COMP2521 25T2
Revision Lecture (Based on Requests)

Sim Mautner

cs2521@cse.unsw.edu.au

What if my question isn't answered today?
Hamiltonian Path Algorithm
Doubly Linked Lists
Specific Student Questions
Exam Questions

What if my question isn't answered today?

- Lots of great questions were asked in the poll
- We won't have time to answer all of them
- For the most general questions, I will produce a reference guide indicating where in the lectures you can find your answers
- For the more specific questions (and if you have some after watching the lectures), I recommend organising a consultation time with me. That way I'll be able to answer your question(s).

Alternatively, if there is high enough demand (eg at least 10 students committing to attending live, online), I'll gladly conduct a follow-on lecture in the same or similar format early next week.

Hamiltonian Path

Clarifying the task:

- Hamiltonian - visit each vertex exactly once
  - Not all edges need to be ”drawn over”
  - Edges can be ”drawn over” more than once

Hamiltonian Path

Approach:

- DFS throughout the graph
- Record which nodes have been visited
- As we unwind (backtrack in the DFS) (before we find a HP) record those nodes as "not visited" as, if we find a path, those nodes will not be on our successful path
- Don't visit a node more than once (this is default in DFS)
- If at the end of the DFS, if all nodes have been visited, then we have a Hamiltonian Path

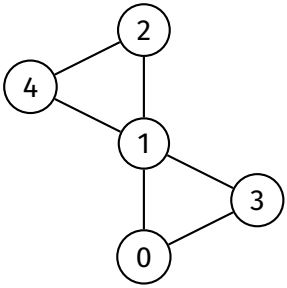
Hamiltonian Path

One more thing to remember:

- Whether a Hamiltonian Path is found or not depends on which node we start on
- To address this, we need to repeat the above process using each node in the graph as a starting node, until we find a Hamiltonian Path
- Once we have done this (DFS starting on each node), if no Hamiltonian Path has been found, then there is no Hamiltonian Path in the graph.

Hamiltonian Path

Why set `visited[`v`]` to false at the end of `dfsHamiltonianPath`?
Why start DFS on every node, not just one?



Doubly Linked Lists - Benefits

Why might we want doubly linked lists?

- It allows us to move forwards and backwards within a list.
- Depending on the implementation, it can allow easier access to items towards the end of the list.
- Inserting and deleting from the middle of the list (eg ordered lists) in particular can be considered ”less fiddly”

Doubly Linked Lists - Implementation


```
struct node {
    Item data;
    struct node *prev;
    struct node *next;
};

struct list {
    struct node *head;
    struct node *tail;
};
```

Doubly Linked Lists - Inserting

- At the start
- At the end
- In the middle

Doubly Linked Lists - Deleting

- From the start
- From the end
- From the middle

How to decide whether to use adj matrix, or adj list when answering a graph question (exam style)

Best way to make use of the Data Structures and Algorithms Manual for exam style questions

Deriving time complexity from written code.
Let's look at Q1c in
`https://cgi.cse.unsw.edu.au/~cs2521/25T2/past-exam/22T1`.

More theory exam questions:
Let's look at Q3a,b, Q4, Q5b in
`https://cgi.cse.unsw.edu.au/~cs2521/25T2/past-exam/22T1`.

Programming exam question(s):
Let's look at Q9 or Q10 in
`https://cgi.cse.unsw.edu.au/~cs2521/25T2/past-exam/22T1`.

Cheapest Flights Within K Stops leetcode:
`https://leetcode.com/problems/cheapest-flights-within-k-stops/description/`