CS3000

5/31 - Weds

Admin

- Long HW3 due today 9pm
- Long HW4 out tom, due 6/6
- Tomorrow - fun optional recitation

Agenda

1. Amortized analysis
2. Aggregate analysis example
3. Graph representation

Recap

$$\sum_{i=0}^{n} 2^i = 1+2+4+\ldots+2^n = 2^{n+1}-1$$

(ex) $1+2+4+8+16 = 31 = 2^5-1$

2^0 ... 2^4

1. Amortized Analysis

$T(n)$ = # steps on input of size n

Sequence of operations ⇝ worst case

(ex) $\Theta(n)$

Traditional w.c. analysis { n of these operations ↳ $\Theta(n^2)$

Amortized version:

- sequence of operations
- worst case can't possibly happen every time
- count the # times the worst case happens over a sequence of n operations
- get a new total run-time, and per-operation run time

Trip to Data Structures/Systems

- let $A[1..n]$ be a new array (allocation)
- Insert at:
  - pos1
  - pos2
  - pos3
  - ⋮
  - pos n

  } each insert costs $\Theta(1)$

- Call insert again, now want position n+1

↳ need to allocate new array!

new array has size 2n

A [1.. 2n] ⟶ copy over elements from old array to new array

| cost: n+1 |

worst case

Traditional w.c. analysis:

$\Theta(n)$ in worst case for one insert

A sequence of n insert operations: $\Theta(n^2)$

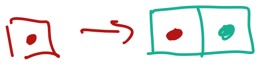
2. Aggregate Analysis

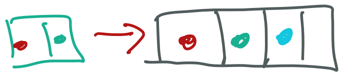
- simplification: don't worry about cost of allocation/deallocation

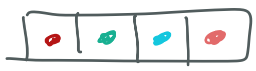
- aggregate: look at cost of all n operations, not just worst case of one at a time

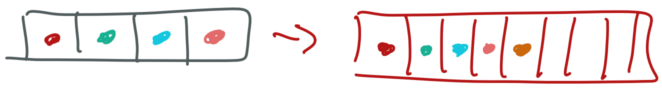
Start:

insert:  1st insert cost: 1

insert:  2nd insert cost: 2

insert:  3rd insert cost: 3

insert:  4th insert cost: 1

insert: 

5th insert cost: 5

i	1	2	3	4	5	6	7	8	9
cost	1	2	3	1	5	1	1	1	9
size	1	2	4	4	8	8	8	8	16

- worst case does happen (next worst case: when i = 17)
- but, not every single time!

In general: worst case when $i = 2^k + 1$ for some k

otherwise, cost is 1

Agg. analysis Add all the i's together when it costs i, and all the 1s together otherwise

- What's total cost of worst case?

$$\sum_{j=0}^{\lg n} 2^j = 2^{\lg n+1} - 1 = \Theta(n)$$

- what's total cost of non-worst case?

n - lg n of these

$\leq n$ of these

- In total over n operations?

$\leq \Theta(n) + n$

$= \Theta(n)$

- per operation?

$\Theta(1)$

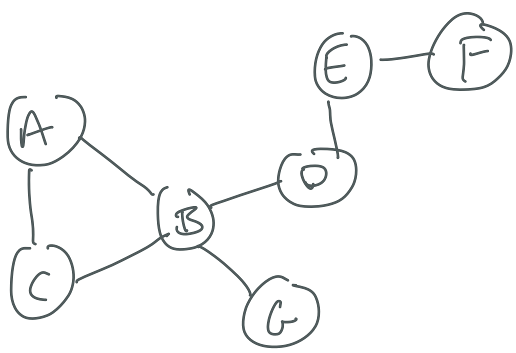
- Generalization: sequence of n operations in which the ith operation costs i if i is a power of 2, and 1 otherwise

10:41

2. Graph Intro

Assumptions:

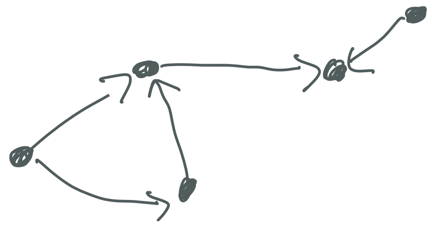
- we know graphs are important
- we know some stuff about graphs
  - ↳ focus on how we implement graph algorithms, run-time, space, etc.



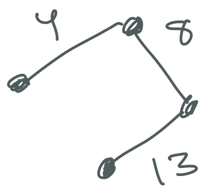
What do we know about this graph?

- has a cycle
- unweighted
- undirected
- 7 vertices/nodes
- 7 edges
- connected (path from u to v)
- break ties alphabetically
- deg (B) = 4

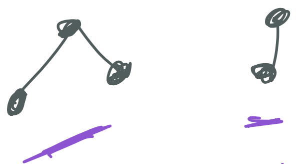
What new can we say about these?



- directed
- not strongly connected
- acyclic
- (DAG)

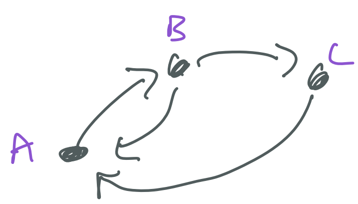


- weighted
- connected



connected components

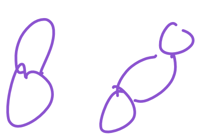
- unconnected
- each connected component is a tree



- strongly connected
  (u,v) and (v,u) path

(A,B) (B,A) (C,A)
(A,C) (C,B)

All graphs so far are Simple

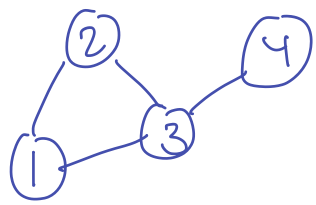


- no self loops
- no multi-edges

Representation

- Adjacency list
- Adjacency matrix

ex



Adj List

1: 2, 3

2: 1, 3

3: 1, 2, 4

4: 3

In pseudocode structure:

Func (G) → Function takes G as param

G.Adj[u] → the adj list for vertex u

⋮

for each vertex v in G.Adj[u]

↳ explore all of u's neighbors

Space Complexity

V = # vertices

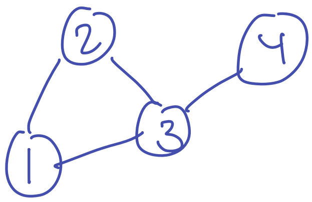
E = # edges

$V + 2 \cdot E$

$= \Theta(V + E)$

Pros/Cons of Adj list

- space complexity :)
- does (u,v) exist in the graph? :($\Theta(E)$



Adj Matrix

	1	2	3	4
1	0	1	1	0
2	1	0	1	0
3	1	1	0	1
4	0	0	1	0

In pseudocode structure

Func (G) → Function takes G as param

G.A[u][v] → value in adj. matrix for (u,v)

↳ 0 or 1 if edge from u to v

↳ assume that u, v can be numbers or vertex labels

Space Complexity

$\Theta(V^2)$ matrix is $V \times V$

↳ quadratic

Pros/cons

- space complexity :((unless dense graph)
  (unless small)
- is there an edge from (u, v)? :) G.A[u][v]
  $\Theta(1)$