COMP2521 19T0
Week 4, Thursday: Graphic Content (III)!

Jashank Jeremy

jashank.jeremy@unsw.edu.au

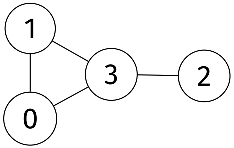
computability
directed graphs

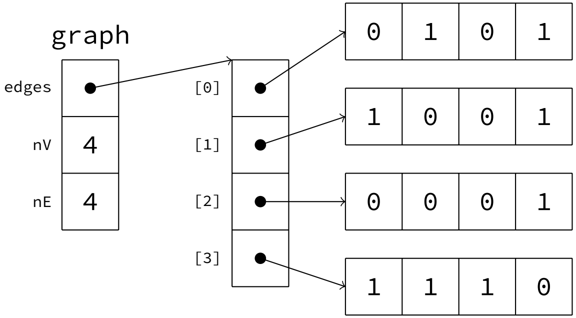
Graph Representation

Recap: Ways of Representing Graphs

Adjacency Matrices

```
struct graph {
    size_t nV, nE;
    bool **matrix;
};
```

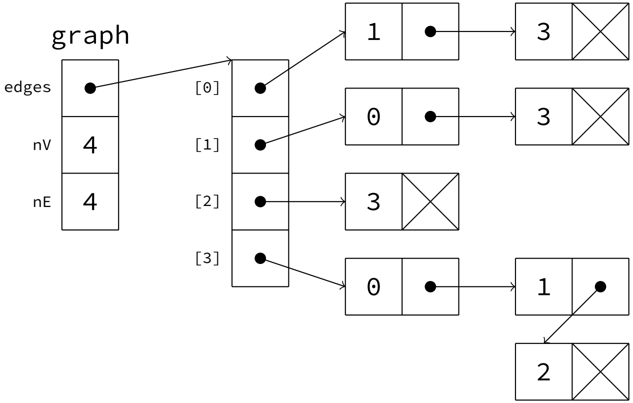




Recap: Ways of Representing Graphs

Adjacency Lists

```
typedef
    struct adjnode
    adjnode;
struct graph {
    size_t nV, nE;
    adjnode **edges;
};
struct adjnode {
    vertex w;
    adjnode *next;
};
```



Recap: Ways of Representing Graphs

Edge Lists

```
struct graph {
    size_t nV, nE;
    edge *edges;
};
```

v	w
0	1
0	3
1	3
2	3

Recap: Ways of Representing Graphs

Linked Data Structure

```
typedef struct vertex {
    Item it;
    size_t degree;
    vertex *neighbours;
} vertex;

struct graph {
    size_t nV, nE;
    vertex *root;
};
```

Recap: Ways of Representing Graphs

Comparison

	matrix	**adj.list**	**edge list**	**node links**
space	V^2	$V + E$	E	$V + E$
initialise	V^2	V	1	V
destroy	V	E	E	$V + E$
insert edge	1	V	1	E
find/remove edge	1	V	E	E
is isolated?	V	1	E	1
degree	V	E	E	E
is adjacent?	1	V	E	E

Hamilton Paths and Tours

Hamilton Path:
a simple path connecting two vertices
that visits each vertex in the graph exactly once

Hamilton Tour:
a cycle
that visits each vertex in the graph exactly once

* * *

Given a list of vertices or edges, easy to check.
Given a graph … how do we know if one exists?
Do we have to find one? If so, how do we find one?

Hamilton Paths and Tours

Brute Force Is Best Force

IDEA brute force!
enumerate every possible path, and check each one.

hack a BFS or DFS to do it:
keep a counter of vertices visited in the current path;
only accept a path only if count is
equal to the number of vertices.

PROBLEM how many paths?
given a simple path:
no path from t to w implies no path from v to w via t...
so there's no point visiting a vertex twice on a simple search
... but that's not true for a Hamilton path!

Hamilton Paths and Tours

Computability

we must inspect every possible path in the graph.
in a complete graph, we have $V!$ different paths ($\approx (V/e)^V$)

there are well-known, well-defined subsets of this problem
which are easy to solve (Dirac, Ore) … but in general
this is a non-deterministic polynomial, or NP problem

Euler Paths and Tours

Euler Path:
a simple path connecting two vertices
that visits each edge in the graph exactly once
… exists iff the graph is connected
and has exactly two vertices of odd degree

Euler Tour:
a cycle
that visits each edge in the graph exactly once … exists iff the graph is connected
and all vertices are of even degree

… these can be found in linear time.

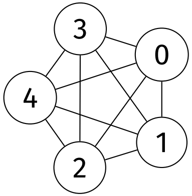
Graph Problems

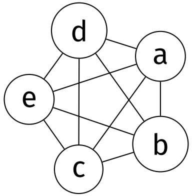
Tractable and Intractable

- tractable: can we find a simple path connecting two vertices in a graph?
  tractable: what's the shortest such path?
  intractable: what's the longest such path?
- tractable: is there a clique in a given graph?
  intractable: what's the largest clique?
- tractable: given two colours, can we colour every vertex in a graph such that no two adjacent vertices are the same colour?
  intractable: what about three colours?

Graph Problems

Bonus Round!





Graph isomorphism:
Can we make two given graphs identical by renaming vertices?

No general solution exists.
We don’t know if one can exist.

WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
...AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.
IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

xkcd 1425 “Tasks” // CC BY-NC 2.5