University of Central Lancashire
UCLan

Tutorial 2.3 discussion

CO2412

14th December 2021

Problem 2.3.1: Performance of doubly linked lists

A list with *n* elements is given.

Iterate over the whole list, and for each element:

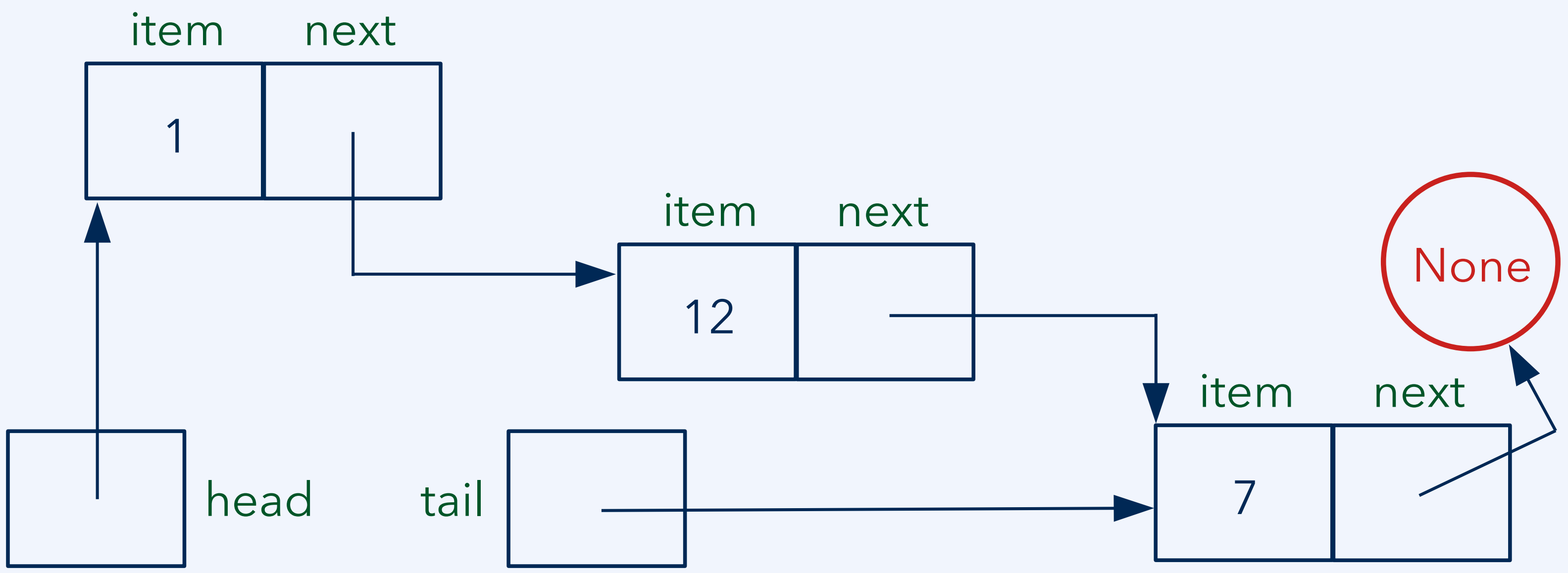
- If it is a multiple of 3, delete it from the list;
- If it has a remainder of 1 upon division by three, do nothing;
- If it has a remainder of 2, insert a copy of the element right next to it.

In this way, *e.g.*, [19, 12, 20, 12, 4] is modified to become [19, 20, 20, 4].

Problem 2.3.1: Performance of doubly linked lists

In a **singly linked list**, each node contains a data item and a reference (or pointer) to the **next node**. This facilitates traversal in **one direction**, namely forward, and **inserting** a new data item **after** any given node, in constant time.

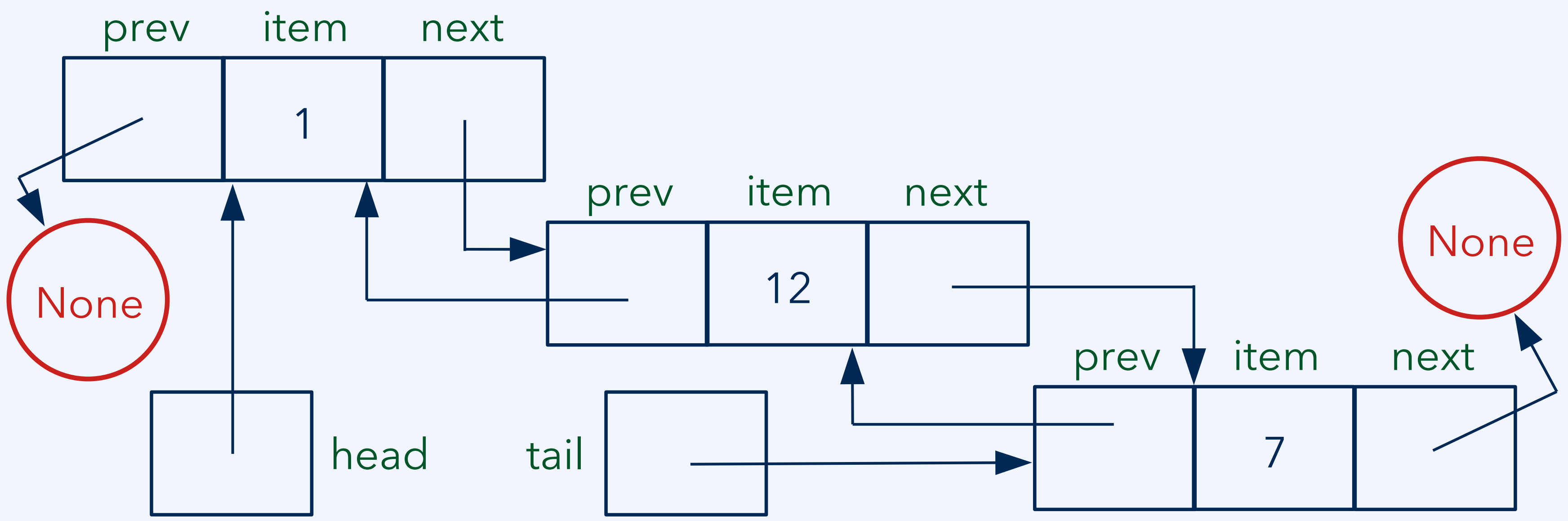
Singly linked lists require two variables per data item (item and next).



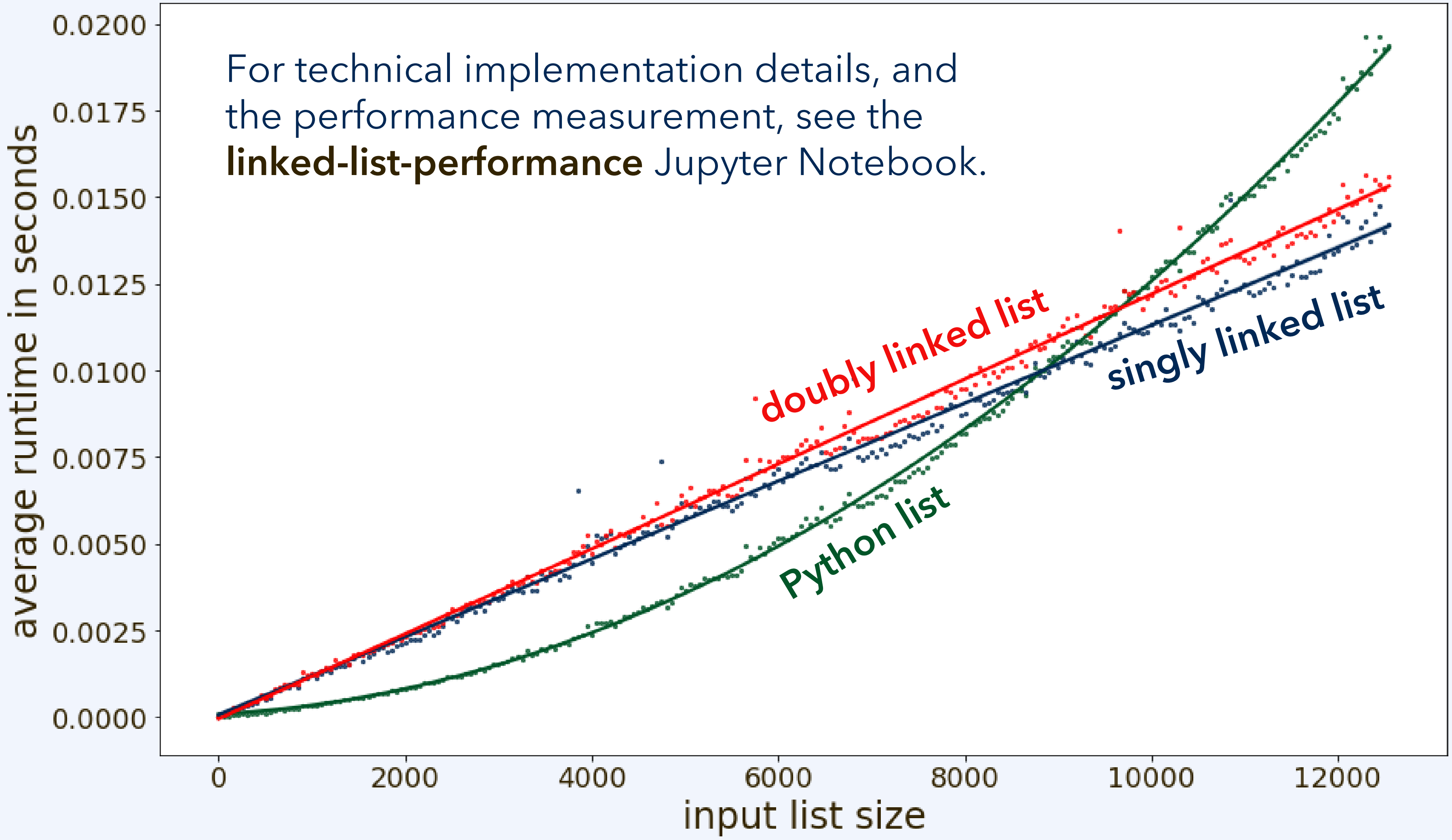
Problem 2.3.1: Performance of doubly linked lists

In a **doubly linked list**, each node additionally contains a reference to the **previous node**. This facilitates traversal in **both directions and inserting** a new data item **before** any given node (rather than only after it), all in constant time.

Singly linked lists require two variables per data item (item and next).
Doubly linked lists require three variables per data item (prev, item, and next).



Problem 2.3.1: Performance of doubly linked lists



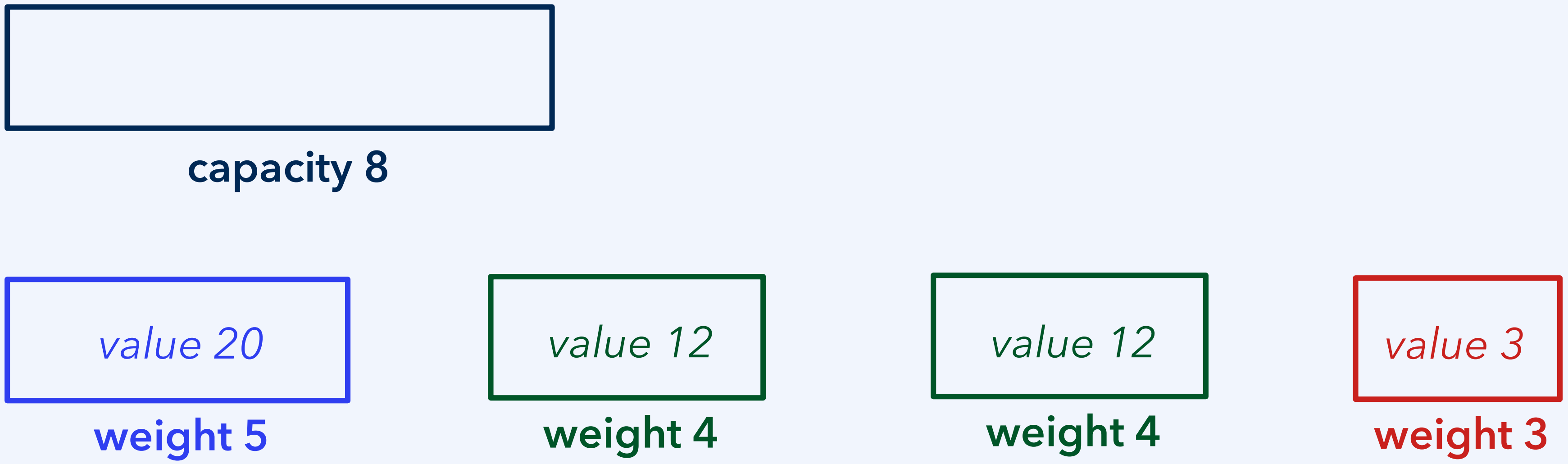
For technical implementation details, and the performance measurement, see the **linked-list-performance** Jupyter Notebook.

Problem 2.3.2: Dantzig's algorithm

Greedy algorithm for the knapsack problem:

- There is a limited capacity c.
- Loadable items each have a weight *w*[*i*] and a value *v*[*i*].
- Dantzig's algorithm selects them in descending order of *v*[*i*] / *w*[*i*].
- The algorithm terminates when no more items fit into the capacity.

The question was: Does this algorithm always determine the best solution?

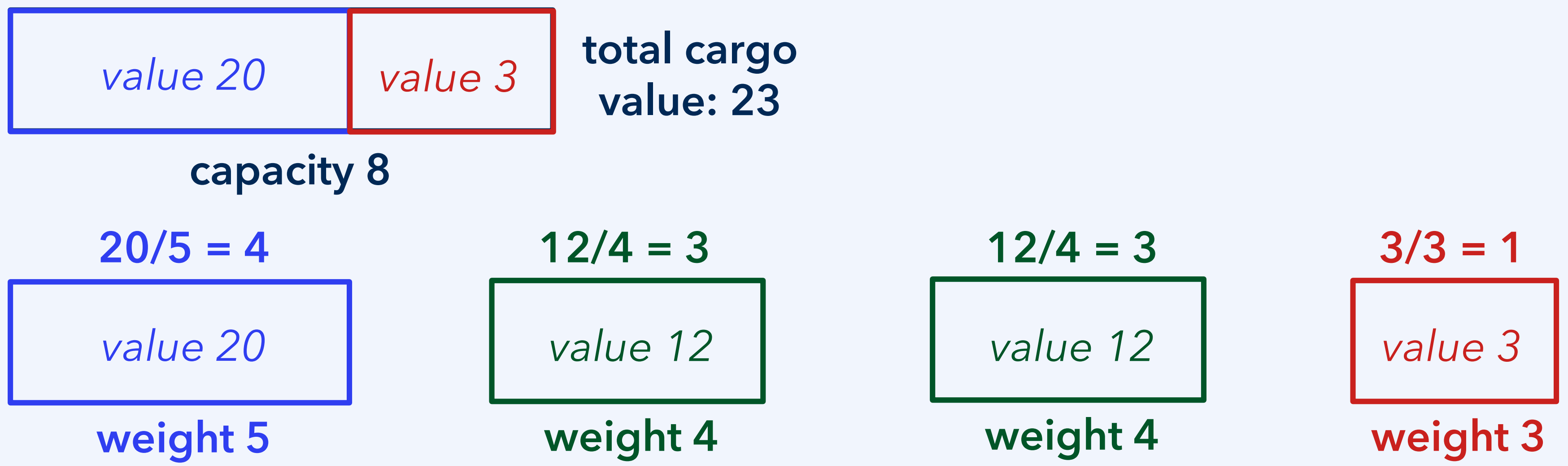


Problem 2.3.2: Dantzig's algorithm

Greedy algorithm for the knapsack problem:

- There is a limited capacity c.
- Loadable items each have a weight $w[i]$ and a value $v[i]$.
- Dantzig's algorithm selects them in descending order of $v[i]$ / $w[i]$.
- The algorithm terminates when no more items fit into the capacity.

The question was: Does this algorithm always determine the best solution?

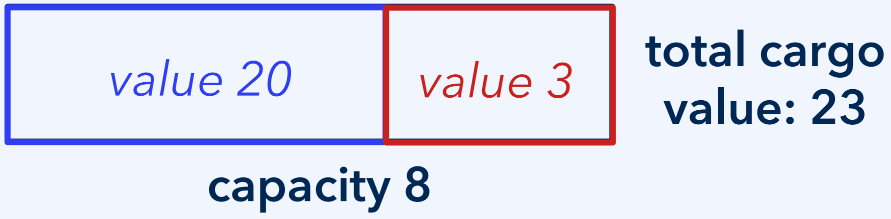


Problem 2.3.2: Dantzig's algorithm

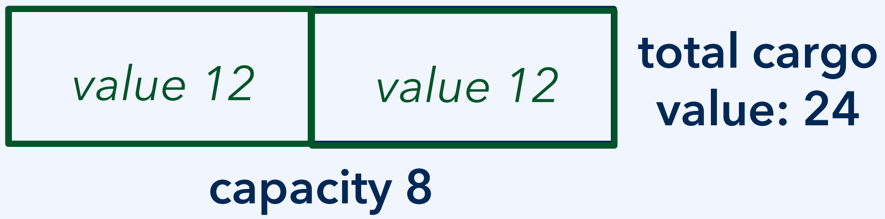
Greedy algorithm for the knapsack problem:

- There is a limited capacity c.
- Loadable items each have a weight *w*[*i*] and a value *v*[*i*].
- Dantzig's algorithm selects them in descending order of *v*[*i*] / *w*[*i*].
- The algorithm terminates when no more items fit into the capacity.

The question was: Does this algorithm always determine the best solution?



Optimal solution, not found by Dantzig's algorithm:


value 12
value 12
total cargo value: 24
capacity 8