] → [8|null]

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: Head → [60|·] → [24|·] → [15|·] → [42|·] → [20|·] → [11|·] → [90|·] → [8|null]

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: 112141 Student Name: 詹詠璞

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]

i = i → next = 24 ?

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 ([____], [____]).

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: 112141 Student Name: 李詠琪

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	(1) O(1)	(2) O(n)	Array allows direct indexing; linked list needs traversal.
Find Minimum	(3) O(n)	(4) O(n)	Both must scan all remaining elements/nodes.
Swap Operation	(5) O(1)	(6) O(1)	In array, swap by indices; in linked list, swap node values.
Traversal Between Elements	(7) O(n)	(8) O(n)	Linked list traversal requires pointer navigation.
Overall Time Complexity (Selection Sort)	(9) $O(n^2)$	(10) $O(n^2)$	Both involve nested traversal to find minima; linked list adds traversal overhead.
Space Complexity	(11) O(1)	(12) O(1)	Both sorts are in-place if swapping values, not nodes.
Implementation Overhead	(13) Low or Moderate	(14) Low or Moderate	Linked list needs pointer operations and careful null checks.

Student ID: 1121411 Student Name: 曾詠琪

(1)	O(1)	(2)	O(n)
(3)	O(n)	(4)	O(n)
(5)	O(1)	(6)	O(1)
(7)	~~O(n)~~	(8)	O(n)
(9)	$O(n^2)$	(10)	$O(n^2)$
(11)	O(1)	(12)	O(1)
(13)	Low.	(14)	Moderate

1

Student ID: 112l4ll Student Name: 曾詠琪

Characteristics (54 pts, 3 pts for each)

Aspect	Array	Linked List
Storage	(1) contiguous	(2) non-contiguous.
Access	(3) Direct access with index	(4) Sequential access
Extra Variables	(5)	(6)
Traversal	(7) Direct	(8) Sequential
Overhead	(9)	(10)
Visualization	(11)	(12)
Swaps	(13)	(14)
Flexibility	(15)	(16)
Overall	(17)	(18)

(1) 連續配置，使用連續記憶空間存資料

(2) 非連續配置，每個 node 分散在記憶體中

(3) O(1)，可直接透過 index 存取

Student ID: 112414 Student Name: 曾詠琪

(4) O(n)，需要從頭依序走訪才能找到目標

(5) 較少，只需要 loop 內的 counter，例如 ~ 指出要換的
j 指出要被換的

(6) 較多，每個節點需額外存 pointer，例如 Head，sorted tail...
指出前一個或下一個 node

(7) 較快，可直接透過 index 快速移動

(8) 較慢，需要透過 pointer 一個個節點移動

(9) 較少，無額外的記憶體負擔

Student ID: 112411 Student Name: 曾詩琪

(10) 較多，需要額外記憶體存 pointer

(11) 較簡單，可直接用整齊的一列表示

(12) 較複雜，需畫出 node，node 之間的 pointer

(13) 簡單、快速，可直接用 index 交換 value

(14) 同樣快速，找到目標 node 後便可直接換 value
但如果要整個 node 交換就會需記錄 node 之間的 pointer 並更新之間的指向關係，較複雜

(15) 固定 size，宣告後大小固定，無法動態調整

Student ID: 112141 Student Name: 黃詠琪

(16) 動態 size，可依需求 insert 或 delete

(17) 實作簡單，適合用於已知資料大小、需要頻繁隨機存取

(18) 實作較複雜，需額外處理 pointer，適合用於需要頻繁 insert 或 delete 資料時。

-4